

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Decision Model and Notation: state of the art and simplification algorithms“

verfasst von / submitted by

Galina Mikhnyova BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

UA 066 926

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Masterstudium Wirtschaftsinformatik

Betreut von / Supervisor:

Assistant Prof. Dipl.-Wirtsch.Inf Dr.rer.pol.
Dominik Bork

Acknowledgements

I would like to express my deepest gratitude to my professor, Mr. Assistant Prof. Dipl.-Wirtsch.Inf Dr.rer.pol. Dominik Bork, for his great support in the complete process, valuable feedback, and for his patience.

Thanks should also go to ao. Univ.-Prof. Dipl.-Ing. Dr. Eduard Mehofer for his support in the rather complicated beginning of this thesis journey. I am also thankful to the teaching staff of the University of Vienna for the knowledge I've gained during all my years at the University, which helped me to write this thesis.

I would be remiss in not mentioning my family, especially my role model - my mother, for creating the motivation and possibilities for me to begin this University journey abroad. Lastly, I would like to mention my great partner and my sweet dogs, who supported me and my mental health throughout this journey.

Thank you!

Abstract

Recently, more and more organizations have been trying to go through digital transformation in business processes. For this purpose, different Business Process Modeling tools, alongside the Business Process Modeling and Notation, are used. However, the decision logic in the models, represented by cascading gateways, often becomes challenging. This is when Decision Model and Notation (DMN) comes into play. Due to its narrow focus, DMN is less researched as BPMN. In the meantime, the use of DMN itself grows, and so does the complexity of the models. Especially for automatically generated models, there is a lack of solutions for analyzing and simplifying such models. This gap is the main problem this work aims to solve. The master thesis includes the research on DMN-tools; research status in the fields of verification, validation, and simplification of DMN models. Based on the study, the gaps in DMN simplification are identified, and a simplification algorithm is suggested. Current work includes an extension for an open-source tool (BPMN.io), which contains mechanisms and algorithms for automatic analysis and simplification of DMN models to increase the quality of the models.

Kurzfassung

In letzter Zeit versuchen immer mehr Organisationen, ihre Geschäftsprozesse digital zu transformieren. Zu diesem Zweck werden neben dem "Business Process Modeling and Notation" verschiedene Geschäftsprozessmodellierungswerkzeuge verwendet. Aber oft wird die Entscheidungslogik in den Modellen durch die komplexen Gateways dargestellt. Hier kommt "Decision Model and Notation" (DMN) ins Spiel. Aufgrund seines engen Fokus ist DMN weniger erforscht als BPMN. Inzwischen wächst der Einsatz von DMN selbst und damit auch die Komplexität der Modelle. Insbesondere für automatisch generierte Modelle fehlt es an Lösungen zur Analyse und Vereinfachung solcher Modelle. Diese Lücke ist das Hauptproblem, das diese Arbeit versucht zu lösen. Diese Masterarbeit beinhaltet die Forschung zu DMN-Tools; Forschungsstand in den Bereichen Verifikation, Validierung und Vereinfachung von DMN-Modellen. Basierend auf der Forschung werden die Lücken in der DMN-Vereinfachung identifiziert und ein Vereinfachungsalgorithmus vorgeschlagen. Aktuelle Arbeiten umfassen eine Erweiterung für ein Open-Source-Tool (BPMN.io), das Mechanismen und Algorithmen zur automatischen Analyse und Vereinfachung von DMN-Modellen enthält, um die Qualität der Modelle zu erhöhen.

Contents

Acknowledgements	i
Abstract	iii
Kurzfassung	v
List of Tables	ix
List of Figures	xi
List of Algorithms	xiii
Listings	xv
1 Introduction	1
1.1 Research question	1
1.2 Research methods	2
1.3 Target group and benefits	2
1.4 Thesis outline	3
2 Foundation	5
2.1 Modeling methods	5
2.2 DMN and usage of DMN	5
3 Related works	11
3.1 Surveys	12
3.2 Verification & Validation, DMN Simplification	13
3.3 Summary	14
4 Research methods	17
4.1 Design Science Research	17
4.2 Literature review	21
4.3 Systematic tool review	22
5 Survey of DMN tools	25
5.1 DMN tools overview	25
5.2 Evaluation criteria	29
5.3 Selected tools	33
5.4 Tool review process	35

5.5	Results discussion	43
6	DMN simplification	45
6.1	Rule-based systems	45
6.1.1	RBS Verification	46
6.1.2	Redundancy	47
6.1.3	Reduction	48
6.2	Reasoning	49
6.3	Geometric DMN simplification	49
6.4	Simplification algorithm	50
7	Implementation	53
7.1	Technology and functionality	53
7.2	Prototype	55
7.2.1	Test case generation	56
7.2.2	Test case execution	60
7.2.3	Simplification	60
7.3	Result discussion	62
8	Case-based evaluation	65
9	Conclusion	69
	Bibliography	71

List of Tables

3.1	Related works summary	15
5.1	DMN tools	28
5.2	Criteria catalog	29
5.3	Functional criteria (1/2)	37
5.4	Functional criteria (2/2)	38
5.5	Technical criteria	39
5.6	Feasibility and usability criteria	40
8.1	Test 1 results	68

List of Figures

2.1	DMN concepts [34]	7
2.2	DRD components [34]	9
4.1	DSR lifecycle [35]	18
4.2	ISDT lifecycle [3]	19
4.3	Literature filtering process	21
5.1	Test DMN model - DRD [39]	36
5.2	Test DMN model - decision table [39]	36
6.1	Expert system [24]	46
7.1	Project environment	54
7.2	Use-case diagram	56
7.3	UI overview	57
7.4	Prototype: class diagram	58
7.5	UI generate test cases	59
7.6	UI get complexity	59
7.7	Test case generation and execution	61
8.1	Test case 1	66
8.2	Test case 1 with added rules	67

List of Algorithms

1	DMN Backwards Chaining algorithm	61
---	--	----

Listings

7.1	Implementation of test case execution	60
7.2	Implementation of simplification algorithm	62

1 Introduction

The importance of the Decision Model and Notation is growing, and it is on its way to becoming the help tool of Business Process Modeling and Notation in realizing comprehensive business process representation by separating tasks. Thus, the flow of the actions can be represented using BPMN, while decision-making logic is externalized into Decision Model and Notation models (DMN).

Considering the described advantages of Decision Model and Notation, the main focus of the master thesis is on DMN itself (Decision Model and Notation), DMN-tools and the introduction of the DMN into the business flow through the model execution.

Alongside the well-known BPMN, the OMG (Object Management Group) introduced a so-called "Triple crown", which includes three notation standards for process improvement: BPMN, DMN, and CMMN (Case Management Model and Notation). The included notations can be used separately; nevertheless, they are compatible and can be used in combination to represent complex processes [33].

With the rise of machine learning and artificial intelligence, there is also a possibility to generate DMN models automatically. These can quickly become huge and unreadable due to the sizes of data sets. It brings to the question: Is there a possibility of automated analysis and simplification of models generated by a machine? But also, other complex models can be unnecessarily complicated. These topics were the main focus of the investigation.

The main gaps that were discovered through the research process and addressed in the current work are (1) the gaps in research on the current DMN tools and (2) simplification methods for the DMN models, especially analysis and simplification on the DRD level of the models.

This research adds value to the DMN research on DMN tools and verification, validation, and simplification of DMN models. The results can be used in further research and tool implementations. These results will support the modelers, especially in the fields of automatization of the processes. For example, the automatically generated complex models could be post-processed and simplified, using the suggested and implemented in the thesis functionalities. The benefits are further explained, at the end of the current Chapter.

1.1 Research question

The area of DMN has scientific research papers and includes various tools but is still not as explored as, for instance, the area of BPMN, and the following research questions arise:

1 Introduction

- RQ 1.1: What is the current research status on DMN?
- RQ 1.2: What is the current status of DMN tools?
- RQ 1.3: What can be done to enhance the value of DMN?

The current papers on DMN are reviewed and systematically analyzed to answer the first question. The results are represented in Chapter Related works.

Chapter 2 explains the main theoretical foundations of DMN.

Consequently, to answer the second question, a reliable survey is devised. The survey, performs as an overview of the existing DMN-tools by comparing them according to the own criteria catalog. The process of the tool review is conducted following the established Systematic Tool Review method, derived from the SLR (Systematic Literature review). The planning of STR, as well as Design Science Research methodology that was applied in the research, can be found in Research methods. The STR produced an informative survey of existing DMN tools. The process and results of the survey are shown in Chapter Survey of DMN tools.

The results of the first two research questions were analyzed to answer the third research question. As an area of DMN that needs to be enriched, mechanisms and algorithms of DMN verification, validation, and reduction were chosen. Proposed ideas and the implemented prototype are described in Chapters DMN simplification and Implementation.

1.2 Research methods

The first step is extensive literature research on the topic's current research status. It took place on Google Scholar and Publish or Perish, using a meaningful search query (see Chapter 4). According to the research, a criteria catalog could be created using an SLR (Systematic Literature Review), [11]. The criteria catalog included different characteristics of the tools on DMN-functionalities, technical criteria, availability, feasibility, and usability.

Alongside the first step, there will be research on the DMN and its importance, followed by a search for experts or expert interviews on this topic.

The second step is the research on existing DMN-tools. Here, the tools for the review were found and selected following the Systematic Tool Review, e.g., according to availability (using only open-source tools and trial versions of some proprietary tools that were received from the companies).

The third step is to implement an innovative prototype. For this, the Design Science Method [35] will be applied. Thus, the research on gaps and advantages of existing tools and BPMN.io extensions (especially DMN) will flow into creating an idea that can increase the added value of DMN usage. The focus here lies on automatic processing and analysis of DMN models [21].

1.3 Target group and benefits

The outcome of the master thesis can support the scientific research and highlight the research status and some weak points in knowledge in the field of decision modeling,

and DMN in particular. This research will also help other researchers and organizations in researching and implementing verification, validation, and simplification of DMN. It can help to choose open-source/academic tools for their needs. Besides, it will support DMN-software providers by identifying the gaps and recommending the best features for the tools reviewed. It will also give a general overview of the reviewed tools' current status and strong and weak sides. It will help DMN-modelers choose the best-fitting modeling software. Furthermore, it will bring some ideas for further work on DMN-tools, to enhance the benefits of the tool's usage. Additionally, the provided plugin will be able to support modelers in DMN-modeling and analysis.

1.4 Thesis outline

The thesis is divided into 9 chapters. The next Chapter, Chapter Foundation, includes the foundational knowledge about modeling methods and DMN. Chapter Related works shows the current status of the research, including the gaps in the existing research. The research is based on different DMN literature and surveys on DMN and modeling languages. Chapter Research methods includes a defined structured survey approach that is based on the adapted Design Science Research methodology and a systematic tool review that is adapted from the Systematic Literature Review. Chapter Survey of DMN tools contains a complex criteria catalog for the tool review, research on the existing DMN modeling tools, and a review of selected tools based on the criteria. The criteria catalog is based on the literature on tool review and criteria that comply with the main focus of the thesis. Chapter DMN simplification theory on DMN simplification, Rule-Based System; alongside with a suggested simplification algorithm for DMN models on Decision table level, as well as Decision Requirements Diagram. Chapter Implementation implementation of a DMN tool with suggested simplification functionality. A case study that shows the tool functionality is described in Chapter Case-based evaluation, and Chapter Conclusion suggests a few ideas for future works in this field.

2 Foundation

To conduct deep research on any topic, having basic knowledge of the subject is essential. This Chapter includes the definition of the modeling language and modeling procedure as the main concepts for further thesis research. It contains the modeling methods, DMN definitions, the basics of the modeling language, such as modeling components, and DMN specification.

2.1 Modeling methods

It is important to consider the modeling methods to start with the theoretical foundation. As determined in [25], modeling methods include the following components:

- modeling language,
- modeling procedure,
- mechanisms, and algorithms.

Thus, a modeling method includes the complete workflow from the language definition, usage of the language (modeling) to further work with the models, and usage of different functions. This Chapter includes the definition of the modeling language and modeling procedure. The following chapters include some existing mechanisms and algorithms for DMN and highlight the gaps and possible updates for the existing mechanisms. This should enrich the existing modeling methods for DMN.

2.2 DMN and usage of DMN

DMN is a modeling language designed under the support of the Object Management Group (OMG) and BPMN. DMN 1.0 was published by OMG in September 2015, and the current last version, DMN 1.3, in February 2021. The Beta version of DMN 1.4 was already published in March 2022, but only for informational purposes. Until the DMN standard, decision-making in a similar form took place within BPMN processes or was defined using common decision logic languages, such as PMML, and different specific solutions. It is created to support the clear specification of business decisions and rules and business automation. DMN combines business processes and decision logic. Technicians and stakeholders can use this notation to translate complex decision trees into readable tables and diagrams and vice versa: tables and diagrams that everyone can instinctively understand can represent complex decisions. The notation is made to

be easily understood without any training. The existing decision models consist of the Decision Requirements Level and the Decision Logic Level. The Decision Requirements Level includes the concept of the Decision Requirements Graph that consists of (1) the elements, such as Decision, Business knowledge, Input data, and others (which will be further explained); (2) connection rules and (3) converted into a Decision Requirements Diagram (DRD). The Decision Logic Level can be represented in several ways: as plain text, executable logic, decision tables, boxed expressions, or in a service invocation. For the execution, S-FEEL or FEEL-language can be used, as well as Java and PMML, where the tools less use the last ones. [34]

DMN can be widely used in different areas. Not dependent on the field where DMN is being applied, the DMN-models are used to model human decisions, conditions for automatization of decision-making, and its implementation. These aspects can be used separately and combined into a complete modeling method that includes every step from decision modeling to decision execution. [34]

Figure 2.1 nicely demonstrates the concept of DMN on an example from the DMN specification [34]. The model on the left is a BPMN model. One element of this BPMN, named "Decide routing", includes some decision logic. A DMN model on the right represents the logic. On the top, there is a Decision requirements level of the DMN. It includes an overview with three decision elements, three business knowledge elements and one input element. Decision Logic Level on this cutout shows only one table for the "Eligibility rules" element. This model decides on the routing for the application. Depending on the input (application, some eligibility rules, and other knowledge from the model), the user's eligibility and application risk will be decided, following the rules behind the decision elements. Then, the last decision will be made based on the output from the two bottom decisions. The output of the last decision, the routing, will be further used in the BPMN.

DMN-models are great stand-alone artifacts. Although, their benefits can be multiplied by using them in different scenarios. As the most common combination, DMN models can complement BPMN diagrams by describing the decision-making processes, as seen in Figure 2.1. [34]

As discussed in [6], DMN can also elevate the usability of BPMN. Some complex decisions in BPMN can be outsourced into a DMN diagram and only be used in the process diagram. This would simplify not only the Business Process Diagram but also present the decision part of the process much clearer in the Decision Model. The authors discuss the decoupling of decisions from processes and its benefits. The last is enhanced adaptability to changes and a lower risk of failure when modeling. [6]

DMN specification

As of today, there are already 4 DMN versions since the beginning: 1.0, 1.1., 1.2 and 1.3. The specification is being written and updated by several companies who are working on/with DMN. They include requirements for DMN models, explaining the visual appearance of models, decision semantics and attributes. The specification explains the DMN basics, the art of modeling decisions and other basic concepts. There are more

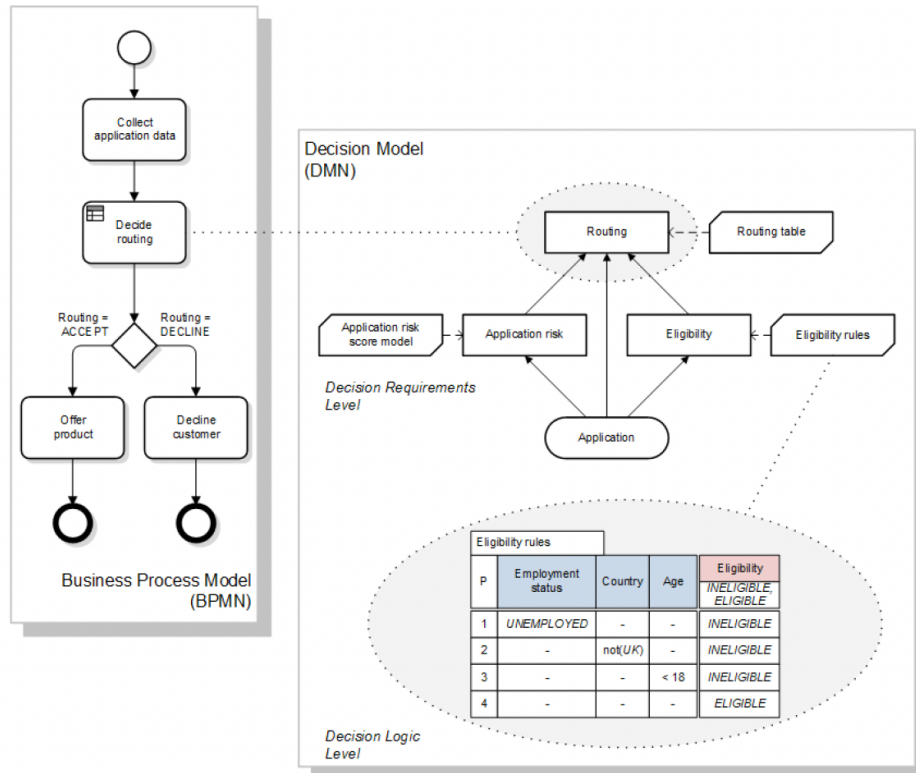


Figure 2.1: DMN concepts [34]

2 Foundation

requirements on notation, connections, decisions, metamodel, logic and complete decision tables. The specification also includes the expression language (S-FEEL and FEEL) that can be used in DMN, as well as some examples and diagram exchange formats and rules.[34]

DMN specification allows three implementation levels for DMN tools. These are so-called "Conformance levels". Many tools even have information on the conformance level of the tool on their web pages. This directly shows the user the complexity of the tool, including some functionality. A primary tool would have conformance level one. Level 2 includes the functionality of level 1 and more. And Level 3 respectively includes Level 1 and 2. [34]

The requirement can defer from DMN 1.0 to DMN 1.3. Here the last version is being examined. So, to conform to Level 1 following functionalities must be included:[34]

- DRG and DRD can be created The main DMN notation can be created. A tool should allow the creation of the following elements: decision, business knowledge models, input data, knowledge source, decision service, and connections between them. Some more non-strict requirements can be found in the specification. This should have a classic notation, as shown in 2.2.
- Decision logic can be included Here the focus is on the decision logic level and how this should be implemented. There should be several possibilities: literal expression, decision table and invocation. A literal expression can be a plain non-executable text, a first-order logic, a Java program, or a PMML document for Level 1.
- Decision tables can be created A decision table should include an item name, a list of input and output clauses, and a set of outputs, annotations and rules. Here several hit policies can be implemented, where the standard is "unique" (no overlap is possible), which always has to be implemented.

Level 2 conformance includes S-Feel support in addition to the Level 1 requirements. S-FEEL is a subset of FEEL. FEEL is a "friendly enough expression language"; thus, S-FEEL is a "Simple"-FEEL. This allows the execution of some simple unary expressions with inputs and outputs, including the arithmetic expressions, arithmetic comparison, intervals and string comparison. [34]

Level 3, accordingly, should support, additionally to the requirements of Level 2, FEEL. This is introduced as a boxed expression and can have a complex structure. In addition to the S-FEEL functions, FEEL supports complex textual and arithmetic expressions, loops, conditions, and more. [34]

Hit policy

DMN decision tables, different from other rule-based systems, can have different rules for hit policy. There are two groups, depending on the output kind. For a single result, there are: Unique hit (U - only one rule can match), First-hit(F - first matching rule is taken), Priority hit (P - rule with the highest priority for the output match), Any hit (A -

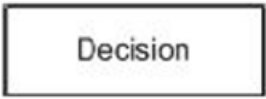
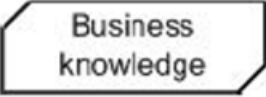
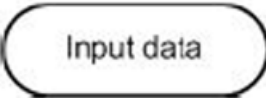
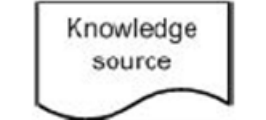
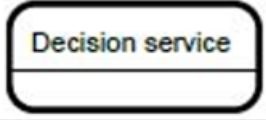
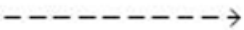
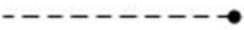
Elements	Decision	
	Business Knowledge Model	
	Input Data	
	Knowledge Source	
	Decision Service (expanded)	
	Decision Service (collapsed)	
Requirements	Information requirement	
	Knowledge requirement	
	Authority Requirement	
Artifacts	Text Annotation	
	Association	
	Group	

Figure 2.2: DRD components [34]

2 *Foundation*

all matching rules have the same output), Collect and aggregate (C+ Sum, C< Minimum, C> Maximum, C Number - the result is calculated or chosen according to the hit policy). For multiple results: Collect (C - all rules collected), Rule order (R - all rules in the order from the decision table) and Output order (O - all rules by decreasing output priority). [14]

3 Related works

This chapter includes state-of-the-art on DMN research, related works about conducting a survey and surveys on modeling languages and tools, current works on DMN verification, validation and simplification.

The first results in DMN research go in a few different fields, such as using DMN and BPMN, modeling, verification and validation. The researchers from the polish AGH University of Science and Technology in [27] highlighted the research directions on DMN in the literature from 2014 to 2018 (since the first Beta version of DMN was released in 2014). There, seven research fields were found and classified[27]:

1. Verification and validation of models
Formal verification and validation of models and requirements, as well as understandability and consistency of the models.
2. Formalization
Formalization enhances the quality of semantics, analyses the models, and supports the integration and execution processes.
3. Adjustment
This field includes the adjustments for model extensions, model synthesis with BPMN, and integration into the other modeling methods.
4. Acquisition
Manual acquisition, translation from other models or standards, automatic decision generation.
5. Modeling
DMN usage in different domains, its suitability, comparison of the advantages against the other modeling methods and different kinds of analysis of DMN.
6. Tool support
Tool support for DMN modeling, verification, serialization of the models, maintainability, scalability and portability.
7. Enactment
Simulation, execution and optimization of the DMN models.

All of these points are interesting for improving DMN and its usability.

Speaking about acquisition, there is an article written by Bazhenova, Zerbato, Oliboni and Weske [4], which proposes an acquisition aspect - derivation of DMN from the BPMN

3 Related works

processes data. The authors suggest a pattern-based approach for processing parts of BPMN models by extracting the data elements. They are classifying all BPMN elements for possible data in the content. Next, they identify each element's decision patterns and map them into a decision model. There is a possibility for BPMN simplification by extracting the data-based decision into a built-in DMN. This process can also be automatized after the pattern base is expanded. This supports the concept of DMN usage within BPMN with its advantages. [4]

To further enhance DMN usefulness, there is some research in automated decision-making. So in work by Etinger, Simić and Buljubašić [17], a method for automated creation of DMN tables from decision trees is introduced. They are suggesting intelligent support for DMN modeling. This innovation can be used for BPMN as well. With this, two steps of DMN modeling are supported: formulation of the decision logic and creation of decision services. The proposed algorithm converts decision trees to a machine learning model using a Python ML-library - scikit-learn. This ML model creates a DMN decision table in an XML-format in the next step. This research supports the automatization of the DMN-modeling process. Though it has a limitation for the cases with too many attributes. [17]

Another important work that influenced this thesis is [26]. The paper is about Systematic Reviews and the review process, including the planning step, conducting the review and reporting of the results. The authors describe all processes, including the documentation of the research process.

3.1 Surveys

As current research includes a survey on the status of the DMN tools, some works focused on tool surveys were chosen.

Currently, few works on surveys for modeling languages, but even fewer are somehow aimed at DMN. Bork, Karagiannis and Pittl in [10] performed a systematic analysis based on the concepts of modeling methods and evaluation of modeling standards. The authors used a three-phase research method from [26]. In their different work, [9], the authors analyzed diverse specifications, inter alia, the DMN specification. They showed different specification techniques that have been used in the analyzed specifications based on the three-phase research method mentioned above.

Another recent close work is [15], which is directed at the validation functionality of DMN tools. [21] also includes a comparison of verification functionalities in different researches. [32] provides a survey on BPMN tools. The authors used the Systematic Literature Review method for conducting the survey. The tools are being reviewed according to the different criteria. The method can be applied to other modeling languages. [19] surveyed modeling and simulation software frameworks. The authors also decided on some criteria and conducted a performance analysis of the frameworks.

Eichelberger, Eldogan and Schmid in [16] concentrated the work of the survey on the UML-tools, precisely on verification of tools by the compliance to the OMG standard. This is relevant for the future survey criteria described in Chapter 4. [11] includes research and

a survey on modeling languages, their metamodels and specifications. Bork, Karagiannis and Pittl conducted a Systematic Literature Review to analyze the state-of-the-art and, in conclusion, compare different modeling languages by their specification techniques for particular modeling concepts.

3.2 Verification & Validation, DMN Simplification

Figl, Mendling, Tokdemir and Vanthienen explain in [18] the DMN background as well as DMN benefits. The last is discussed on a technological level, organizational and individual level. Their different research directions are highlighted. Among them is research on validation and verification of DMN models, DMN tables and Rule-Based System (RBS), and simplification of the tables. The central anomalies for the rules in the rule-based systems are highlighted in the paper. The authors covered several current kinds of research on the validation and verification of DMN and RBS. Some papers and discovered algorithms are mentioned below and used further in this paper. In [18], current research status on other fields is exhibited, such as research status on code generation, decision mining, as well as the semantics of DMN and its visual presentation.

Since the beginning of this research, a paper by Corea and Delfmann [20] was introduced. The authors analyzed the current tools for the verification functionality. The authors combined all possible verification criteria based on the papers highlighted in this research and created a complex list of verification/validation criteria. They discovered the low level of this functionality support within the industrial software. Thus, only 5 of 14 tested tools support at least one verification functionality, whereas already 11 scientific papers provided some ideas and tools with at least one DMN verification capability. Previous to this research, the authors published a tool implementation that includes some mentioned verification criteria on the decision logic level. [20] [15]

Some authors propose algorithms to verify and simplify DMN tables and support them with implemented solutions and tool extensions.

As part of the research, [21], a tool for verification of DMN models was created. The authors meant the current research was mostly on the logic level (decision tables), where seven other research works were reviewed. The authors submitted a tool based on the open-source Camunda Modeler. The tool covered verification on the logic level, as well as on the DRD level. Main verification criteria were based on the rule verification (identical, equivalent, subsumed rules, overlapping rules, missing rules, rules reduction), input/output values and columns, and data types. [21]

Another research was submitted in [28], with a corresponding tool implementation [29]. It includes verifying decision tables by syntactic check, detecting overlapping and missing rules, and simplifying by rule merging.

Outstanding work on DMN simplification was done in [12]. The authors worked on a solution for semantics, analysis and simplification of DMN decision tables. The main focus was the geometric representation of the rules and the concluding reorganization and reduction of the rules. The suggested method detects overlapping and missing rules and rule-merging possibilities. The method is briefly explained in Chapter 6.

3 Related works

Their last two approaches are further explained in Chapter 6.

Hasić and Vanthienen provided a set of complexity metrics in their work [22]. They highlighted several essential metrics that can be used to evaluate DMN complexity. The metrics are based on different sources from the business process and software engineering.

These metrics can be used primarily for the evaluation of the results of the case study. Thus, it will be used in the current research.

As will be further explained in the current work, DMN is a kind of rule-based system. This fact can be used for further research on associated topics, such as the effectiveness of DMN-rules and rule-based system simplification. Thus, the paper by Ben-David [5] is focused on rule-effectiveness in RBS. The author proposes a method for rule reduction besides the classical logic-based rule reduction. The technique can be used alongside logical and graph-based reduction. The aim is to remove rarely-used rules from the RBS. The method uses machine learning validation, where different models are built and classified for an error rate using the Nearest Neighbor classification algorithm. By using the proposed algorithm, the most informative rules and less informative ones are identified. This method can support the modeler in decision-making for optimizing the rule-base and providing a more compact RBS.

So, the authors of Automated first order natural deduction [7] introduced a complete automated proof-searching algorithm for the first order natural deduction where the natural deduction is a kind of reasoning proof in logic.

3.3 Summary

Following the research above, one can see that there are some state-of-the-art papers on DMN research, as well as verification and validation. Some works are on automated decision-making and DMN generation, e.g., from BPMN data. Various surveys were found on DMN and other modeling languages and tools. Also, some papers on the art of conducting surveys and Systematic Literature Review, where SLR method was used in a few surveys.

On improvement of DMN models, there are some works on verification and validation of decision tables and some poor research on DRD verification and simplification at the DRD level. These results are visually summarized in 3.1. There is a lack of research on the DRD level, and more can be done on the simplification on the decision table level.

Having the results of the literature research, the current work will deepen the research in less covered directions, such as DMN simplification on the decision table and DRD level.

Paper	Decision table verifica- tion	DRD verifica- tion	Simplification	Simplification on Decision table level	Simplification on DRD level	Tool
[20]/[15]	yes	no	no	no	no	yes
[21]	yes	yes	no	no	no	yes
[28]	yes	no	yes	yes	no	yes
[12]	yes	no	yes	yes	no	yes
[5]	no	no	yes	yes	no	no

Table 3.1: Related works summary

4 Research methods

This Chapter describes the research process and methods, including the literature review process description, tool research process, and evaluation criteria analysis.

For this thesis, a Design Science Research methodology was used. The following sections describe (i) the Design Science Research and how it is applied for current research, (ii) the research process for the exploration phase of Design Science Research (DSR), particularly the literature review, systematic tool review, and its evaluation for the results. The next Chapter, chapter 5, is part of the exploration phase, including planning the tool survey and conducting the Survey itself. The second phase of DSR is introduced in Chapters 6 and 7, and the last one respectively in the last two Chapters: Chapter 8 and 9.

4.1 Design Science Research

Design Science Research is a way of problem-solving based on the existing knowledge on the topic in design disciplines. It provides intelligent solutions, including tools and methods. [23]

Horváth defines three phases of DSR: [23]

1. Exploration, finding the problem
2. Design and testing of the solution
3. Verification, validation, generalization

DSR includes 6 steps, as shown in Figure 4.1, from [35]. There are problem identification and motivation, the definition of the objectives for a solution, design and development, demonstration, evaluation, and communication.

Alturki in [3], explains the "Information System Design Theory Lifecycle" (ISDTL). Since DSR cannot always deliver an optimum solution, the technology does not stay the same, and the problems and opportunities change over time. This means that the solution found with the DSR can be reinforced in some time, having new questions and possibilities. [3] This concept fits the IS area the best with its rapidly evolving. Looking ahead in this paper, this could be seen in the research process. Since the process lasted two years, some tools from the Survey were upgraded with new functions, and some new papers on the topic appeared.

When speaking about IT-area, the ISDTL can be seen as the child of the DSR methodology and Iterative development. The main idea of it is to iterate through the process of Design Science Research, having updated problem definitions and opportunities with each iteration. This is nicely shown in Figure 4.2 from [3].

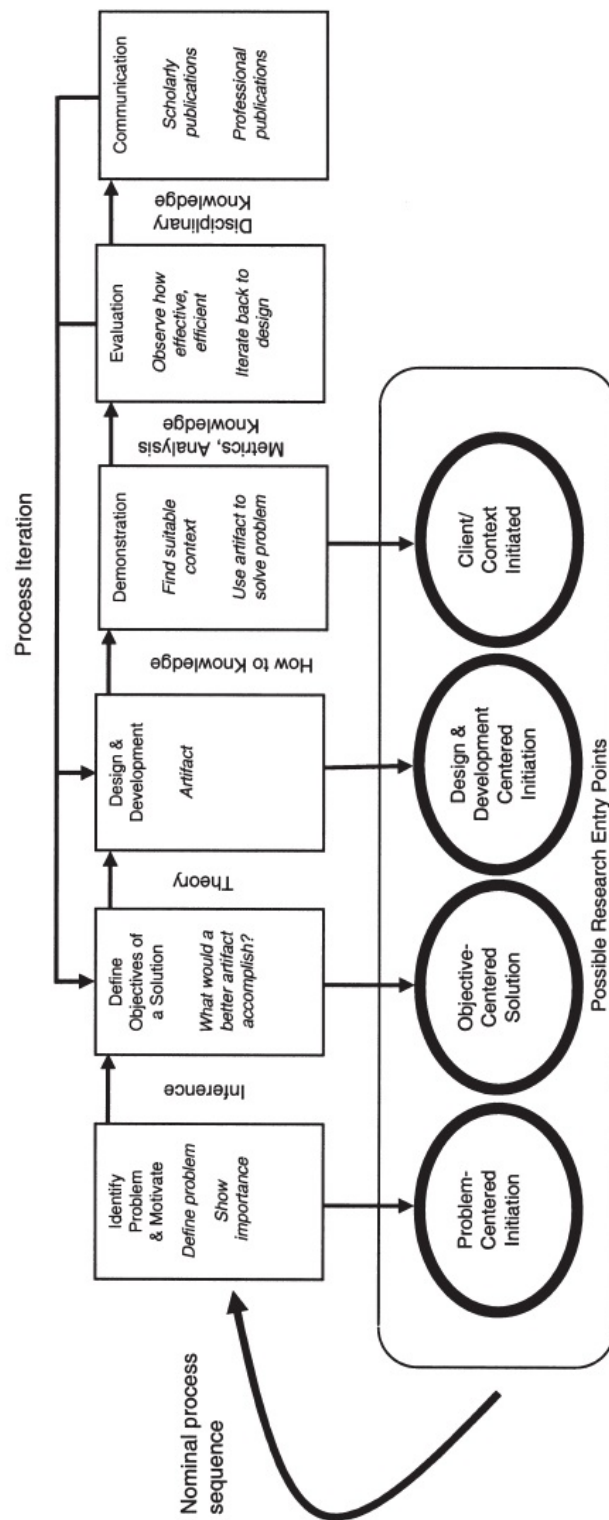


Figure 4.1: DSR lifecycle [35]

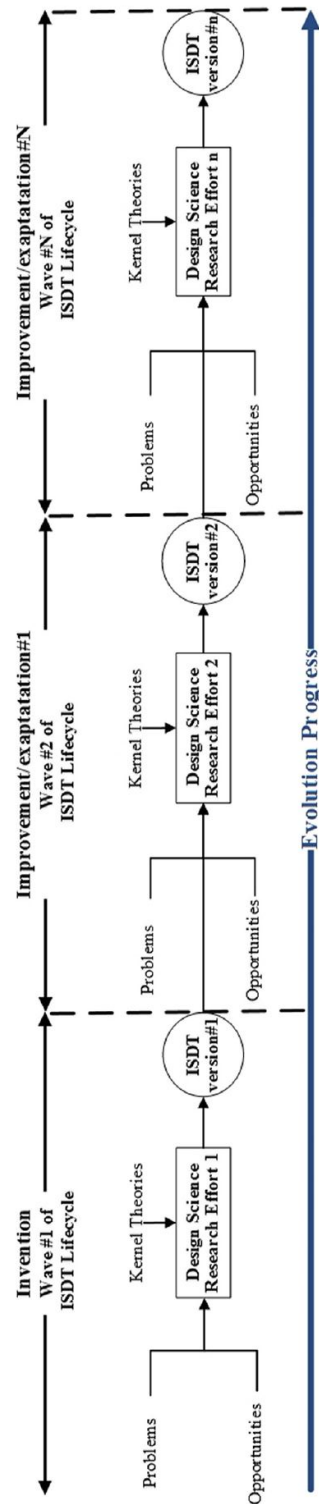


Figure 4.2: ISDT lifecycle [3]

4 Research methods

Having this described, the current work already had two iterations in the research process in the background, which even changed the research questions until the research found the exact direction. The presented result can be seen as one iteration since it eases the reading and nicely fits in the paper structure. This can be used for successive waves as version 1 within the Design Theory Lifecycle.

Applying DSR to this research, the following tasks were assigned to each step:

Identify problem & motivate	Related works
Define objectives of a solution	In this step, a literature review was conducted, and a survey on existing tools was conducted. The main emphases were mechanisms and algorithms for DMN modeling methods, particularly simplification, verification, and validation of DMN.
Design & development	Implementation
Demonstration	Case study
Evaluation	Case study
Communication	Thesis defense, publication in the university library

The exploration of the area, or process of problem identification and definition of objectives from the 6-step-process, included steps from the research status to the status of the current software/tools. The following steps were taken within the exploration stage of DSR:

- Research status
 - Current research status
 - Search for papers on DMN
- Prepare survey
 - Conduct literature review on surveying process
 - Derive review criteria
 - Search for DMN tools
 - Request research access
 - Choose tools
- Perform survey
 - Install tools
 - Test all functions for the review criteria
- Analyze the research and survey results

4.2 Literature review

For the literature review, a search platform and a query had to be defined at the initial stage. This needs a clear definition of the planned results. The Survey has to be well planned. With this in mind, the literature review intended to identify the existing tool surveys for further analysis. In order to find relevant literature, the following query was created:

(Survey OR evaluation OR review OR overview OR analysis) AND (modeling tool OR tool OR instrument OR suit OR service).

Then with help of Publish or Perish, Google Scholar and small Systematic Literature Review (SLR)[26] [32] [11], 8 publications were chosen for the further review. This was done following the SLR idea of step-by-step illumination.

During the first step, several times on different days, 25 results were selected based on the domain that could be read out of the titles. The focus was on computer modeling since many papers were related to machine modeling. Criteria for exclusion were: single tool reviews and papers that do not review software tools. Papers that were included were focused on modeling methods, having phrases such as "as a tool", "tool for the evaluation", "analysis on ... modeling", "mathematical modeling" and similar.

In the second stage, the abstracts were read, and 17 relevant publications were selected for further analysis. The next step consisted of examining the paper to see if the review describes its criteria and how the results are presented. Here, eight publications were screened for deep reading. The described selection process can be seen in Figure 4.3.

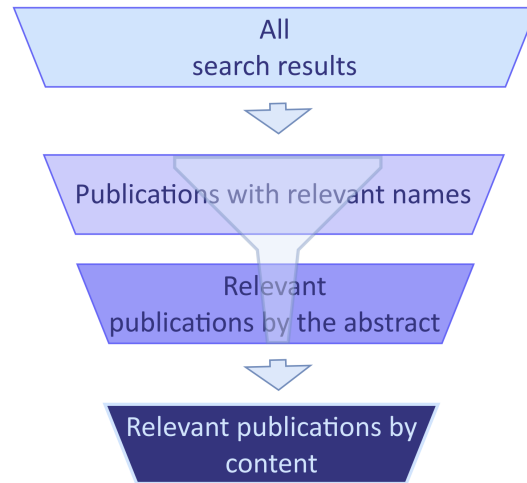


Figure 4.3: Literature filtering process

The chosen papers formed the basis for defining criteria for the tool survey. The

method is explained in Research methods and the results are discussed in Conclusion. The following papers were selected in the process: [32], [1], [37], [38], [16], [19], [31], [41].

Finding the other surveys helps to find the most used criteria and build a complex criteria catalog for the Survey based on the existing knowledge.

4.3 Systematic tool review

Within the current thesis, a tool survey was conducted. The survey process was based on the Systematic Tool Review. The idea of Systematic Tool Review comes from Systematic Literature Review (SLR). SLR includes identifying, evaluating, and interpreting all existing relevant sources. To summarize existing knowledge on the topic, identify gaps and provide the basis for the new research, SLR was performed. SLR has to include a predefined search plan of action, having the inclusion and exclusion criteria defined and the information awaited as a result. SLR includes three steps: planning, conducting, and reporting the review. [26][32]

A Systematic Tool Review (STR) can be performed based on SLR. Tools will play the role of the research elements instead of research papers. The SLR procedure, as described in [26], can be applied as follows:

1. Planning the review

To plan the review, first, the *necessity of the review* has to be established.

Second, a review protocol has to be developed. The protocol should include a description of the review process, including the search of the tools and the further review process of each tool. It contains the *search strategy for the tools*; *tool selection criteria* and *tool review strategy*. In the tool review strategy, a *criteria catalog* for the review has to be developed.

This step is presented in the following subsection.

2. Conducting the review

This step includes the selection of tools and tool review process based on the defined criteria. The result of this process is described in Chapter 5.

3. Reporting the review

The review is reported in the current Chapter of the thesis and in Chapter Survey of DMN tools.

Planning the review

As discussed in Chapter 3, the gaps in the DMN research have to be defined. The status of the current tool on the market has to be reviewed. This brings us to the research questions for the following Systematic Tool Review:

- RQ 2.1: What is the market's current status for DMN tools?
- RQ 2.2: Which functionalities are well represented in the current tools?

- RQ 2.3: Which functionalities are less represented or absent in the current tools?

These research questions support the general research questions of the current thesis.

Search strategy for the tools

This step includes the selection criteria for the tools.

Tool search has to be started with the search of all available tools on the internet. For this, search words such as "DMN tools", "DMN software", "DMN framework", "DMN modeling tools" can be applied. This would create a list of all available tools. In the second step, the **inclusion criteria** have to be applied:

- Tool can be found on the company page
- Tool supports DMN modeling
- Company web page has some information about the tool

After the first complete list of the tools was created, all tools had to be checked for accessibility: open-source, trial version available, or proprietary software without trial version. For the last category, a request to access the tools for the research have to be sent to each company. After this, the next **inclusion criterion** can be applied:

- Tool can be accessed for the research

These criteria imply the **exclusion criteria**:

- Tool cannot be found
- Tool does not support DMN modeling
- Tool cannot be accessed for the research - custom solution or no access given
- Plugins, which are not tools

The aim is to find the appropriate and accessible tools for the review. So, the tools can be excluded step-by-step to review just the target group of tools. Thus, the selection workflow will be the following:

- search for the tools;
- retrieve tool information;
- request access if necessary;
- apply inclusion criteria;
- apply exclusion criteria.

Evaluation criteria

For the review strategy, a criteria catalog has to be created. The eight selected sources from Literature review had to be analyzed as follows:

- revealing the review criteria;
- filling into the shared criteria table;
While filling the criteria into the table, taking care of the different names of similar criteria is crucial - the aim is to standardize the criteria as much as possible.
- counting the criteria and taking a deeper look at the principles, repeated in more than one Survey.

Survey criteria have to be defined based on the results and the research questions. Since the main point of interest is additional functionality, such as execution, analysis, verification & validation and DMN simplification, these criteria should be included in the catalog. The criteria must be clearly defined, including the possible values for the evaluation.

Review

After finalizing the criteria, all selected tools must be reviewed. The review process should be documented, and the criteria should be filled with values during the review process of the tool. This should start with the tool's installation, followed by modeling with the tool, using the same test example for each. The other functionalities from the criteria catalog must be tested and filled in in the review.

Next Chapter, Chapter 5, shows the results of the tool survey, that was executed following the described strategy.

5 Survey of DMN tools

In this chapter, the process and the results of the survey on DMN-tools are shown. The chapter also includes a general overview of the existing DMN-tools, the Systematic Tool Review (STR) results, evaluation criteria, and reviews of the chosen tools, following the described criteria. The chapter will be closed with summarized results and a discussion of the ideas that can be added to the tools or upgraded to increase the added value of DMN-usage.

5.1 DMN tools overview

DMN software is not as standard as BPMN software since DMN is generally less widely used than BPMN. This trend can be seen particularly in the search results. A search with different queries was conducted on "Google search" and "Google Scholar", and the following search results came up: (Status for 9th of January 2022)

Google search results:

"business process model and notation" - 176.000 results

"decision model and notation" - 48.300 results

"business process model and notation" AND "tool" OR "software" - 133.000 results

"decision model and notation" AND "tool" OR "software" - 42.100 results

Google Scholar search results:

"business process model and notation" - 15.000 results

"decision model and notation" - 787 results

"business process model and notation" AND "tool" OR "software" - 14.600 results

"decision model and notation" AND "tool" OR "software" - 743 results

It can be clearly seen that DMN is less popular than BPMN, so fewer tools can work with DMN. Nevertheless, few tools were found in the deeper research. However, most companies are creating specific and personalized solutions for the companies that are based on DMN. Thus, their solutions are mostly offered as company services and not presented as complete software to download, or no test or demo version is offered. In the next section, section 5.1, more detailed research results for the DMN tool market are described.

Research

The research on existing tools took place on Google, with search words, like "DMN tools", "DMN software", "DMN framework", "DMN modeling tools". Also, the lists of DMN creators was used from OMG DMN.

The review of the tools was conducted following the STR (Systematic Tool Review), which was described in Research methods. **First**, a list of all discovered tools was created. **Second**, every company/tool was checked for existence. Since the lists can include information that is not up-to-date. Then, the websites of the found companies were searched for tool information. Here three kinds of tools were spotted: open-source tools, proprietary software, and custom software. Proprietary software has different subscription types, such as test or demo versions and academic access. **Third**, for the proprietary software, a trial or academic access was requested. In the scope of this research, only accessible software was taken into consideration. That means all open source solutions, demo/test versions, tools with academic licensing and software with given research access. Described tools were included in the **fourth** step. In the **last** step, all individual solutions were disqualified, alongside software without given access.

This research gave the following results:

In the first round, 29 tools were found. For three tools, no website or no further tool information was found. These were Blueriq, Fujitsu and Oracle. Of the remaining tools, 4 were downloadable open source tools, and the other 24 were proprietary (SaaS or provided custom solutions) or not usable. All the proprietary software was requested, and as a result only three companies provided access to their tools. The others either gave no access or did not respond to the request. These are mostly the companies selling their automated solutions. They were not interested in free access for evaluation, did not reply to the request, or did not have such a possibility. The other did not fit since the function scope was not covered (e.g., jDMN included only execution, IDIOM included only rule management). For one of the tools - Signavio, the Computer Science faculty of the University of Vienna had granted academic access. Since Kogito is powered by Drools, it was counted and reviewed as one. Thus, there are the following statistics:

- Found tools total: 28 tools (not counting Drools)
- Open-source tools: 4 (Camunda, Bee-up, BPMN.io, Kogito/Drools)
- Academic license: 1 (Signavio)
- Trial version: 3 (Trisotech, Pencil, Cardanit)

The results of the research, including all discovered tools, are presented in Table 5.1.

Company	Website	Access type	Included in survey
Actico	https://www.actico.com/	Proprietary	no
ADONIS	https://www.adonis-community.com/en/	Proprietary	no
AlfrescoActiviti	https://hub.alfresco.com/	Proprietary	no
Avola	https://avola-decision.com/	Proprietary	no
Bee-UP	https://austria.omilab.org/	Open-source	yes
BizzDesign	https://service.bizzdesign.com/	Proprietary	no
Blueriq	https://www.blueriq.com/en/solutions/	Proprietary	no
BPMN.io (DMN)	http://BPMN.io/	Open-source	yes
Camunda	http://camunda.org/DMN/tool	Open-source	yes
Cardanit	https://www.cardanit.com/	Trial version available	yes
Decision Management Solutions (DecisionsFirst)	https://www.decisionmanagementsolutions.com/	Proprietary	no
Decisions	https://decisions.com/	Proprietary	no
DecisionsFirst Modeler	http://www.decisionmanagementsolutions.com/	Proprietary	no
Drools (Red Hat)	https://www.drools.org/learn/DMN.html	Free trial	no
FICO	http://www.fico.com/		no
FlexRule	https://www.flexrule.com/archives/decision-model-and-notation-DMN/	Proprietary	no
Fujitsu	https://www.fujitsu.com/	Proprietary	no
IDIOM	http://www.idiomssoftware.com/	Only execution available	no
jDMN	https://github.com/goldmansachs/jDMN	Only execution available	no

Kogito (using Drools)	https://kiegroup.github.io/		Open-source	yes
OneDecision	https://github.com/OneDecision/one-decision		Project on github, unavailable website	no
Open rules	http://openrules.com/ (aws marketplace)		Proprietary	no
Oracle (Process Cloud Service)	Tool not found			no
RapidGen	https://rapidgen.com/		Proprietary	no
RubiconRed (oracle)	https://www.rubiconred.com/		Proprietary	no
Sapiens	https://www.sapiens.com/		Proprietary	no
Signavio	http://www.signavio.com/		Academic access	yes
Sparkling Logic (Pencil)	https://www.sparklinglogic.com/		Trial version available (only modeler)	yes
Trisotech	http://www.trisotech.com/		Trial version available	yes

Table 5.1: DMN tools

5.2 Evaluation criteria

In this section, criteria for the survey are presented. It includes four main categories: Decision Modelling/Functional Criteria, Technical criteria, Availability, and Feasibility and usability. The criteria were selected from the survey research, presented in Section 4. All characteristics are described below and possible values for the evaluation are defined.

Table 5.2 shows the overview of the criteria catalog. It contains the chosen criteria, grouped by their fields. Further description of every point is placed below.

Decision Modelling/Functional Criteria	Functionality provided Visual appearance Attribute functionality DMN version Correctness checker (validation) Interoperability with other models Model execution Analysis Simplification
Technical criteria	Import and export formats Platform requirements Hardware constraints Programming language requirements
Availability	Availability License Tool notability
Feasibility and usability	GUI Usability Effort required Documentation

Table 5.2: Criteria catalog

• Decision Modelling/Functional Criteria

This criterion's main focus is on the tool's global characteristics, which are important for the modeling aspect. These are functionality, visual look, Attribute functionality, version of DMN specification supported, correctness validation, interoperability, and the existence of the model execution.

Functionality provided

Values: Level 1, level 2, level 3.

This point describes the conformance to DMN specification. The levels correspond to the conformance levels of the specification. This describes the compliance of the tool with DMN specification. Here it is important to look up the tool providers' specifications. [34]

5 Survey of DMN tools

- Level 1 includes support of decision requirement diagrams (DRDs), decision logic, and decision tables.
- Level 2 includes, additionally to the requirements in conformance level 1, support of Simplified Friendly Enough Expression Language (S-FEEL) expressions and fully executable decision models.
- Level 3 includes, additionally to the requirements in levels 1 and 2, support of Friendly Enough Expression Language (FEEL) expressions within boxed expressions.

Analysis functionality

Values: provided (what kind of), not provided. This criterion outlines the possibility of analyzing the DMN model according to the complexity, usage of elements, correctness, or others.

Simplification functionality

Values: provided (what kind of), not provided. This criterion shows if the tool has an additional function that helps to reduce the complexity of the DMN in any way.

Visual appearance

Values: limited, minimal, extended.

This element describes the use of graphs, shapes and markers, as defined in the DMN specification. The values correspond to "does not exist, meets the minimal requirements, includes complex elements". The main elements can be found in Figure 2.2.

- Limited: not all elements included or no graphical presentation of DMN elements at all
- Minimal: meets the minimum specification requirements.
For DMN version 1.1: Decision, Business knowledge, Input data, Knowledge source, Information requirement, Knowledge requirement, Authority requirement, Text annotation, Association.
For DMN version 1.2 additionally - Decision service.
For DMN version 1.3 additionally to version 1.1 and 1.2 - Group.
- Extended: an extended set of shapes used or extended possibilities for presentation of single elements (e.g., changing color).

Attribute functionality

Values: complete, not complete, none.

Here the compliance to the attribute specification is depicted:

- Complete: fully provided functionality, fulfills all recommendations from the specification. The required attributes are marked in the XML-schema of the corresponding DMN specification as "required".
For DMN version 1.1 - "href"-attribute for the reference element [tDMNElementReference], "namespace"-attribute for the named elements [tNamedElement]

in definitions [tDefinitions] and import data type [tImport] and “importType”-attribute for the named elements in import data type.

For DMN version 1.2 - “name”-attribute for all elements [tDMNElement] additionally to the attributes from 1.1.

For DMN version 1.3 - a "DMNElementRef"-attribute for all shapes [DMN-Shape] and Edges [DMNEdge], that is needed to connect components for the reproduction of the diagram.

- Not complete: provided functionality, but no graphical representation provided
- None: no attribute functionality provided, adding attributes is not possible

DMN version

Values: 1.3 (current), 1.2, 1.1, 1.0.

Describes which DMN version is supported by the tool. Currently, the last version is 1.3 (1.4 Beta version is already available).

Correctness checker/validation

Values: provided, not provided.

Shows if the tool can analyze models for accuracy.

Interoperability with other models

Values: provided, only import, only export, not provided.

This criterion depicts the ability of the DMN Diagram Interchange. This can be determined by controlling the validity of the Output-XML following the OMG Diagram Definition Specification [34]. This can be controlled through the XML validation using the XSD-schema provided by OMG, or for import, by importing an example model provided by OMG.

Model execution

Values: provided, not provided.

Possibility to evaluate decisions using a decision service, e.g., using a provided interface.

• Technical criteria

In this category, the technical criteria for the software execution are shown. This can become an inclusion/exclusion criterion for the tools if, for example, it cannot be run by the OS used for the reviews.

Import and export formats

Values: import formats, export formats. At this point, all the possible formats for import into the tool and export of models should be listed.

Platform requirements

Values: macOS, Windows, Linux, Web.

Here the platform/platforms required for the software is shown.

Hardware constraints

Values: none, <constraints>.

This element shows the hardware constraints for the tool execution, if any.

Programming language requirements

Values: none, <language>.

This characteristic depicts the possible programming languages that are supported by the tool.

- **Availability**

This element describes the availability criteria of the tool: whether it is easy and open for access and its maturity.

Availability

Values: high, middle, low.

Describes the access to download: high - can be downloaded without registration; middle - registration needed; low - no link provided on the website, has to be requested.

License

Values: Open source, academic access, proprietary.

Here, the price category for the tool is depicted: if the tool is free or proprietary.

Tool notability

Values: high, middle, low.

This criterion is derived from the website traffic and search engine results.

- **Feasibility and usability**

Here the ease of the tool execution and usage is depicted.

GUI

Values: Minimal, extended, advanced.

Describes the level of the graphical user interface:

- "Minimal" - covers the minimal requirements of DMN specifications, allows drawing the basic notation;
- "Extended" - has at least one non-obligatory functionality (like sizing or colors);
- "Advanced" - the tool provides a complex GUI, that corresponds to the highest software standards.

Usability

Values: high, medium, low.

Usability shows if the usage of the tool is easy and self-explanatory. This has to be evaluated by organizing some tests. Several test persons should create the same DMN diagram in all test tools. The time spent will be recorded, evaluated and compared with other tools. For this thesis, only the author conducted the review. For less biased results, more test persons are needed.

Effort required

Values: high, medium, low (complicated setup, medium effort, little effort (automatic

setup)).

This criterion shows the ease of installation and the effort needed to start the work with the tool and create the first diagram after downloading.

Documentation

Values: good, poor, none.

This criterion depicts the existence of the documentation for the tool and its richness.

5.3 Selected tools

As explained above, eight tools were chosen for the review from all discovered tools. These are all open-source tools, trial versions of proprietary software, and tools with academic and research access.

The following tools were chosen for the review:

- Bee-Up;
- BPMN.io;
- Camunda;
- Cardanit;
- Kogito;
- Pencil;
- Signavio;
- Trisotech;

Selected tools were reviewed for the available versions, downloaded (if necessary) and prepared for the test. Here is a general overview and the first impressions:

Bee-Up

Bee-Up is open-source software, implemented by OmiLab[8], based at the University of Vienna. The installation process is not complicated: download the zip file and run the setup.exe. Bee-Up supports several modeling languages, as BPMN, DMN, ER, Petri Nets, UML diagrams.

BPMN.io

BPMN.io is open-source software powered by Camunda. It is built to be used as a built-in modeler on websites in a browser. It can be used as it is - by just introducing a library into the web page, or it can be extended to meet developers' needs. A user has libraries for BPMN, DMN and CMMN. The software is easy to install, has a web version for a test and a downloadable version.

Camunda

is a Java-based framework that supports BPMN, DMN and CMMN. Camunda deeply works on process automation and has an expert blog, many resources, as well as white papers on different aspects of it. It has three different versions: an Open Source Modeler, the Community Edition and an Enterprise Edition with a 30-days trial. We are testing the most available open source and trial versions, so for this tool it is its' open source modeler.[13]

Cardanit

Cardanit supports BPMN and DMN modeling, including using DMN within BPMN. The company provides a 30 days trial. The trial version is restricted and includes only the modeling functions, 5 BPMN/DMN projects, no collaboration, customization, version history and documentation. The tool is Web-based; no installation is needed.

Kogito

Kogito is a company that works on different cloud-based automation solutions for companies. Kogito builds Java as well as JavaScript applications. The company provides a Drools-based DMN modeler. It includes an online editor, a GitHub extension and a desktop preview. We will test the online editor since the desktop version could not be installed. The developers mention in their release information that Kogito is a young and developing software. Thus it may include not fully tested features.

Pencil

Sparkling logic provided to us only their modeling software. The execution software represents complex custom solutions and does not have any universal version that can be provided for a trial, as company representatives explained in the conversation. It is essential to note the excellent customer service and fast responses regarding the access to test the tool. Pencil is a web-based modeler with good documentation, including tutorials on how to start and best practices for DMN usage.

Signavio

Signavio Process manager is constantly developing. So, there were seven updated versions in the time from August 2021 to December 2021, which is more than one update in a month. The last analyzed version was 15.10.1.

Since the beginning of 2021, Signavio became a part of SAP in the business process intelligence area. The product that covers process modeling and DMN, is called SAP Signavio Process Manager. Software is cloud-based and accessible in web. Signavio is proprietary software, with a free 30-days-trial, as well as an academic license. The academic version was analyzed for this survey.

Trisotech

Trisotech provides a so-called "Digital Enterprise Suite", that includes a "Digital Modeling Suite" and a "Digital Automation Suite". The company presents different business cases with its solutions. Trisotech provides proprietary software and a 7-day trial on their website. After the request for the current research was sent, the company provided more extended access for a few months. The software is web-based, no installation is needed.

5.4 Tool review process

In this step, the tools were tested with the same DMN example. A model from [39] was taken as a test example. As seen on the diagram 5.1, it has the most important elements and functions included. In Figure 5.2, the **Calculate customer discount** table is shown. As for the other decisions, they include some calculations:

Product value: sale items

$\text{sum}(\text{Product value: Sale items} * \text{Sale discount} * \text{Calculate customer discount})$

Product value: Ordinary products

$\text{sum}(\text{Product value: Ordinary items} * \text{Calculate customer discount})$

The main functionalities were tested for the survey criteria described above. The overall results for the functional criteria are presented together with some availability criteria in the Table 5.3 and 5.4, for the technical criteria, in the Table 5.5, and for feasibility and usability criteria - in the Table 5.6. Some testing highlights for every tool are represented below.

5 Survey of DMN tools

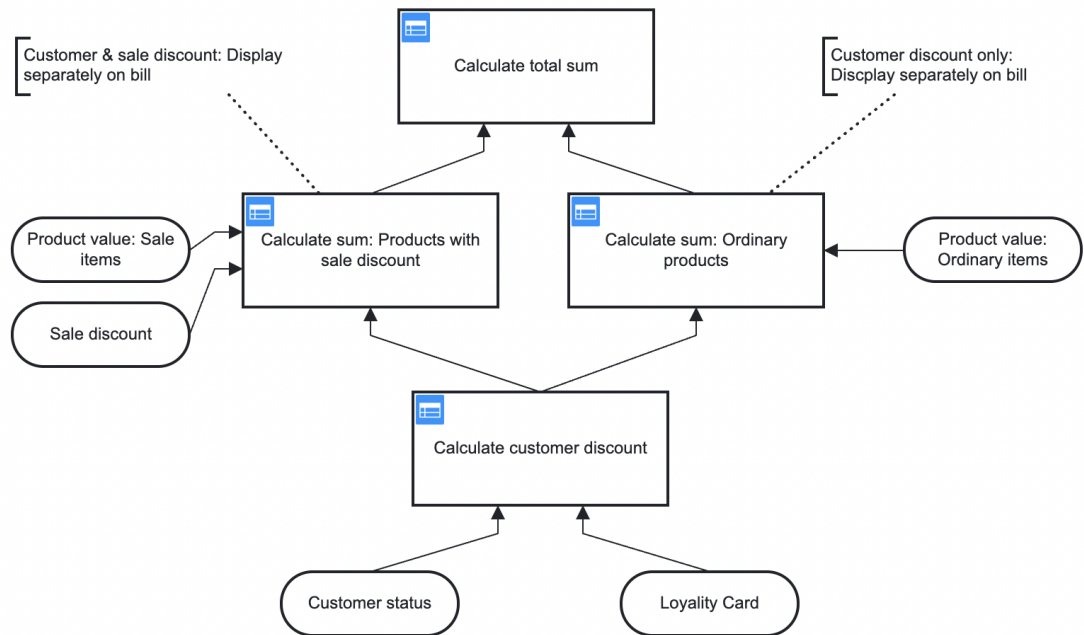


Figure 5.1: Test DMN model - DRD [39]

View DRD

Calculate customer discount					Hit Policy: Unique
	When	And	Then		Annotations
	Customer status	Loyalty Card	Discount		
	"Standard","Premium","Gold"	boolean	number		
1	"Standard"	false	0.00%		
2	"Standard"	true	10.00%		
3	"Premium"	false	15.00%		
4	"Premium"	true	20.00%		
5	"Gold"	-	30.00%		
+	-	-			

Figure 5.2: Test DMN model - decision table [39]

Name	Tool version	DMN version	Availability	License	Functionality provided
Values		1.3 (current), 1.2, 1.1, 1.0	high, middle, low	open source, academic access, proprietary	L1, L2, L3
Bee-Up	1.5	1.1	high	open source	L2
BPMN.io (DMN)	11.0.2	1.3	high	open source	L3
Camunda	4.11.1	1.3	high	open source	L3
Cardanit		1.3	middle: login required	30 days trial or 5 projects for free	L1
Kogito	Sandbox	1.3	high	open source	L3
Signavio	15.10.1	1.2	low: request demo version	30 days trial or academic	L3
Sparkling Logic (Pencil)	product demo online	1.3	low: request needed	free trial	L2/3
Trisotech	10.15.3	1.3	low: login and request required	7-30-days-trial	L3

Table 5.3: Functional criteria (1/2)

Name	Visual appearance	Attribute functionality	Correctness checker	Inter-operability	Model execution	Analysis	Simplification
Values	limited, minimal, extended	complete, not complete, none	provided, not provided	provided, not provided	provided, not provided		
Bee-Up	extended	not complete	provided	not provided	not provided	-	-
BPMN.io (DMN)	extended	not complete	provided	provided	not provided	-	-
Camunda	minimal	complete	provided	provided (limited)	provided	-	-
Cardanit	extended	complete	provided	provided	not provided	-	-
Kogito	extended	complete	provided	provided	provided	Through the execution	-
Signavio	extended	complete	provided	provided (only export)	provided	Rule analysis	-
Sparkling Logic (Pencil)	extended	complete	provided	provided	provided, but not obvious	Model verification	-
Trisotech	extended	complete	provided	provided	provided	Model analysis, verification	-

Table 5.4: Functional criteria (2/2)

Name	Import format	Export format	Platform requirements	Hardware constraints	Programming language
Values	formats	formats	macOS, Windows, Linux, web	none, <constraints>	none, <language>
Bee-Up	ADL, XML	ADL, XML	macOS, Windows, Linux	none	Java
BPMN.io (DMN)	DMN	DMN	web	none	JavaScript
Camunda	DMN, CMMN, BPMN, XML	DMN, PNG, SVG, XML, JPEG	all	none	Java
Cardanit	BPMN, DMN	DMN, PNG, PDF	web	none	-
Kogito (using Drools)	BPMN, DMN	DMN, SVG	web	none	Java
Signavio	BPMN, As-tah, jPDL 4	DMN, PNG, SVG, PDF, Drools	web	none	
Sparkling Logic (Pencil)	DMN, XSLX	DMN	web	none	-
Trisotech	text, TXT, XSLT, XBL, XSL, XML, DMN, VSD, VDX, VSDX	DMN1.1, DMN1.2, DMN1.3, JPEG, PNG, TIFF, PDF, word, HTML	web	none	-

Table 5.5: Technical criteria

Name	GUI	Usability	Effort required	Documentation	Other models
Values	minimal, extended, advanced		high, medium, low	good, poor, none	
Bee-Up	advanced	medium	high	poor	BPMN, UML, EPC, ER, Petri Nets
BPMN.io (DMN)	minimal	high	low	poor	BPMN, CMMN (separated tools)
Camunda	extended	high	low	good	BPMN, CMMN
Cardanit	extended	medium	low	poor	BPMN
Kogito (using Drools)	extended	medium	low	good	BPMN
Signavio	extended	high	low	good	BPMN, CMMN
Sparkling Logic (Pen- cil)	extended	medium	medium	good	-
Trisotech	advanced	high	low	good	BPMN, CMMN

Table 5.6: Feasibility and usability criteria

Bee-Up

Bee-Up works with scripts so that models can be executed, including the scripts. DMN model execution is not yet straightforward. The tool's learning curve is quite steep; thus, the execution functionality was not tested in this research. Bee-Up also allows model interconnection so that BPMN models can use the DMN data. Thus, direct execution of the complete DMN diagram seems to be unavailable.

Official DMN examples could not be imported since the import in the special ".DMN" format is not supported by the tool. The tool supports import and export in its ADL format and XML. Nevertheless, exported by the tool DMN diagrams in XML format cannot be imported by the tool again. Some DMN functionalities, such as this, seem to be in development. Models can be saved in different image formats: PNG, JPG, BMP, PCX, EMF, SVG.

BPMN.io

As explained before, this tool is supported/offered by Camunda, and the interfaces are the same; it looks like their DMN-js plugin is being used. Includes all main features. Testing went well, and interchange example could be imported. Input values do not have any attributes, such as type.

Decisions can be decision tables or literal expressions (use FEEL, JavaScript, ..), but no execution functionality. Literal expressions have six possible variable types (string, Boolean, integer, long, double and date), with no custom data types. Possible expression languages are FEEL, JUEL, JavaScript, Groovy, Python and JRuby. Diagram interchange is available, also with other DMN versions. There are no special functions available; simple, basic functionality. Some documentation on how to use the plugin is provided, but not on the modeler itself. The tool provides only modeling functionality, and no execution.

Camunda

For the installation, the user has to download and unzip Zip from the Camunda web page. It has an easy installation/startup, and no further steps are needed. There are versions for Mac, Linux, and Windows available. The new version only supports DMN 1.3. Diagram interchange is provided within the version. Model execution is shown in the web tutorial, not included in the demo version. The interface is simple but easy to understand. Jumping ahead, it has the same UI as BPMN.io, since BPMN.io is powered by Camunda. The execution functionality is not included in the open-source tool but can be tested in the web interface.

Cardanit

As a help, the program brings one to DMN specification 1.3, but model import for 1.3 does not work, just 1.2, but only partially. The import function is working, and output can be created. Output as XML, PNG and PDF is possible, as well as a nice PDF file:

that has the model and all the elements separately. The tool has a comfortable UI, is very flexible, and is easy to use. The new version allows the creation of decision tables. Data types can be defined, but no correctness checker is implemented (one can enter text into a number field). FEEL is being mentioned, but no execution, however. Therefore, it is not easy to distinguish the conformance level (1, 2 or 3), since every input is acceptable. The tool creates different beautiful diagrams. The appearance can be adapted to one's needs as the non-functional criterion.

Kogito

Kogito software contains all main modeling functionality. It is not very intuitive, especially for the input menu and creation of decision tables. Custom data types can be defined in the tool. Interchange examples can be imported, but some information, such as decision diagrams, is disappearing. As for the attributes, there is a possibility to add some, also using expression language. Wrong connections are not accepted while modeling - correctness check during the modeling process. An interchange of diagrams within the tool is possible - the user can add another diagram as a decision. The tool provides execution services that can be downloaded and used in the modeler.

Signavio

Input and output data from DRD will be automatically added to the decision table logic, which supports error prevention and the autocomplete function when typing. The tool includes all elements from the DMN 1.3; an extended List of data types, and the possibility to add a new data type. Import of other models is not possible. Export is working only for correct diagrams. The software has an analysis function, the "Uniqueness and completeness check".

The decision tables have the functionality to import and export values. Here the table can be imported as a text with different delimiters, or export for Drools or Test cases is possible.

Fields with the type "text" can be automatically transformed into enumerations if the data type is changed. A decision from another diagram from the collection can also be linked and reused in another diagram or as a service. Services can also be linked to decisions; no logic can be added.

Sparkling Logic (Pencil)

In Pencil, input and output data from DRD will be automatically added to the decision table logic that supports error prevention. Companies' expression language, "Sparkling language", is supported instead of the FEEL. A decision from another decision model can be linked to decisions. The visual appearance can be changed, and the color schemes can be edited. Import function was added in the new version; only exporting as DMN XML is possible now. Validation functionality for decisions was added as well. Pencil has a complex UI, that provides less usability, and it takes time to find some functions or elements; not self-explanatory. The decision table, rules, or text can be chosen for the

logic input. The tool makes it not possible or complicated to redo changes. As for the correctness checker – only the correct elements are allowed to be created/connected.

Trisotech

Trisotech has extended attribute functionality, files and links can be attached to the inputs.

Usability can be upgraded, but it is not very intuitive. The main functions, like switching from DRD view to the table view and back in not intuitively positioned; the symbol for the setting for the input data type is not intuitive. A quick guide and tutorials for FEEL are included in the tool, with easy access. Trisotech allows execution of the model with user inputs and saves the generated test cases. Saved test cases can be executed in bulk to test if the results are identical. Trisotech modeler contains DMN validation and analysis functions. It checks for incorrect elements (e.g., missing names) and missing rules and validates FEEL expressions. Nevertheless, the validation and execution of the imported Dish example didn't work. However, diagram interchange is available, also with other DMN versions.

The first version that was tested was 6.12.2; the second test was with version 10.15.3, which has upgraded some functionalities, as well as the DMN version from 1.2 to 1.3.

5.5 Results discussion

Eight tools were tested for the general technical criteria, tool availability, feasibility and usability, and some DMN-specific functionality, such as diagram interchange, execution, analysis, and simplification functionalities. As a result, the following general statistics for reviewed the tools can be outlined:

- 6 of 8 tools are web-based
- 4 tools do not request registration to use the service
- 4 tools are open source; the others have restrictions (30 days, possible to extend the research for some tools)
- 6 tools support DMN 1.3, 1 – 1.2, 1 – 1.1
- All eight tools have at least a simple correctness checker (mostly not possible to use wrong elements)
- 6 tools support DMN import, seven tools DMN export
- 5 tools fully support model interoperability
- 5 tools have a good documentation
- 7 tools support BPMN as well

5 Survey of DMN tools

- 4 tools have analysis functions
- none of the tools has simplification functionality

As presented in the research, most companies provide proprietary software as custom solutions or SaaS. Nevertheless, there are few open-source projects available.

Most of the presented software is still developing, and companies regularly provide users with new versions. All tools include the main modeling functionality, but some only support older DMN versions or have lower conformance levels than L3. Diagram interchange does not work for all tools for 100%, diagrams are not imported correctly, or some information can be missing. This field can be improved, for sure.

Most interesting were specific functionalities, such as execution, analysis, and simplification or verification and validation. Execution functionalities are provided by some reviewed tools, but mainly as a service and not included in the trial. Some tools have some verification functionality, which is already an improvement. Nevertheless, none of the tools had simplification functionalities implemented. As found in Chapter 3, there are some scientific tools, including some verification, validation, and simplification functionalities to their prototypes. During further research, another tool was found that includes some validation and verification functionality. It is provided by Redhat, based on Drools, and called "Redhat decision manager". Redhat supports the detection of redundancies, subsumption, conflicts, errors in Unique Hit Policy, deficiency, and incomplete tables and ranges.[36]

The result of this survey shows that current tools are missing some verification and validation functionalities, and simplification is not presented in the reviewed tools. How this gap can be filled, is presented in the following chapters.

6 DMN simplification

The complexity of DMN and its decision requirement diagrams' can become so high that diagrams would be challenging to read, having many inputs/facts or rules. Some tools, such as Open Rules, even provide automatic generation of DMN having big company data as input. Having this, there is a new challenge. One of DMN's purposes is to help businesses by simplifying workflows, so it has to be functional, clear and readable. The issue is that AI tools can create too many rules according to specific data, which becomes overfitting. A machine can hardly upgrade this - the human sense is needed here. In Open Rules, they use domain experts to reduce data and combine some of it in each iteration.

As withdrawn in Chapter 5, very few DMN tools support any kind of verification and validation of DMN models. Even if some rule simplification techniques are represented, none of the reviewed works support the simplification of the complete models (DRD). This could help in solving the issue. Some automated analysis of the models has to be included to help the modelers review the complex models and decision tables.

Furthermore, the functionalities of DMN-tools should be uplifted by verification and reduction to enhance reliability, safety, and efficiency issues[30]. These can be consequences of poor-structured and inconsistent DMN diagrams.

The question is if there are other ways to simplify DMN, and reduce its complexity by an algorithm. Research on this question brought just a few ideas. First, DMN can be seen as another representation of a Rule-based System, which is further explained in Section 6.1. Thus, it can be handled as one, and some methods of reasoning can help to simplify some parts of it (Section 6.2). Another possibility is to rearrange the rules using their geometric representation and the Bron-Kebrosch algorithm, as in [12] (Section 6.3). These methods and possible algorithms for DMN simplification are explained in this chapter, followed by ideas and suggestions for implementation.

6.1 Rule-based systems

Rule-based systems are a helpful way to represent human knowledge [24]. These are, in general, sets of rules within one domain or application. This representation can be a base for knowledge representation and application development. A good RBS (Rule-Based System) requires high domain expertise to represent the knowledge correctly for further use. An RBS is a set of rules that is built from preconditions and following conclusions. This can be represented in different languages, such as propositional or first-order logic. RBS can have diverse levels of complexity, can be hierarchical, and include several levels. RBS notation can be similar to a Relational database system that allows tabular or multi-tabular representation for complex systems. RBS can be represented as (1) decision

tables, (2) decision lists, (3) decision rules with control statements, (4) decision trees, or (5) binary decision diagrams. [30]

Following the RBS definition and the knowledge about DMN, DMN can be called a multi-layered Rule-Based System, where each decision table represents a set of rules. Thus, all the rules that can be applied to an RBS, can be used with DMN. This means logical verification and simplification can also be executed on a DMN, which brings new possibilities for DMN enrichment.

An advanced Rule-Based system, a so-called knowledge-based system, needs a human expert, as well as a knowledge base engineer, to represent the domain knowledge in the correct way. This workflow in this so-called knowledge-based system, or expert system, is presented in Figure 6.1. [24]

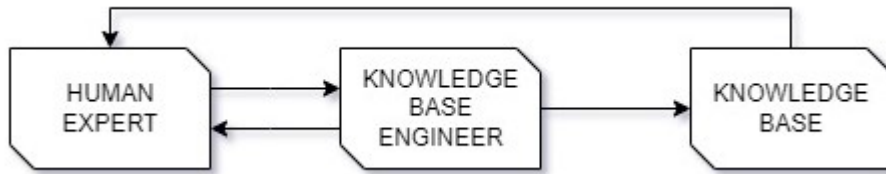


Figure 6.1: Expert system [24]

So DMN systems need an expert, where the knowledge-based engineer can be replaced/supported by the software. An intelligent system can represent knowledge base engineers. The software would suggest some upgrades for the DMN. The idea for the upgrade is to look into verification and simplification principles for the RBS.

6.1.1 RBS Verification

There are different ways to work on completeness and upgrade RBS. Thus, A.Ligeza in [30], highlights four verifiable characteristics of RBS:

1. Redundancy (identical rules, subsumed rules, equivalent rules, unusable rules)
2. Consistency (indeterminism, conflict, logical inconsistency)
3. Reduction (reduction of rules, canonical/specific reduction of rules, elimination of unnecessary attributes)
4. Completeness (logical completeness, specific completeness, detection of incompleteness, missing rules)

As shown in Chapter 3, some research works include DMN verification. Every one of the four concepts above is at least partially represented (following the review in [15]). The mentioned works also include corresponding research tools. In this work, the focus lies more on simplification characteristics: redundancy and reduction.

6.1.2 Redundancy

Rules representation can include redundant components. These are mostly not crucial for the system functionality but for the memory, speed, and ease of reading, understanding, and extending the rules. A.Ligeza in [30] defines two kinds of redundancy: logical and operational or functional. Logical redundancy is caused by similar or the same rules, one of which can be removed without loss of content. These can be *identical rules*, *subsumed rules* or *equivalent rules*. If a rule can be removed without causing changes in systems behavior - it is operational/functional redundancy. These are the *unusable rules* that are never being fired.[30]

Redundant rules can be identical or similar rules corresponding to the same set of cases. Here, only one of each rule should exist. There also can be redundancy within the inputs of the rule, for example, unnecessary inputs. This can be identified using different techniques from relational databases, e.g., but should also be confirmed by a domain expert.[30]

As for subsumed rules, that means one rule can be seen as a subset of the other - less general; it is primarily the less general rule that can be removed, but also a more general to cover some exceptional cases. However, equivalent rules cover the same cases since both rules include the other. Subsumption verification is covered by PROLOG, which can be detected by general theorem proving or by algebraic comparisons (for tabular systems).[30]

The unusable rules happen when a precondition is never satisfied - for DMN, input attributes or attribute combinations can never happen. [30]

Thus, applying this to DMN, the following redundancies can be detected:

- Identical rules, with same inputs and same outputs,

$$\begin{array}{l} A \wedge B1 \rightarrow C \\ A \wedge B1 \rightarrow C \end{array}$$

- Other redundant rules, e.g., irrelevant input,

$$\begin{array}{l} A \wedge B1 \rightarrow C \\ A \wedge B2 \rightarrow C \end{array}$$

B1 and B2 are irrelevant

- Subsumed and equivalent rules - can be detected using theorem proving,

$$\begin{array}{l} A1 \in A \\ A \rightarrow C \\ A1 \rightarrow C \text{ - irrelevant} \end{array}$$

- Unused rules that can never be fired.

6.1.3 Reduction

Reduction stays for minimization, optimization, and simplification of the RBS. In case of reduction, the result, however, has to be equivalent to the raw version. Here the rules can be modified, separated, combined, and reorganized between each other into a reduced elegant solution.

There are several kinds of reduction, explained down below.

Total and partial reduction

The idea is to minimize the number of rules. This can be reached by combining rules with the same conclusion (output for DMN). There are a few cases where the merge can be easily done by implementing simple algorithms.[30]

- Dual resolution principle

If only one precondition is changing, and the other precondition and all outputs are identical - the rules can be combined into one set.

$$\begin{array}{rcl}
 A \wedge B1 & \rightarrow & C \\
 A \wedge B2 & \rightarrow & C \\
 & \dots & \\
 A \wedge Bn & \rightarrow & C \\
 \hline
 A & \rightarrow & C
 \end{array}$$

- Attribute reduction

If attributes don't influence the output, the rules can be combined without them.

$$\begin{array}{rcl}
 A1 \wedge B1 & \rightarrow & C \\
 A2 \wedge B2 & \rightarrow & C \\
 & \dots & \\
 An \wedge Bn & \rightarrow & C \\
 \hline
 A1 \wedge A2 \wedge \dots \wedge An & \rightarrow & C
 \end{array}$$

Specific partial reduction

For the cases where complete reduction cannot be applied, a specific partial reduction can help. Here a new rule can be introduced to combine the other.

$$\begin{array}{rcl}
 & \text{Assuming} & \\
 B1 \vee B2 \vee \dots \vee Bn & | = & B \\
 A \wedge B1 & \rightarrow & C \\
 A \wedge B2 & \rightarrow & C \\
 & \dots & \\
 A \wedge Bn & \rightarrow & C \\
 \hline
 A \wedge B & \rightarrow & C
 \end{array}$$

6.2 Reasoning

Reasoning in logic is the process of problem-solving or finding relations between the rules that connect to the conclusion. Forward chaining and Backward chaining are two methods for applying reasoning to expert systems. [2]

In Forward chaining, decisions are being derived from the facts. It is a bottom-up process. It can be applied when lots of goals and fewer facts. The result of the process is a/are found goal/goals. [24][2]

Backward chaining reasoning method starts from the goals and proceeds to the facts. It is a top-down process. In Backward chaining - few conclusions and much information (known facts). It is used when specific goals should be reached or checked. [24][2]

Backward chaining on DMN

As explained above, Backward chaining searches through the decision tree, starting from the goals, moving through the rules to the facts that would get to the goal. [2] The aim is to find all the rules that can satisfy the given goal. [24] The goals in DMN are the top decisions, and the facts are the input data. Going through the decision model from the top decision to the inputs would chain all rules being fired to reach the decision. This means that if Backward chaining is performed on all rules from the top decision, all used rules can be highlighted in the process. This way, all unused rules can be discovered.

Drools provide Backward chaining functionality in their Java service. It can be performed on a rule system in the DRL - Drools Rule Language. The function *fireAllrules()* performs backward chaining and shows all used rules. Theoretically, DMN rules can be transformed into DRL to perform reasoning. There is one difficulty in the DMN model tree, compared to a simple rule system - DMN can include complex rules and even code in JavaScript or FEEL, e.g., that could restrict the execution of the resolution with the help of Drools.

The main idea of performing Backward chaining on DMN (without Drools) is to create an algorithm to go through all decisions from top to bottom and recursively find all used/unused rules in the chain.

6.3 Geometric DMN simplification

Calvanese and others introduced another method of DMN simplification in [12]. The authors introduced a method to reduce DMN by reducing overlapping rules and detecting gaps on the DMN table level. The method includes a geometric representation of the rules and a rearrangement of the rectangles that represent the rules. [12]

The main algorithm to detect overlapping rules goes as follows: [12]

1. map the rules into hyper-rectangles (N-dim)
2. search for rules overlaps/intersections of rectangles
3. eliminate not overlapping rules

6 DMN simplification

4. generate closed groups using Bron-Kebrosch algorithm
5. output the groups as separate sets of overlapping rules

Algorithm for the rule simplification in 3 steps: [12]

1. Divide rules into groups based on the same output
2. Construct adjacency graph (for rectangles that share a common side), using Tarjan's algorithm
3. Split rectangles, so they share only one complete side
 - run a sweep line along each dimension in turn
 - re-merge

Within the research, the authors of [12] implemented a tool with described functionalities. The algorithms are based on the geometric representation of the rules and are made to be used within the decision table, not the complete DRD.

6.4 Simplification algorithm

An even greater algorithm for DMN simplification can be created, having all the researched possibilities, and discovered algorithms. As the basis, the following methods can be taken:

- reduction methods of RBS for rule reduction within a decision table;
- Backward chaining for detecting the unused rules through the complete DMN-model;
- geometric simplification by [12].

With all methods described, a complex skeleton for an algorithm can be created:

1. Check for redundancies
 - a) Identical rules
Compare all rules between each other and detect duplicates.
 - b) Subsumed rules
Theorem proving
 - c) Find unused rules, Irrelevant input or unreachable goals
Backward chaining, forward chaining
2. Check for total and partial reduction:
 - a) Dual resolution
 - b) Attribute reduction

c) Specific partial reduction

3. Reorganize overlapping rules, using geometrical representation, following [12]

All checks from the list can happen separately or in a different order. This can be further determined if the order makes a difference by comparing the possibilities in a case study. It can be considered future work.

7 Implementation

The following chapter includes the leading decisions on the implementation process and the results of the implemented prototype. The technology and the architecture are shown, and finally, the result of the implementation is presented and discussed.

As discussed in the current work, one of the less researched areas in DMN is the simplification of models. The previous chapters highlighted the current research points, focusing on simplification possibilities for DMN models. Thus, the following prototype should extend the current possibilities of DMN and show one of the growth directions for the existing tools and plugins.

There are several possibilities to implement the suggested DMN simplification in a completely new prototype. Some open source tools provide their modeling plugins: e.g., BPMN.io, Drools (red hat); some companies provide some execution functionalities: e.g., Drools, Camunda. Therefore, there are already a few works with the tools implementations, based on these possibilities, that were mentioned in Chapter 3. The following chapter shows the other way, how the existing solutions can work together in a completely new tool, and what the open source stage provides nowadays.

In general, the main functions of the implemented prototype are:

- DMN modeling functionality, using the BPMN.io plugin;
- Test case generation;
- Test case execution (single test cases and bulk execution) and
- suggested simplification functionality.

The following sections explain these functions and the implementation process in detail.

7.1 Technology and functionality

The current implementation's main aim is to discover possibilities for the extension of DMN tool functionalities. The overview of the project environment can be seen in Figure 7.1. First, the `dmn-js` plugin (<https://github.com/bpmn-io/dmn-js>), offered by BPMN.io, was considered as the basis for the prototype. This is a plugin that can be used within the JavaScript project. It includes a simple DMN modeler.

Next, for the complex functionalities, a Java back-end was considered. The back-end includes several REST APIs that can be called from the JavaScript application. The APIs include the calls to generate test cases, execute test cases, calculate execution complexity, and simplification functionality. This will be shown in detail in the next section.

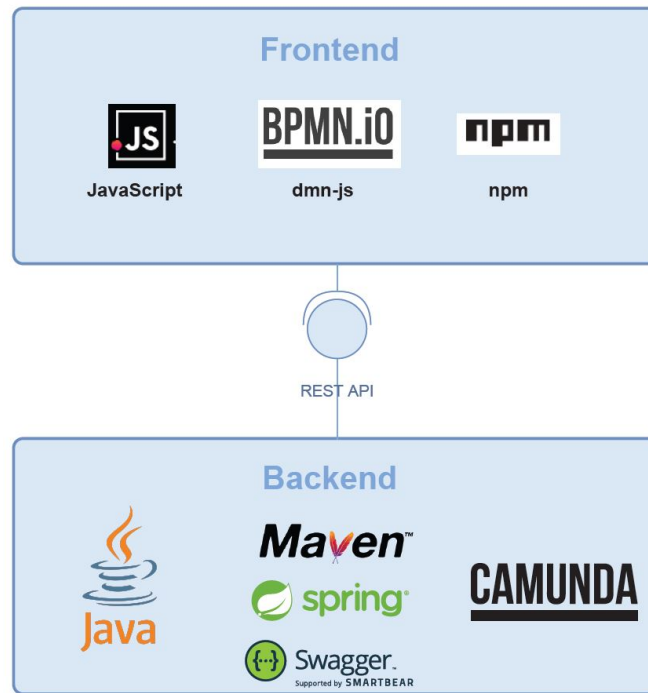


Figure 7.1: Project environment

The back-end uses Spring Boot (<https://spring.io/>) for the web application nature. The dependencies are managed with Apache Maven (<https://maven.apache.org/>) and Swagger (<https://swagger.io/>) provides the Rest APIs. As for the DMN structure, DMN Engine from Camunda (<https://docs.camunda.org/manual/7.16/user-guide/dmn-engine/>) is used, as well as its provided execution functionality.

The overview of prototype functionalities is shown in the Use-case diagram in Figure 7.2.

Use-Case 1: Create model

This use case includes the modeling stage of the prototype. Users can submit an XML-file of an existing DMN-model or create one from scratch. The model will be used in the other use cases as input.

Use-Case 2: Generate test cases

The application generates test cases with all possible input combinations given the model. A user can set a threshold to receive fewer test cases.

Use-Case 3: Execute test case

The generated test cases can be executed: one-by-one or in bulk. Users can execute a single test case. The result will be the result of the top decision and will be shown in the browser. The results of the bulk execution will be saved in a file and shown in the browser.

Use-Case 4: Calculate complexity

The user can calculate the complexity of the model by the amount of possible unique test cases based on the Cartesian product of all inputs.

Use-Case 5: Simplify

The simplification use case can include all simplification methods from Chapter 6. The current prototype includes a checkup for the unused rules. The prototype gives a user suggestions about rules that are not being used and can be removed (or maybe the model has to be extended, so the rule is used).

The implemented algorithms are further described in the next section.

7.2 Prototype

Within this thesis, a tool prototype with several DMN-related functions was implemented. The implementation process focuses on discovering current possibilities, existing open-source tools, and plugins that can be combined with some newly-implemented algorithms.

The implementation process of the front-end and the technological decisions were pretty clear from the beginning: the `dmn-js` plugin supports DMN modeling and was used in other research works, making the results comparable. The plugin has proven to be easy to install and simple. The implementation was based on the official documentation, using the official template. Based on the tool, the remaining UI was built. As can be seen on Figure 7.3, the UI includes the **Modeler (1)**, a **Control panel (2)**, a **Test-case overview (3)** and a **Result block (4)**. As the user interacts with the UI, different APIs are called. The implementation was done in HTML and JavaScript.

The back-end includes a more complex structure and complete logic. It is a Java Spring Boot application with a Maven nature. With the help of Swagger it provides APIs for the test case generation, execution and simplification.

The APIs are defined in **controller** package. There is an *ExecutionController* and *SimplificationController*. The controllers call the services from the **services** package. It includes *TestcaseGenerationService*, *ExecutionService* and *SimplificationService*, which are using the packages **generator**, **dmn.executor**, and **simplifier** accordingly. The complete class structure can be seen in Figure 7.4. The structure is scalable for different approaches, like different simplification approaches or another execution service.

It makes sense to present the main functions in detail separately.

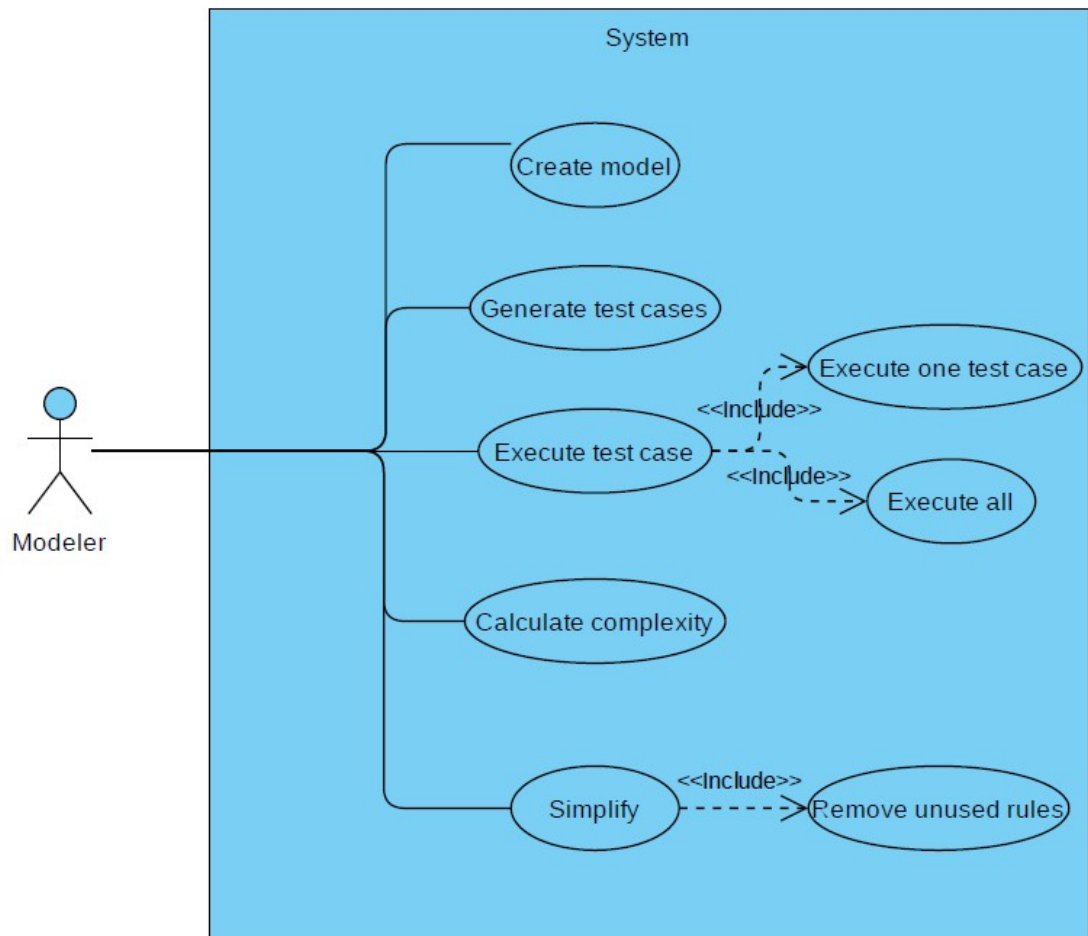


Figure 7.2: Use-case diagram

7.2.1 Test case generation

The test case generation provides all possible test cases created as a Cartesian product from the inputs and their values from all rules.

E.g., a decision has two inputs: Input1 and Input2. Input1 is a number, and Input2 is a string. The decision table has eight rules. The rules have four different number ranges for Input1 and only two unique values for Input2. This simple example makes $4 \times 2 = 8$ unique input combinations. This also means the complexity of the case is 8.

The implementation searches for the inputs in the XML document and connects them with the possible values. As for the numeric values, from the ranges, one number is being taken to create one case, as inputs are numbers, not ranges.

The results of the test case generation are saved into the file system line-by-line. This allows the processing of even complex DMNs.

Figure 7.5 shows the functionality in the UI. After clicking the button, API returns

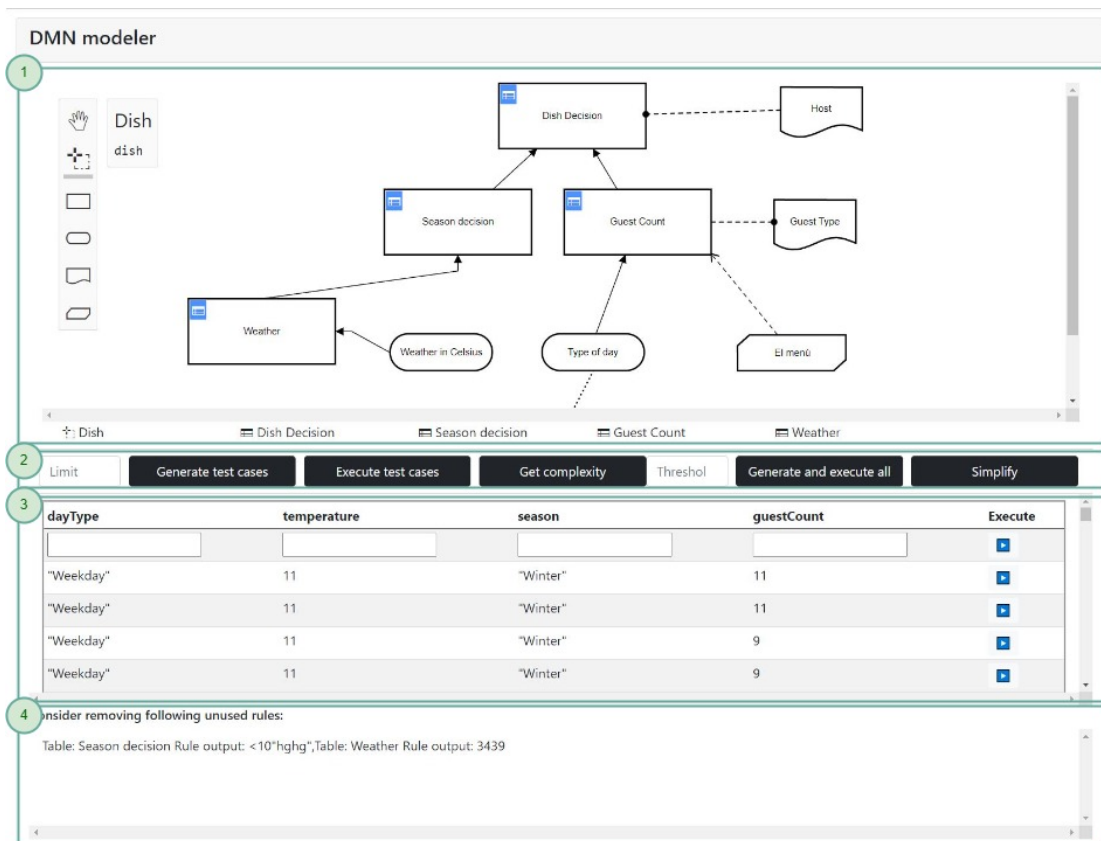


Figure 7.3: UI overview

58



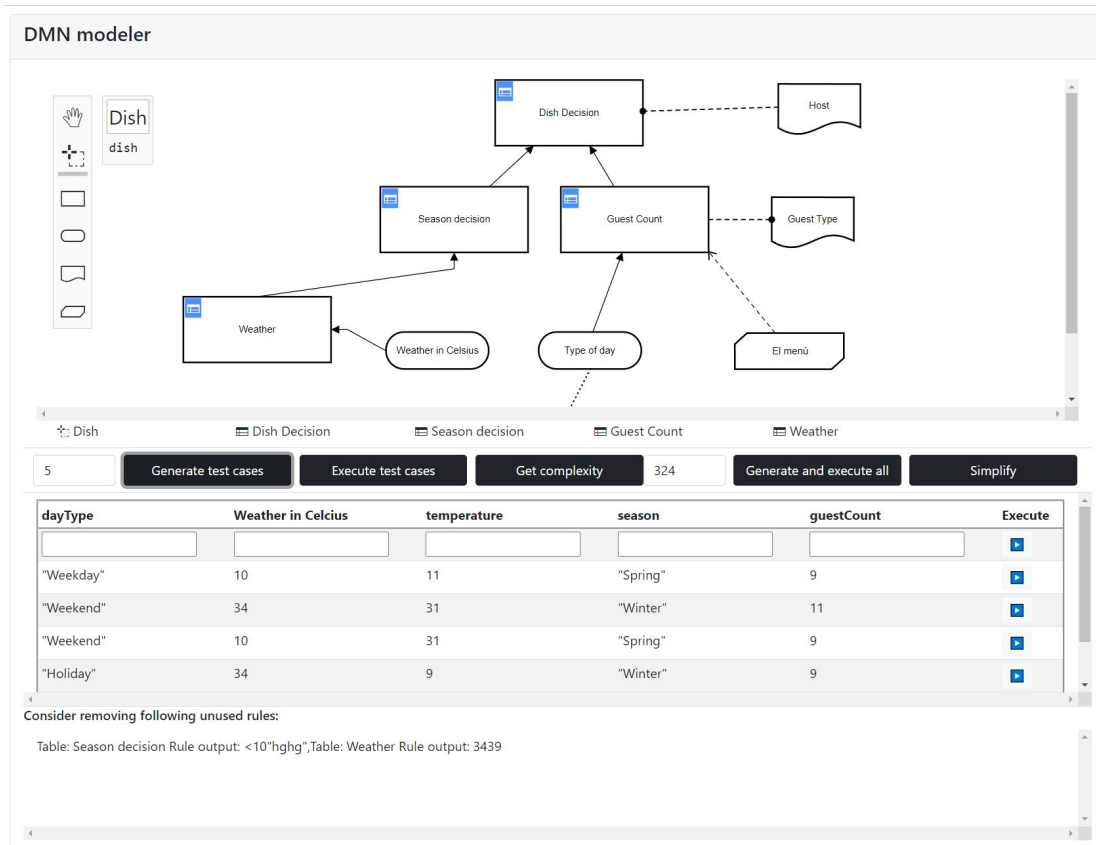


Figure 7.5: UI generate test cases

the list of test cases that are shown below the modeler in the test-case overview. The columns are dynamically generated from the received inputs. Each test case includes values for all inputs and a button for execution. Users can also enter and execute his/her own use case in the first row. The number of received test cases can also be reduced by entering a threshold value. The test cases will be taken from different parts of the test case list, so the results are well-mixed.

Figure 7.6 highlights the complex functionality that shows the maximal number of created test cases. It can be used before test cases are generated to determine the complexity and set up the threshold if needed.

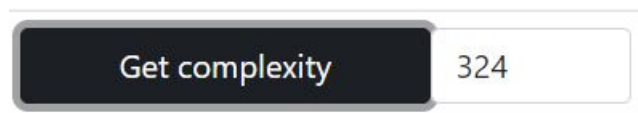


Figure 7.6: UI get complexity

7.2.2 Test case execution

There are two possible scenarios for the execution: execute one test case and execute all. For the first scenario, the DMN model in XML format and the input values from the test case are proceeded to and executed with the DmnEngine by Camunda. The execution has to be done on the top decision. Otherwise, given the input, each decision will be executed, which leads to unnecessary results from the other decisions. The following code snippet shows the execution of one single decision:

```

1 private List<String> evaluateSingleDecision(DmnDecision decision,
    VariableMap variables) {
2     for(DmnDecision requiredDecision : decision.getRequiredDecisions()){
3         variables.putAll(getVariableMap(evaluateSingleDecision(
4             requiredDecision, variables)));
5     }
6     DmnDecisionResult result = dmnEngine.evaluateDecision(decision,
7         variables);
8
9     List<String> resultArray = new ArrayList<String>();
10    if (!result.collectEntries(decision.getKey()).isEmpty()) {
11        String res = decision.getKey() + ":" + result.collectEntries(decision
12            .getKey()).get(0);
13        System.out.println(res);
14        resultArray.add(res);
15    }
16    return resultArray;
17 }

```

Listing 7.1: Implementation of test case execution

For bulk execution, the file system is being used. If test cases were previously generated and saved into the file system – the file is read by the program line-by-line, executed, and saved into the new file with the results. If a bulk test case generation with execution is requested – the possible test cases are created and saved into the file and then executed, as shown in the process diagram in Figure 7.7.

7.2.3 Simplification

Current research showed that the simplification part of DMN is less represented. This was highly perceived during the work on the current solution. Having the research on DMN simplification from Chapter 6, the main focus went to the Backwards Chaining. With the help of Backwards Chaining, unused rules can be detected, which can help to reduce DMN complexity. After some research, the Drools solution was found. Drools (<https://www.drools.org/>)[40] has a service equivalent to the Backwards Chaining. The function *"fireAllrules()"* delivers all used rules, from the top decision on. The difficulty came with the input format for the Drools: the service needs a DRL (Drools Rule Language) format as an input. This transformation algorithm has to be implemented, or another possible way could be Signavio. There is a possibility to export DRL from the DMN diagram. This approach was considered out of scope. Another way was to

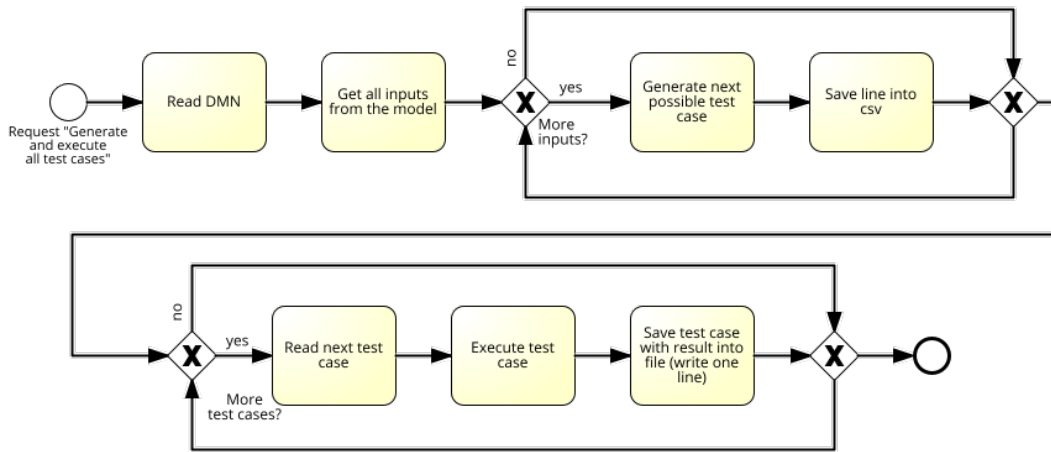


Figure 7.7: Test case generation and execution

implement the Backwards Chaining from scratch. This created the following algorithm:

Data: DMN model

Result: how to find unused rules in DMN model

Get the top decision;

while *no required decisions* **do**

 Get all inputs from the rules (1);

 Get all required decisions;

while *no required decision* **do**

 Get all outputs from the required decision (2);

 Compare the outputs (2) with the inputs (1);

 Find outputs (2) with no fitting inputs (1);

 Add the corresponding rules to the unused list;

 Temporarily remove the rules from decision;

 Repeat from the top recursively with the required decision;

end

end

Algorithm 1: DMN Backwards Chaining algorithm

The algorithm was implemented in the prototype. The functions and models from the DmnEngine from Camunda were used. The code snippet below shows the main functionality. Not shown *"getDecisionDefinitionInputs"* method, called in line 22, extracts all inputs with the names and values from the decision table and *"getUnusedDecisionDefinitionOutputs"* method from line 25 extracts the outputs, search for them in the input list and add the ones not found into the list of the unused rules.

7 Implementation

```
1 public List<String> simplify(String dmnxml) {
2
3     DmnModelInstance dmnmModelInstance = Dmn.readModelFromStream(new
4         ByteArrayInputStream(dmnxml.getBytes()));
5     List<DmnDecision> decisions = dmnmEngine.parseDecisions(dmnmModelInstance
6     );
7     List<String> unusedRules = new ArrayList<String>();
8
9     DmnDecision decision1 = decisions.get(0);
10
11     if(decision1.isDecisionTable()) {
12         unusedRules = GetUnusedRules(dmnmModelInstance, decision1);
13     }
14
15     return unusedRules;
16 }
17
18 private List<String> GetUnusedRules(DmnModelInstance dmnmModelInstance,
19     DmnDecision decision) {
20     List<String> unusedRules = new ArrayList<String>();
21
22     Collection<DmnDecision> requiredDecisions = decision.
23         getRequiredDecisions();
24
25     List<String> inputs = getDecisionDefinitionInputs(dmnmModelInstance,
26         decision.getKey());
27
28     for(DmnDecision dec : requiredDecisions){
29         unusedRules.addAll(getUnusedDecisionDefinitionOutputs(
30             dmnmModelInstance, dec.getKey(), inputs));
31
32         unusedRules.addAll(GetUnusedRules(dmnmModelInstance, dec));
33     }
34
35     return unusedRules;
36 }
```

Listing 7.2: Implementation of simplification algorithm

The implemented algorithm finds unused rules between the two closest decisions and not through the whole "chain". For even better results, the unused rules should be deleted from the decision for the subsequent iterations, as shown in the Algorithm 1. This can be implemented in the following research iterations.

After clicking of "Simplify"-button, the resulting block shows the table and the rule that is not used in the next decision. The program only suggests that the modeler remove the eventually unnecessary rules.

7.3 Result discussion

This chapter showed the possible open source solutions working together. It showed the usage of the dmnm-js modeler, Camunda DmnEngine, and possible extension of given

functionalities. A prototype was implemented that included functionalities for modeling, test case generation, execution of test cases and simplification, using the suggested backwards chaining algorithm. The prototype shows some possibilities that can be reused and extended by tool owners and researchers.

Thus, one suggestion for the possible uplifting of DMN-value came up during the research and implementation process. The idea is to upgrade some great existing tools, use already supported features, and include other existing solutions. Those combining different tools, a nice simplification feature can be implemented into Signavio, for example. As mentioned above, there is a backwards chaining functionality implemented by Drools. As an input, a model in DRL (Drools Rule Language) is needed. Signavio supports the export of DMN models as DRL. What if the companies could work together? A simplification functionality can be added to Signavio. The following algorithm can support the implementation:

1. User clicks on the "Simplify"-button
2. In back-end convert DMN-model to DRL
3. Use the DRL-model in a KIE-session (Drools) to perform Backwards chaining, using the "fireallrules()" -function
4. Highlight all rules that were not fired as possibly unimportant since they are unused
5. Suggest simplification to the user

As found out from the Survey, some tool owners are already extending their software with verification and validation functionalities. The trend is set right, and it is possible that more amicable solutions will see the light by the time of this publication. Furthermore, let us see if this prototype can be a part of this process.

8 Case-based evaluation

The practical part of the research aimed to implement a DMN tool, including some DMN functionalities, such as test case generation, execution, and simplification of the DMN models. This case-based evaluation will be focused on the evaluation of the simplification functionality. Thus, to evaluate the results, a difference in the complexity of the models will be determined. For this, the complexity of the original model should be calculated, and the complexity after applying the simplification algorithm and delta will be calculated in the end.

Since the simplification in the prototype happens on the rules level, the complexity can be determined in two ways:

1. Number of rules;
2. Number of possible input combinations - Cartesian product.

For the Cartesian product, all possible values that are used in the rules are combined. For the intervals - only one value from the interval is used.

The essential requisite for the evaluation is a DMN model. For the test model, two models will be tested. Every model will have some unused rules. The models will be imported into the tool, and the simplification function will be executed. Suggested rules will be removed, and, as a result, the difference in complexity will be calculated and compared to the expected result.

To test the functionality of the tool, an official DMN example from [34] is used. Given DMN decides on the special dish, depending on the type of the day and the weather. The model is presented on the Figure 8.1. It has three decision tables and two input data elements. The calculated complexity of the possible test cases for the model was 162. 162 test different test cases can be generated for this model by generating the Cartesian product.

A few additional rules were added to the actual model, and some rules were changed to have a smaller range of values that fulfill the rule requirements. The new tables are presented in the Figure 8.2. It has three new rules and some changes, where 2 of the new rules do not fit any requirements of the dish decision. The complexity of the new model was 360.

On the updated model, a simplification function was applied.

Simplification function suggested:

Consider removing, following unused rules:

Table: Guest Count Rule content: "Wedding"100,

Table: Season decision Rule content: [10..20]"Autumn"

8 Case-based evaluation

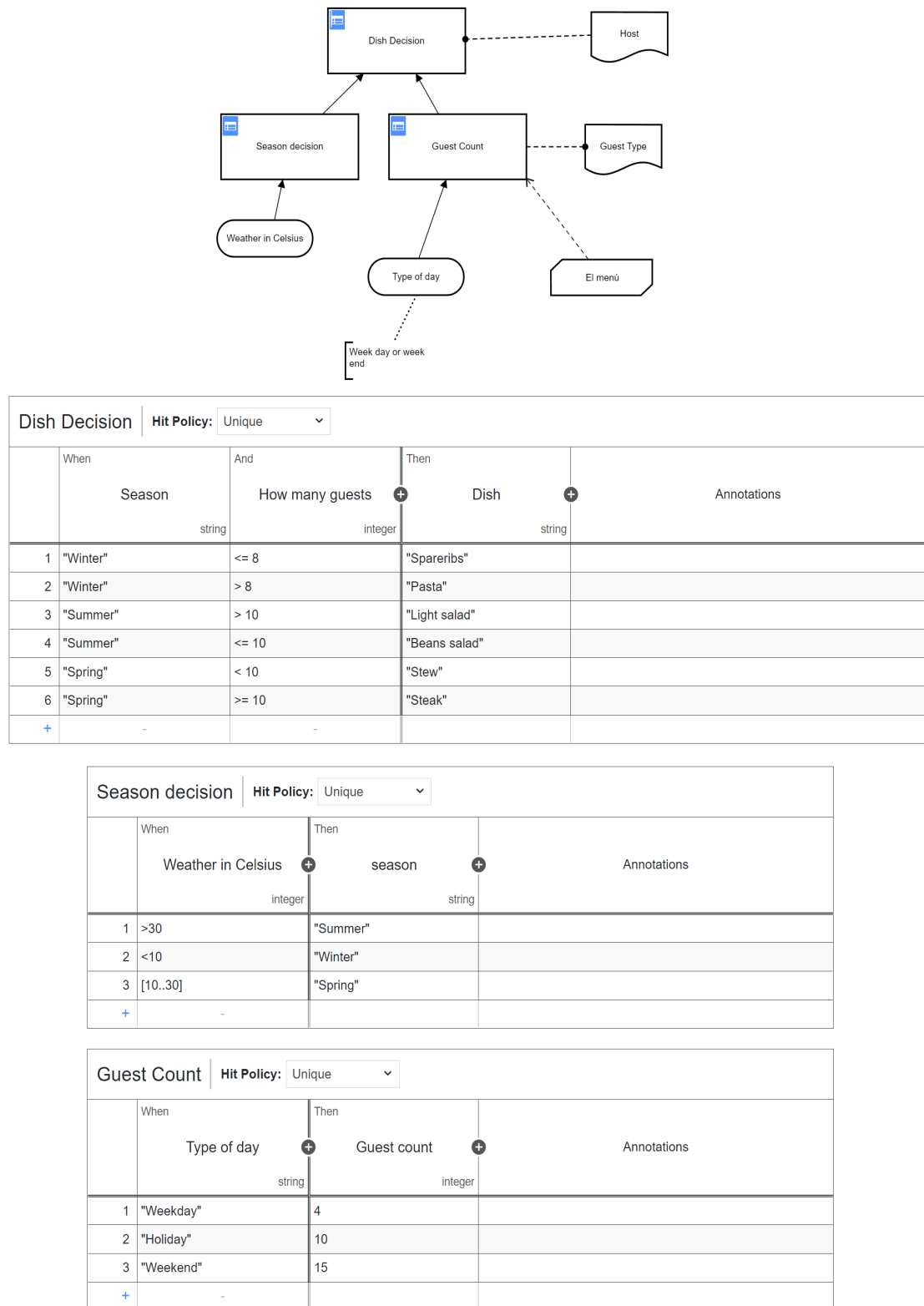


Figure 8.1: Test case 1

Dish Decision Hit Policy: Unique				
	When	And	Then	
	Season	How many guests	Dish	Annotations
	string	integer	string	
1	"Winter"	<= 8	"Spareribs"	
2	"Winter"	[8..15]	"Pasta"	
3	"Summer"	[10..12]	"Light salad"	
4	"Summer"	<= 10	"Beans salad"	
5	"Spring"	< 10	"Stew"	
6	"Spring"	[10..12]	"Steak"	
+	-	-		

Season decision Hit Policy: Unique				
	When	Then		
	Weather in Celsius	season		Annotations
	integer	string		
1	>30	"Summer"		
2	<10	"Winter"		
3	[20..30]	"Spring"		
4	[10..20]	"Autumn"		
+	-			

Guest Count Hit Policy: Unique				
	When	Then		
	Type of day	Guest count		Annotations
	string	integer		
1	"Weekday"	4		
2	"Holiday"	10		
3	"Weekend"	15		
4	"Special day"	20		
5	"Wedding"	100		
+	-			

Figure 8.2: Test case 1 with added rules

After removing the 2 suggested rules, new complexity was calculated. The complexity became 216.

Number of rules before simplification	15
Number of rules after simplification	13
Complexity before simplification	360
Complexity after simplification	216
Expected number of rules to be removed	2
The actual number of rules removed	2

Table 8.1: Test 1 results

Table 8.1 shows the overview of the simplification results.

The current thesis’s background work included further test case research. The aim was to find complex DMN models that could be tested with the tool. Another idea is to find the test cases from the other tools with verification or simplification functionalities and compare the results. <https://dmn-tck.github.io/tck/tests.html> includes different kinds of tests on different levels. Although, there are no tests for the compatibility in the rule usage between decisions. Different models were found on the internet, but most were too simple (didn’t have more than one decision table), which is not covered by the implemented tool; some models could not be imported due to the wrong DMN version or corrupted DMN file. Future work should include requests for diverse complex DMN models to companies that actively use DMN.

Results

Tests showed, that the complexity on the rule level can be reduced by removing unused rules. The complexity of big models, especially models, that were automatically generated, can be reduced by applying different simplification methods, such as presented backwards chaining. Even better results can be expected, if different methods can be applied, as suggested in the algorithm from DMN simplification. This test case showed the importance of simplification functionality for the real use cases.

9 Conclusion

Answering the research questions from the Introduction, it was proved that the current research on DMN is weaker regarding the verification, validation, and simplification of DMN models. Current DMN tools have many functionalities but predominantly fail on these topics. This thesis showed the research status in these fields, including the abovementioned gaps in the topics.

This research produced some valuable artifacts that can be reused in future works:

- Status of DMN research
- Systematic Tool Review - process description
- Criteria catalog for tool reviews (specifically DMN, but can be adapted to the other tools)
- Survey on currently available tools
- Theoretical basis for DMN simplification, with algorithm suggestions
- Implemented prototype with several functionalities

In the scope of the current work, a few simplification methods for detection redundancy and reduction were introduced, as well as Backwards chaining application on DMN. The prototype included Backwards chaining method, which was tested in the Case-based evaluation.

To summarize the results, DMN and its possibilities are growing, but there is still room for further improvement. DMN verification and validation can be further researched, and more tools should be extended with the abovementioned functionalities. This will help to work with big tables, automatically generated tables from some big data, and other complex solutions, where a modeler is only a person that cannot manually analyze the volume of the model.

Based on this research, different methods for DMN verification can be reviewed and implemented for future work. The complete simplification algorithm, that was suggested in DMN simplification, can be implemented into the tool. The results of the analysis can gain more usability and show proposed changes directly on the model and maybe with an ability to apply the changes automatically.

If DMN keeps growing at the same pace as now, or even faster, this could be used by even more companies and applied in more use cases!

Bibliography

- [1] Sameera Abar, Georgios K Theodoropoulos, Pierre Lemarinier, and Gregory MP O'Hare. Agent based modelling and simulation tools: A review of the state-of-art software. *Computer Science Review*, 24:13–33, 2017.
- [2] Ajlan Al-Ajlan. The comparison between forward and backward chaining. *International Journal of Machine Learning and Computing*, 5(2):106, 2015.
- [3] Ahmad Alturki. Information system design theory: A lifecycle perspective. In Jeffrey Parsons, Tuure Tuunanen, John Venable, Brian Donnellan, Markus Helfert, and Jim Kenneally, editors, *Tackling Society's Grand Challenges with Design Science*, pages 186–194, Cham, 2016. Springer International Publishing.
- [4] Ekaterina Bazhenova, Francesca Zerbato, Barbara Oliboni, and Mathias Weske. From bpmn process models to dmn decision models. *Information Systems*, 83:69–88, 2019.
- [5] Arie Ben-David. Rule effectiveness in rule-based systems: A credit scoring case study. *Expert Systems with Applications*, 34(4):2783–2788, 2008.
- [6] Thierry Biard, Alexandre Le Mauff, Michel Bigand, and Jean-Pierre Bourey. Separation of decision modeling from business process modeling using new “decision model and notation”(dmn) for automating operational decision-making. In *Working Conference on Virtual Enterprises*, pages 489–496. Springer, 2015.
- [7] Alexander Bolotov, Vyacheslav Bocharov, Alexander Gorchakov, and Vasilyi Shangin. Automated first order natural deduction. ISCAI, 2005.
- [8] Dominik Bork, Robert Andrei Buchmann, Dimitris Karagiannis, Moonkun Lee, and Elena-Teodora Miron. An open platform for modeling method conceptualization: The omilab digital ecosystem. *Commun. Assoc. Inf. Syst.*, 44:32, 2019.
- [9] Dominik Bork, Dimitris Karagiannis, and Benedikt Pittl. How are metamodels specified in practice? empirical insights and recommendations. In *24th Americas Conference on Information Systems, AMCIS 2018, New Orleans, LA, USA, August 16-18, 2018*. Association for Information Systems, 2018.
- [10] Dominik Bork, Dimitris Karagiannis, and Benedikt Pittl. Systematic analysis and evaluation of visual conceptual modeling language notations. In *12th International Conference on Research Challenges in Information Science, RCIS 2018, Nantes, France, May 29-31, 2018*, pages 1–11. IEEE, 2018.

Bibliography

- [11] Dominik Bork, Dimitris Karagiannis, and Benedikt Pittl. A survey of modeling language specification techniques. *Information Systems*, 87:101425, 2020.
- [12] Diego Calvanese, Marlon Dumas, Ülari Laurson, Fabrizio M. Maggi, Marco Montali, and Irene Teinemaa. Semantics, analysis and simplification of dmn decision tables. *Information Systems*, 78:112–125, 2018.
- [13] Camunda. Camunda. <https://camunda.com/>.
- [14] Camunda. Choosing the dmn hit policy. <https://camunda.com/best-practices/choosing-the-dmn-hit-policy/>.
- [15] Carl Corea, Jonas Blatt, and Patrick Delfmann. A tool for decision logic verification in dmn decision tables. In *BPM (PhD/Demos)*, pages 169–173, 2019.
- [16] Holger Eichelberger, Yilmaz Eldogan, and Klaus Schmid. A comprehensive survey of uml compliance in current modelling tools. *Software Engineering 2009*, 2009.
- [17] D. Etinger, S.D. Simić, and L. Buljubašić. Automated decision-making with dmn: from decision trees to decision tables. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1309–1313, 2019.
- [18] Kathrin Figl, Jan Mendling, Gul Tokdemir, and Jan Vanthienen. What we know and what we do not know about dmn. *Enterprise Modelling and Information Systems Architectures (EMISAJ)*, 13:2–1, 2018.
- [19] Romain Franceschini, Paul-Antoine Bisgambiglia, Luc Touraille, Paul Bisgambiglia, and David Hill. A survey of modelling and simulation software frameworks using discrete event system specification. In *2014 Imperial College Computing Student Workshop*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2014.
- [20] Carl-Christian Grohé, Carl Corea, and Patrick Delfmann. Dmn 1.0 verification capabilities: An analysis of current tool support. In *International Conference on Business Process Management*, pages 37–53. Springer, 2021.
- [21] Faruk Hasić, Carl Corea, Jonas Blatt, Patrick Delfmann, and Estefan ´ia Serral. A tool for the verification of decision model and notation (dmn) models. *RCIS 2020. Lecture Notes in Business Information Processing*, vol 385., 2020.
- [22] Faruk Hasić and Jan Vanthienen. Complexity metrics for dmn decision models. *Computer Standards & Interfaces*, 65:15–37, 2019.
- [23] Imre Horváth et al. Comparison of three methodological approaches of design research. In *DS 42: Proceedings of ICED 2007, the 16th International Conference on Engineering Design, Paris, France, 28.-31.07. 2007*, pages 361–362, 2007.
- [24] Dimitris Karagiannis and Patrik Burzynski. *Logical Foundations of Knowledge Engineering*. Faculty of Computer Science, University of Vienna.

- [25] Dimitris Karagiannis and Harald Kühn. Metamodelling platforms. In *EC-web*, volume 2455, page 182. Citeseer, 2002.
- [26] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33(2004):1–26, 2004.
- [27] Krzysztof Kluza, Weronika T. Adrian, Piotr Wiśniewski, and Antoni Ligęza. Understanding decision model and notation: Dmn research directions and trends. In Christos Douligeris, Dimitris Karagiannis, and Dimitris Apostolou, editors, *Knowledge Science, Engineering and Management*, pages 787–795, Cham, 2019. Springer International Publishing.
- [28] U Laurson. *Verification and Simplification of DMN Decision Tables*. PhD thesis, Master’s thesis, University of Tartu, Tartu, Estonia, 2016.
- [29] Üleri Laurson and Fabrizio Maria Maggi. A tool for the analysis of dmn decision tables. In *BPM (Demos)*, pages 56–60, 2016.
- [30] Antoni Ligęza. *Logical foundations for rule-based systems*. Studies in computational intelligence ; 11. Springer, Berlin [u.a.], 2. ed.. edition, 2006.
- [31] Theresa Mannschatz, T Wolf, and S Hülsmann. Nexus tools platform: Web-based comparison of modelling tools for analysis of water-soil-waste nexus. *Environmental Modelling & Software*, 76:137–153, 2016.
- [32] A. Meidan, J.A. García-García, M.J. Escalona, and I. Ramos. A survey on business processes management suites. *Computer Standards Interfaces*, 51:71 – 86, 2017.
- [33] OMG. Triple crown, 2013. <https://www.omg.org/intro/TripleCrown.pdf>.
- [34] OMG. Decision model and notation specification, December 2019. <https://www.omg.org/dmn/>.
- [35] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007.
- [36] Redhat. Redhat documentation. <https://access.redhat.com/documentation/>. [Accessed: 15-May-2022].
- [37] Kai Riemer, Justus Holler, and Marta Indulska. Collaborative process modelling-tool analysis and design implications. 2011.
- [38] Hans-Kristian Ringkjøb, Peter M Haugan, and Ida Marie Solbrekke. A review of modelling tools for energy and electricity systems with large shares of variable renewables. *Renewable and Sustainable Energy Reviews*, 96:440–459, 2018.
- [39] Signavio. Signavio dmn example. <https://documentation.signavio.com/suite/en-us/Content/process-manager/userguide/dmn/create.htm/>. [Accessed: 01-Aug-2020].

Bibliography

- [40] The JBoss Drools team. Drools expert user guide. .
- [41] Takuya Uemura, Shinji Kusumoto, and Katsuro Inoue. Function-point analysis using design specifications based on the unified modelling language. *Journal of software maintenance and evolution: Research and practice*, 13(4):223–243, 2001.

Acronyms

BPMN Business Process Modeling and Notation. iii, v, 1

CMMN Case Management Model and Notation. 1

DMN Decision Model and Notation. iii, v, 1

DRD Decision Requirements Diagram. 1, 3, 13, 14

DRL Drools Rule Language. 49

DSR Design Science Research. 2, 3, 17

RBS Rule-Based System. 3, 13, 45

SLR Systematic Literature Review. 3, 14

STR Systematic Tool Review. 2, 22

