

# *Patterns for Derivation Business Rules*

## Master Thesis

*E. Y. de Haan*

**Name** : Eline de Haan  
**Student number** : 4183134  
**Date** : July 13<sup>th</sup>, 2015  
**1<sup>st</sup> Supervisor** : Marco Spruit  
**2<sup>nd</sup> Supervisor** : Floris Bex  
**Daily Supervisor** : Martijn Zoet  
**Company Supervisor** : Peter Straatsma



## General Information

**INSTITUTION** : **UTRECHT UNIVERSITY**  
**NAME** : Eline Yvette de Haan  
**STUDENT NUMBER** : 4183134  
**CONTACT** : eline\_de\_haan@hotmail.com  
**TITLE THESIS** : Patterns for Derivation Business Rules  
**RESEARCH PERIOD** : November 10, 2014 – July 13, 2015  
**DATE OF DEFENSE** : July 13, 2015

**INSTITUTION** : **UTRECHT UNIVERSITY**  
**DEPARTMENT** : Information and Computing Sciences  
**MASTER** : Business Informatics  
**SUPERVISORS** : Dr. M.R. Spruit  
Dr. F. J. Bex

**INSTITUTION** : **ZUYD UNIVERSITY OF APPLIED SCIENCES**  
**UNIVERSITY OF APPLIED SCIENCES UTRECHT**  
**DEPARTMENT** : Commercial and Financial Management  
**SUPERVISOR** : Dr. M. Zoet

**HOST INSTITUTION** : **DUTCH TAX AND CUSTOMS ADMINISTRATION**  
**DEPARTMENT** : B/CAO Service Control  
**SUPERVISOR** : P. Straatsma



## **Preface & Acknowledgements**

Presented here is the thesis resulting from my graduation project for my master's degree in Business Informatics at Utrecht University. The research project has been performed at the Dutch Tax and Customs Administration in cooperation with Utrecht University and the University of Applied Sciences Utrecht. During the last eight months, this challenging research assignment allowed me to expand my knowledge in the domains of Business Rules Management (BRM) and Controlled Natural Languages (CNLs). Hopefully, this document could be of value for people interested in exploring these research domains.

At first, I would like to thank everyone who participated in this research project and had a share in its successful completion. In particular, I would like to thank my first supervisor Marco Spruit for his guidance during the entire project and providing me with his insights and feedback. Thanks to my second supervisor, Floris Bex, for reviewing and providing valuable comments in order to enhance the quality of my work.

Furthermore, my appreciation goes out to my daily supervisor Martijn Zoet for the time and effort he put into the guidance of this project. His guidance and expertise had a significant contribution to the success of this research. Moreover, I am grateful to my company supervisor Peter Straatsma for providing the opportunity, feedback and support during this project.

Besides my supervisors, I also consulted three specialists of the Dutch Tax and Customs Administration throughout this research. Therefore, I wish to express my gratitude to Diederik Dulfer, Gertrude Sangers–van Cappellen, and Frans Fokkenrood for reviewing and taking their time to share their knowledge with me.

Finally, I would like to thank my family and friends for their enthusiastic encouragement, support and understanding throughout this process.

## Abstract

In the last decade, derivation business rules have become an increasingly valuable asset for organizations. Derivation business rules are “expressions that evaluate facts, by means of a calculation or classification, leading to a new fact”. To specify and manage these business rules, a multitude of business rule languages and systems is available. The abundance of available systems and languages, and the fact that they differ to a large extent regarding their expressive power, causes two problems. The first problem organizations may encounter are difficulties in selecting an appropriate business rules management system or business rule language, since no set of criteria exists which could be used as reference point for comparison. This may for instance lead to the selection of a language with a too extensive or too low level of expressive power. A second problem can occur when a language, tailored to a particular business rules management system, is selected. In case an organization transfers to a new or additional system, the business rules have to be re-specified which is highly inefficient, expensive and error prone. The two identified problems resulted in the formulation of the following problem statements:

- “How can the problem be addressed that no tailored set of formal requirements exist, which can be used to verify if a business rule language is able to formulate derivation rules?”
- “How can the problem be addressed that business rules need to be re-modeled to comply with a new implementation dependent language?”

In order to tackle the outlined problems above, research was conducted based on the following research question: “How can derivation business rules be specified precisely and implementation independent?”

The answer to the main research question is: by using a *controlled natural language (CNL)*. A CNL is a specific notation form that can comply with a high level of precision, without being restricted to be applied by solely one automated information system. An additional benefit is that a CNL can resemble a natural language to a certain extent which enables humans to specify and verify the business rules. During this research, a CNL is created especially focused on specifying derivation business rules.

The CNL incorporates 15 fundamental constructs (i.e. building blocks of the language), which are required to compose a precise derivation business rule: 1) Conclusion part, 2) Condition part, 3) Modal Claim Type, 4) Construct, 5) Connective, 6) Expression, 7) Subject, 8) Quantifier, 9) Relation, 10) Ground, 11) Classification, 12) Propositional Operator, 13) Value, 14) Mathematical Operator, and 15) Mathematical Function. To enforce the syntax of the fundamental constructs, an underlying formal grammar is created. This formal grammar includes 40 grammar rules. In addition, a set of 19 different patterns is devised to restrict the CNL even more. The created artifacts have been validated in four rounds. During the first three rounds, the fundamental constructs and grammar rules have been validated by means of: 37 business rule patterns, 150 business rules, and the implemented business rules and components of six business rules management systems. The validation revealed that no fundamental constructs lacked or were superfluous. In the fourth validation round, a data set derived from the Dutch Tax and Customs Administration is specified by means of the patterns to validate the completeness of the pattern catalogue. The validation revealed that each of the 45 business rules in the data set could be specified with patterns from the pattern catalogue.

The resulting artifacts are seen as foundation for prolongation of this research. Possible future research could include the refinement of the developed CNL and pattern catalogue. Moreover, the applicability of the artifacts for other types of business rules besides derivation business rules could be investigated.



# Table of Contents

General Information .....	2
Preface & Acknowledgements .....	3
Abstract .....	4
1 Introduction.....	7
1.1 Research triggers and problem definition.....	8
1.1.1 Scientific Triggers.....	8
1.1.2 Practical Triggers .....	11
1.1.3 Business Triggers .....	11
1.1.4 Conclusion .....	12
1.2 Research Question .....	12
1.3 Research Method .....	13
1.4 Research Model.....	15
2 Literature Review .....	19
2.1 Business Rules Management and Business Rules .....	19
2.2 Derivation Business Rules.....	22
2.3 Controlled Natural Language (CNL).....	26
2.4 Formal Grammar .....	33
2.5 Patterns and pattern catalogues .....	35
2.6 Summary literature review.....	39
3 CNL Creation.....	41
3.1 Equivalent Underlying Fundamental Constructs.....	43
3.2 Unique Fundamental Constructs Conclusion Part .....	53
3.3 Unique Fundamental Constructs Condition Part .....	54
3.4 Summary.....	56
4 Preliminary Validation .....	58
4.1 Data Collection .....	58
4.1.1 Validation Round 1 – Pattern Level View .....	58
4.1.2 Validation Round 2 – Instance Level View.....	59
4.1.3 Validation Round 3 – Implementation Dependent Level View .....	60
4.2 Data Analysis .....	61
4.2.1 Overall Data Analysis Process and Method.....	61
4.2.2 Results Validation Round 1 – Pattern Level View.....	62
4.2.3 <i>Results Validation round 2 – Instance Level View</i> .....	70
4.2.4 <i>Results Validation round 3 – Implementation Dependent Level View</i> .....	77



5	Pattern Catalogue Creation .....	83
5.1	Subdivision Patterns .....	83
5.2	Modeling Choices .....	85
6	Pattern Validation .....	90
6.1	Data Collection .....	90
6.1.1	<i>Validation Round – Instance Level View</i> .....	90
6.2	Data Analysis .....	91
6.2.1	<i>Results Validation Round – Instance Level View</i> .....	91
7	Discussion .....	94
8	Conclusion .....	96
	References .....	98
	Appendix 1: Business Rule Classification Schemes .....	103
	Appendix 2: Detailed Explanation Pens .....	109
	Appendix 3: Formal Grammar .....	112
	Appendix 4: Pattern Catalogue .....	115
	Appendix 5: Validation Round 1 – Pattern Level View .....	122
	Appendix 6: Validation Round 2 – Instance Level View .....	123
	Appendix 7: Validation Round 3 – Implementation Dependent View .....	124
	Appendix 8: Decomposed Case Study Business Rule Set .....	125
	Appendix 9: Validation Patterns – Instance Level View .....	136

# 1 Introduction

In the 1960s, software was only composed of source code (Van der Aalst, 1996). However, it turned out that in this way the software was not agile enough to comply with the rapidly changing business environment (Pesic & van der Aalst, 2006). To address this issue, a shift in the information technology domain came up which is called the separation of concerns (Dijkstra, 1982). The term separation of concerns, within the information technology discipline, can be seen as a best practice or design principle to design information technology architectures (Van der Aalst, 1996; Versendaal, 1991). This principle implies that systems are divided into distinct sections which can be adjusted separately. By applying this principle, the complexity can be reduced and the comprehensibility can be enhanced (Dijkstra, 1982).

During the 1970s, the first layer was separated by means of introducing databases to segregate the data from the business logic. Ten years later, in the 80s, the user interface became an individual concern. This made it possible to adjust the interface without affecting the business logic or changing the data. During the 90s, also the process-layer followed and was put down as separate concern. Currently, in the last ten to fifteen years, more and more organizations regard their business logic as a separate concern. Business logic is nowadays captured in the form of business rules and decision models. A business rule is defined as: *“a statement that defines or constrains some aspect of the business, intending to assert business structure or to control the behavior of the business”* (Hay & Healy, 2000). Therefore, business rules play a crucial role in an organization’s daily operations. For example, business rules are used to: diagnose illness of patients, determine the amount of tax a citizen has to pay, determine eligibility, and restrict the order in which activities can execute (Hay & Healy, 2000; Liao, 2004; Von Halle, 2001).

When business rules were hard-coded, the ability to change their implementation in systems was mainly an IT department concern. To realize the actual change, developers and programmers were required. By separating business rules from the source code, they become more tangible and can be considered as individual objects which can be managed separately (e.g. by business people). This shift resulted into the development of a variety of languages to express business rules. For instance: RuleSpeak, The Decision Model (TDM), the Simple Rule Markup Language (SRML), the Semantic Web Rules Language (SWRL), the Production Rule Representation (PRR), the Semantics of Business Vocabulary and Business Rules (SBVR), SRL, N3, and IRL (Zur Muehlen & Indulska, 2010). The proliferation of business rule languages can be explained by the fact that these languages differ with regard to their philosophy, semantics and maturity (Zur Muehlen & Indulska, 2010).

Since various differences between these languages exist, research has been initiated to compare the business rule languages. Examples of such studies are Zoet, Ravesteyn, and Versendaal (2011) and Zur Muehlen and Indulska (2010). Zur Muehlen and Indulska compared the representational capabilities of four different business rule languages, by mapping the fundamental elements of these languages onto the constructs of the Bunge-Wand-Weber (BWW) representation theory. The BWW representation model allows to analyze the degree to which a modeling language is capable of representing elements of the real world (Wand & Weber, 1993).

However, previous studies focused on high-level elements (e.g. thing, property) of business rule languages. This view is applicable to analyze business rule languages at a global level, but not to evaluate the details of the syntax and semantics of the languages. During this research, the aim is to evaluate business rule languages from a more detailed and practical view.

## 1.1 Research triggers and problem definition

This section gives a description of the scientific, practical and business triggers of this research. The triggers are summarized into a conclusion at the end of this section.

### 1.1.1 Scientific Triggers

To get grip on the business rules, organizations can apply Business Rules Management (BRM) which is considered as the discipline comprising the representation, organizational structure, techniques, methods and tools to manage business rules (Von Halle, 2001; Zoet, 2014; Zur Muehlen & Indulska, 2010). Closely related to the BRM domain is Business Process Management (BPM), which provides the tools, methods, and languages to support organizations for managing and modeling their business processes (Van der Aalst, Ter Hofstede, & Weske, 2003; Weske, 2007). With regard to business process modeling, an abundance of languages is developed and available across the domain such as: Workflow Nets, Business Process Modeling Notation (BPMN), Business Process Execution Language (BPEL), Event-driven Process Chains (EPC), Petri Nets, and Graph based Workflow Language (Recker, Indulska, Rosemann, & Green, 2005; Weske, 2007; Wohed, Van der Aalst, Dumas, Ter Hofstede, & Russell, 2006). It even reached the point where a process modeling language was created entitled 'Yet Another Workflow Language' abbreviated as YAWL (Van der Aalst & Ter Hofstede, 2005). This underpins the problem of the proliferation of modeling languages as identified by Van der Aalst, Ter Hofstede, Kiepuszewski, and Barros (2003). Van der Aalst et al. (2003) stated that the creation of new languages and dialects is caused by different philosophies of such languages and applying different semantics. For example, some languages are able to model and execute multiple instances of the same activity while others can only loop once. Moreover, some languages are able to directly execute while others need to be transformed first. To shed some light on this problem and to be able to compare the different business process modeling languages, Van der Aalst et al. (2003) devised a set of design patterns. This pattern catalogue is now applied to model business processes and to make in-depth comparisons between languages and business process management systems. Furthermore, these design patterns are formulated with an implementation independent language which ensures that the applicability is not influenced by choice of technology or modeling language. An implementation independent language is considered as: a language that complies with a certain level of naturalness but has a delimited predefined expressiveness and is not tailored to be applicable for a specific automated information system (Zoet & Versendaal, 2013). In contrast, an implementation dependent language is a language that complies to a specific software formalism, has a delimited predefined expressiveness, and is tailored to be interpreted by a particular information system (Zoet & Versendaal, 2013).

In recent years, some attempts have already been made in the BRM field to establish pattern catalogues as well (Caron, Vanthienen, & Baesens, 2013; do Prado Leite & Leonardi, 1998; Hoppenbrouwers, 2011; Morgan, 2002; Von Halle, 2001; Wan-Kadir & Loucopoulos, 2004). All these pattern catalogues are written in an implementation independent language. However, only the one of Caron et al. (2013) adheres to a formalism with a high precision level since it is specified with first order logic (Kuhn, 2013). When a business rule set is specified with this *precise* implementation independent pattern catalogue, it is possible to automatically transform it into several different implementation dependent business rule sets. This transformation can be performed by means of a parser, a software component, which is able to analyze constructs of a business rule and transform those with a set of transformation rules into a specific grammar that complies with the execution language (see Figure 1.1). Due to the fact that the other catalogues are less precise, this automatic transformation is not possible. So, a major advantage of using a *precise* implementation independent language to specify (the patterns for) business rules is that the business rule set only has to be specified once instead of specifying it for each specific system and thereby reducing deployment time (see Figure 1.1).



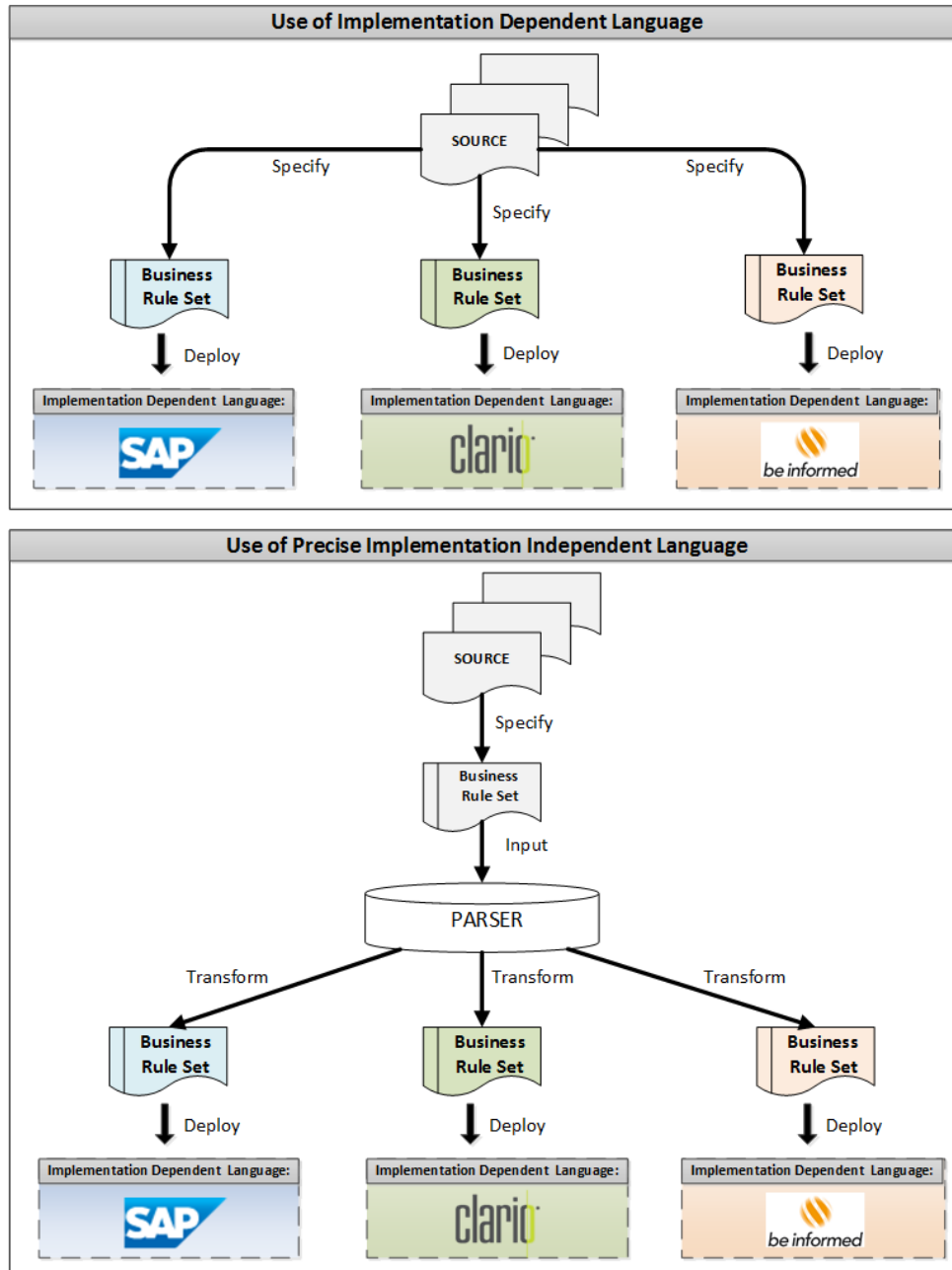


Figure 1.1: Implementation Dependent vs. Implementation Independent Business Rule Language

Another aspect that distinguishes the pattern catalogue of Caron et al. (2013) from the others is the purpose for which it is established. The patterns from Caron et al. (2013) are especially focused on the specification of one type of business rule, namely business rules that guide or constrain business processes (i.e. process rules). An example of a business rule pattern of Caron et al. (2013) is: *“If an activity of type a1 is performed then an activity of type activity a2 must be performed”*. In contrast, the other business rule catalogues focus on a variety of business rules types like: computation rules, guidelines, mandatory constraint rules, process rules, inference rules etc.

In summary, from literature solely one business rule pattern catalogue, the of Caron et al. (2013), emerges which complies with the following characteristics 1) it is specified in an implementation independent language that is precise enough to enable automatic transformation, and 2) it is focused on one specific type of business rules. With these premises, the goal of this research is to create a pattern catalogue specified in a precise implementation independent language that is focused on a different type of business rule namely: **derivation business rules**. In short, derivation business rules correspond to calculation and classification business rules. More formally, derivation business rules can be defined as: “expressions that evaluate facts, by means of a **calculation** or **classification**, leading to a new fact (i.e. conclusion)” (Hay & Healy, 2000; Von Halle & Goldberg, 2009). To make the definition more clear, two examples of a derivation business rule are provided within the dashed border below:

- 1) Total order amount **is calculated as** the amount of sold units multiplied by the unit price, if the customer has no outstanding invoices.
- 2) The customer status **must be set to** preferred, if the price of the product is more than 50.

Caron et al. (2013) focus on patterns that constrain the order in which activities in a business process should be executed, see the arrow with ‘**process business rules**’ in Figure 1.2. In addition, the execution of individual activities should be constrained. In some activities, decisions are being made based on derivation business rules (see the rectangle with ‘**derivation business rules**’ in Figure 1.2). This research focusses on creating patterns for derivation business rules. So, the pattern catalogue resulting from this research can be considered as a complement to the pattern catalogue of Caron et al. (2013).

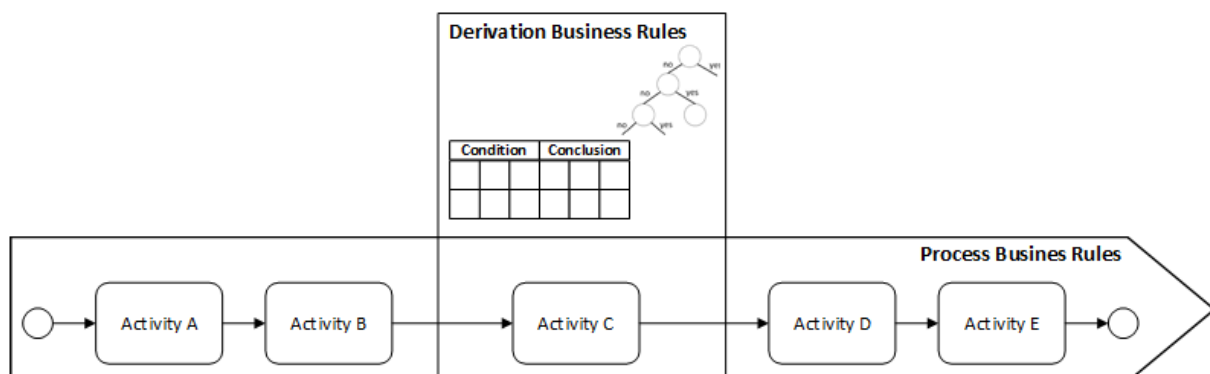


Figure 1.2: Intersection Patterns for Process Business Rules and Derivation Business Rules (adapted from (Zoet et al., 2011)

To achieve the overall goal, research has to be conducted to identify the **fundamental constructs** that are necessary to specify a derivation business rule in a precise implementation independent way. For this research, fundamental constructs are considered as: “the building blocks, i.e. sentence parts, i.e. rule clauses, of which a business rule is constructed” (Hay & Healy, 2000; Von Halle, 2001). Furthermore, the relations between these fundamental constructs (i.e. meta-model) and the occurring patterns for derivation business rules should be determined. By creating a pattern catalogue written in a precise implementation independent language specifically applicable for derivation business rules, this research has a scientific contribution by adding a new type of pattern catalogue to the scientific knowledge base. Furthermore, this research provides further insight into the fundamental constructs of derivation business rules.

### **1.1.2 Practical Triggers**

Business rules are applied for many different applications and software systems, such as: case-based reasoning systems, knowledge management systems, expert systems, business rule engines, and neural network systems (Liao, 2004). In previous years, many systems that apply business rules have been implemented (Nelson, Peterson, Rariden, & Sen, 2010). A few examples of specific business rule engine implementations are: 1) the “Immigratie- en Naturalisatiedienst” (IND) implemented the business rule engine Aquima, 2) “Dienst Regelingen” implemented the business rule engine Be Informed, and 3) the “Dutch Tax and Customs Administration” transferred multiple times to different business rule engines.

Above described examples of implementations show that organizations choose to switch occasionally to a new or additional software system to incorporate their business rules. Reasons to change can be explained by different external factors like the dynamic environment and increased legislation (Boyer & Mili, 2011; Graham, 2006). This transition or addition of a new business rule engine forces organizations to modify their business rules when an implementation dependent business rule language is applied to specify the business rules. Given the fact that the business rule set cannot be reused, it has to be specified again to comply with the language of the new business rule engine. As a result, the specification of business rules directly from a source into a specific implementation dependent language can in some cases be very inefficient, expensive and error prone. So, it is desirable for organizations to have an additional layer, in the form of an implementation independent business rule language, from which the business rules can be directly transformed to an implementation dependent language (see Figure 1.1). In addition, it is beneficial to facilitate the people that are responsible for specifying the business rules with guidelines to make the writing process less error prone, more consistent, and less time consuming. A commonly used mechanism for this purpose is a pattern catalogue (Kuhn, 2013).

As can be concluded from the previous paragraphs, implementation dependent business rule languages have a major drawback from a practical point of view. To avoid this drawback of repeatedly re-designing current business rule sets, which would obliterate current investments, a pattern catalogue specified in a precise implementation independent language seems crucial. However, the authors are not aware of the existence of such a pattern catalogue tailored to specify derivation business rules. This research aims at providing a practical solution for this obstacle.

### **1.1.3 Business Triggers**

The Dutch Tax and Customs Administration is currently developing an implementation independent language to specify the products and services they offer. The triggers for the Dutch Tax and Customs Administration to develop such a language are: 1) only having to specify the business rules once which can be deployed to multiple environments, 2) reducing the risk of errors, 3) decreasing the amount of time spent on transforming implementation independent business rules to implementation dependent business rules, and 4) decreasing the costs of transformation.

With regard to the further specification of the products and services, the Dutch Tax and Customs Administration distinguishes the following six different areas: 1) Interaction rules, 2) Classification structure and classification rules, 3) Calculation structure and calculation rules, 4) Data exchange rules, 5) Data business rules, and 6) Process business rules. Each of the areas needs to be further researched and specified. Due to time and resource constraints, it is impossible to focus on all these areas during this research. Therefore, this research will focus on two of the six domains namely ‘classification structure and classification rules’ and ‘calculation structure and calculation rules’. These two domains are chosen since the majority of the business rules of the Dutch Tax and Customs Administration comprises calculation and classification business rules. Both domains, calculation and classification rules, can be considered as: derivation business rules (Ghose & Koliadist, 2007; Park & Choi, 2004).

### 1.1.4 Conclusion

The triggers described above provide substantiation for the demand of creating a pattern catalogue specified in a precise implementation independent language. In addition, a practical and business need for adopting such a pattern catalogue can be identified as described in section 1.1.2 and 1.1.3. However, to the knowledge of the authors there currently does not exist an implementation independent business rule pattern catalogue that is tailored to derivation business rules. This poses a problem that this research will attempt to address. Taking the previous statements into account, the following two problem statements can be formulated:

- *How can the problem be addressed that no tailored set of formal requirements exist, which can be used to verify if a business rule language is able to formulate derivation rules?"*
- *"How can the problem be addressed that business rules need to be re-modeled to comply with a new implementation dependent language?"*

In the next section, a formal research question is formulated along with the related sub-questions.

## 1.2 Research Question

Based on the research triggers and problem definition as described in previous sections, the demand for a pattern catalogue specified in a precise implementation independent language to specify derivation business rules can be identified. Based on this demand, the following formal research question is established:

---

*"How can derivation business rules be specified precisely and implementation independent?"*

---

According to Kuhn (2013), the term 'precise' in the main research question equals to the fact that *"the language, in which the business rules will be specified, should be fully formal on a syntactic level; that is, they are (or can be) defined by a formal grammar. Business rules in such a language can be deterministically parsed to a formal logic representation."* In section 2.3, the term *precise* will be explained more extensively.

In order to answer the above provided research question, several sub-questions need be answered first. These sub-questions are used to structure the research and serve in this way as guidelines to address the main research question.

The following sub-questions are defined:

1. *"Which notation forms can be used to specify derivation business rules?"*
2. *"Which fundamental constructs are necessary to construct a precise derivation business rule?"*
3. *"Which grammar rules should be enforced on the fundamental constructs to specify precise derivation business rules?"*
4. *"Which patterns can be identified for specifying derivation business rules?"*

## 1.3 Research Method

This section describes the research method that will be applied to achieve the main goal of this research, namely the construction of a pattern catalogue specified in a precise implementation independent language. Given the fact that an artifact is created to address the outlined problem in section 1.1.4, it is appropriate and justified to define this research as a design-science research (March & Smith, 1995). The foundation of design-science research implies that an identified problem will be solved through research and the development of a new artifact, for example in the form of a model, construct, instantiation or method (March & Smith, 1995).

Hevner, March, Park, and Ram (2004) provide a method including a conceptual framework and guidelines in order to support the establishment and execution of a design-science research project. The framework is divided into three aspects (see Figure 1.3) and traverses an iterative cycle in order to solve a problem (Hevner et al., 2004):

1. **The Environment:**

This research was triggered by the environment, in response to the demand of the Dutch Tax and Customs Administration for a solution to their business need (problem). The relevance of performing this research can be assured when the issues, expectations, and requirements of the Dutch Tax and Customs Administration are identified and in this way the business need is substantiated;

2. **IS Research:**

The research design consists of two phases, the *Develop/Build* phase and the *Justify/Evaluate* phase, which will be traversed four times. During the *Develop/Build* phase, theory on Business Rules Management (BRM) will be established and as main artifact a pattern catalogue specified in an implementation independent language will be created. This main artifact will be assessed during the *Justify/Evaluate* phase by means of conducting experiments in four different rounds. After each validation round, the artifact will be refined in the *Develop/Build* phase again. The assessment and refinement process will be an iterative process. After completing this process, the developed artifact will be applied by the Dutch Tax and Customs Administration to measure the contribution of the research and it will be added to the Knowledge Base;

3. **The Knowledge Base:**

The Knowledge Base provides the required scientific literature and appropriate methods to perform this research. First of all, theory on the following aspects will be retrieved by means of performing a literature review: Business Rules Management (BRM), Controlled Natural Language (CNL), formal grammars, and design patterns. Furthermore, the validation criteria will be retrieved for conducting the experiments during the *Justify/Evaluate* phase. Research rigor can be obtained when these existing methods from the Knowledge Base are applied appropriately.

The specific research design framework, tailored to this research, as described above is depicted in Figure 1.3.

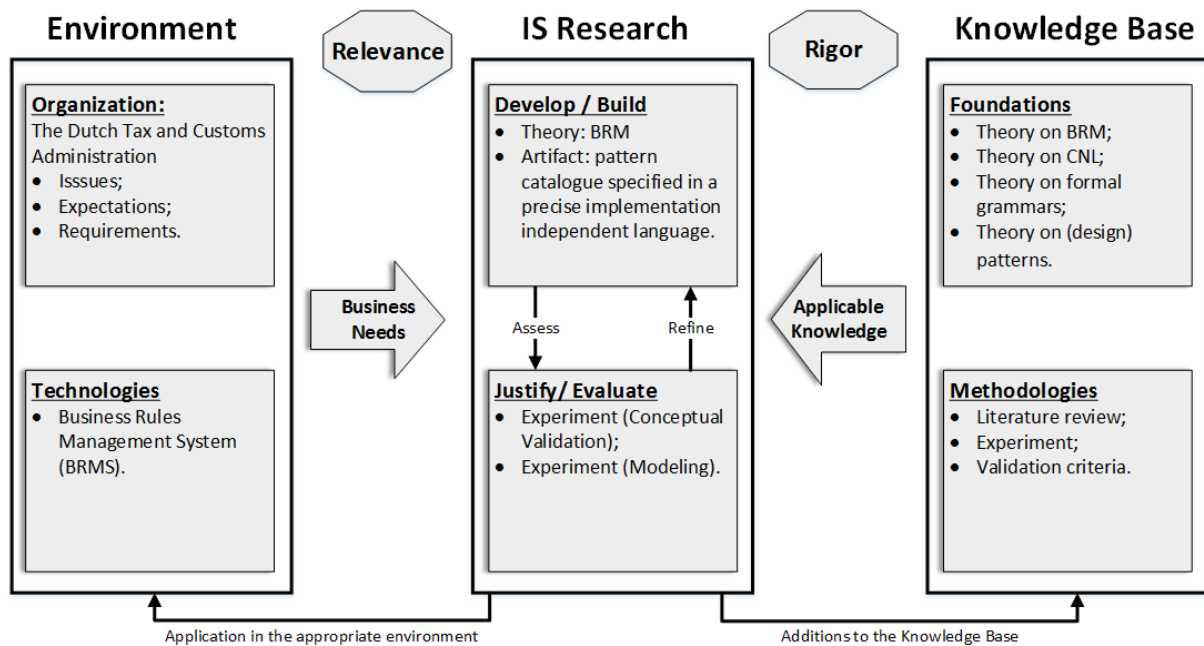


Figure 1.3: Research design framework tailored to this research (adapted from Hevner et al., 2004)

As mentioned previously, Hevner et al. (2004) also provide guidelines along with the framework. The seven guidelines for design research are as follows:

1. **Design as an artifact**  
A design-science research project should produce an artifact.
2. **Problem Relevance**  
Design-science research aims at solving business problems to be relevant for the environment.
3. **Design Evaluation**  
The utility, quality and efficacy of the artifacts resulting from a design-science research project should be demonstrated by applying well-executed evaluation methods.
4. **Research Contributions**  
A design-science research should contribute clear and verifiable knowledge to the knowledge base of the related scientific field.
5. **Research Rigor**  
A design-science research project should be conducted by using rigorous methods for both the construction as the evaluation of the created artifact.
6. **Design as a Search Process**  
Design-science research is a problem solving process which should enable iterative cycles to reach desired ends.
7. **Communication of Research**  
The results of a design-science research project should be communicated to researchers and practitioners.

The next section describes how these above listed guidelines are integrated for this research.

## 1.4 Research Model

The previous section presented the research method, this section describes the more detailed approach of the research by means of providing a research model. This research model is devised by incorporating research methods from Wieringa (2013) for scaling up to practice and by taking the seven guidelines of Hevner et al. (2004) into account. The establishment, compliance to the guidelines, and the explanation of the research model are described below.

The Dutch Tax and Customs Administration faces the problem of repeatedly re-designing current business rule sets when making a transition to or implementing a new business rule engine. This research aims at delivering a solution for this problem which would allow them to specify their business rules once and deploy them to multiple environments. By realizing this solution and solving a business problem, this research satisfies guideline 2 *Problem Relevance* of Hevner et al. (2004).

To comply with guideline 4 *Research Contribution*, Hevner et al. (2004) state that the contributions of the research should be clear and verifiable. Besides the positive effect of only having to specify the business rules once, three other positive effects (i.e. contributions) of the solution are anticipated: reducing risk of making errors, decreasing the amount of time spent on transforming implementation independent business rules to implementation dependent business rules, and decreasing the costs of transformation. However, these last three cannot be verified during this research due to time constraints. In order to verify these three contributions, an experiment has to be conducted during which 1) a set of business rules is specified manually in multiple implementation dependent languages after which the business rules are executed and 2) a set of business rules is specified once with the created pattern catalogue in the precise implementation independent language after which it is transformed into multiple implementation dependent languages and then are executed. Subsequently, both traversed scenarios have to be compared with respect to the amount of errors, the time spent and the corresponding amount of money. Such a time-consuming experiment is not feasible during this research. However, the first contribution will be used for verification. By means of this research, it will be demonstrated that it is possible to specify a business rule set once which then can be transformed into various implementation dependent languages. To be able to demonstrate this, the business rules specified with the devised patterns will be mapped onto different implementation dependent business rule languages (of business rules management systems). In this way, this research also meets guideline 4.

Each of the aforementioned positive effects or contributions are referred to as “effects by mechanisms” by Wieringa (2013); they are the effects that an artifact produces in terms of underlying mechanisms. So, to produce the effects by mechanisms and at the same time address the outlined problem, an artifact is needed (Wieringa, 2013). This artifact corresponds to the pattern catalogue specified in a precise implementation independent language which will be created during this research. By creating an artifact, this research also adheres to guideline 1 *Design an artifact* of Hevner et al. (2004).

According to Wieringa (2013), an artifact is related to its context thus if the context changes also the effects may change. He summarizes and expresses this in the following way: *[Artifact x Context] will produce Effects by Mechanisms*. Therefore, Wieringa (2013) emphasizes the importance of validating an artifact in different contexts to investigate in what kind of contexts what kind of effects are produced. The main goal of this validation is to see if an artifact will produce the desired effects when it will be transferred to the market and is applied in practice.

Wieringa (2013) describes how this validation can be performed by scaling up the context of an artifact to the conditions of practice until ‘street credibility’ is reached (see Figure 1.4). The validation process is divided into the following three phases: 1) conceptual validation, 2) modeling, and 3) field testing. The conceptual validation phase implies that the artifact is tested in a small artificial context. This phase is mainly performed on paper. During the modeling phase, the artifact is tested out in a more realistic context. In the last phase, field testing, the artifact is validated by actually applying it into practice on a large scale. By performing these three phases, the credibility of the artifact can be enhanced from lab credibility to street credibility. This validation and scaling up process is visualized in Figure 1.4, in which the axis represent the two ways of generalizing from lab credibility to street credibility. The horizontal-axis corresponds to inductive generalization which is “*the inference from a sample of test subjects to the population of subjects*”. The vertical-axis, on the other hand, corresponds to analogical generalization which entails “*the inference from models to real-life subjects*” (Wieringa, 2013).

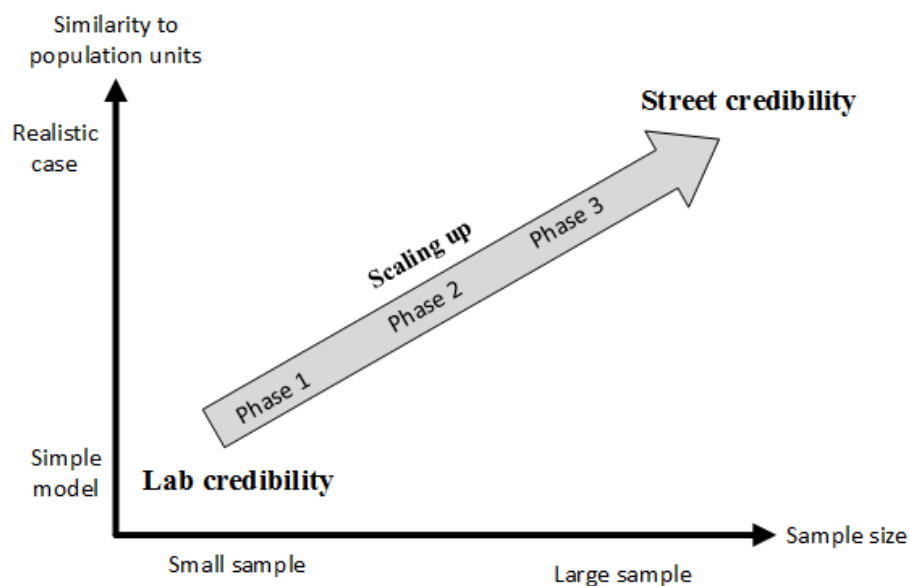


Figure 1.4: Scaling up to practice (adapted from Wieringa, 2013)

Performing the validation in phases provides compliance to guideline 6 of Hevner et al. (2004). Guideline 6, *Design as a Search Process*, entails that an iterative cycle is traversed until the end goal (effect by mechanism) with the artifact (a pattern catalogue specified in a precise implementation independent language) is achieved. An important aspect with regard to this validation is that it is performed to test the utility, quality and efficacy of the artifact (guideline 3 *Design evaluation*). A second important aspect regarding the validation that needs to be considered according to Hevner et al. (2004), is that the design and validation should be performed in a rigorous way (guideline 5 *Research rigor*). Validation is the so-called ‘triangle of evil’ (McGrath, 1981) see Figure 1.5: maximal measurement precision (**Max B**), maximal focus on realism of research context (**Max C**), and maximal focus on generalizability (**Max A**).



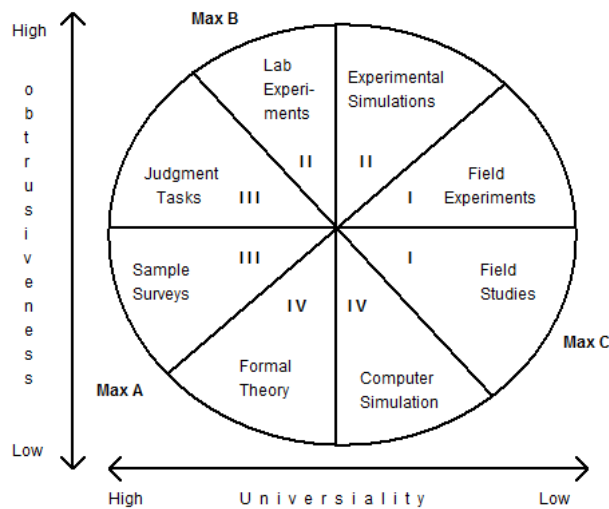


Figure 1.5: Triangle of evil (adapted from McGrath, 1981)

For this research, the focus lies on the first two aspects by conducting (lab) experiments on case study data. The main reason to conduct experiments in this research is because they provide the possibility to discover causal findings. As a result, experiments are often very strong in terms of internal validity which can lead to theoretical well substantiated results (Bryman & Bell, 2003). The case study data is used as input since it provides the possibility to investigate the problem within its real-life context, thus the problem can be viewed from several perspectives defined by its context. The advantage of this broad view is that new insights could be discovered, such as patterns, which initially would not be expected or sought for. Experiments, on the other hand, often isolate the problem from the natural

context and place it in a laboratory setting. This disadvantage of experiments can be eliminated by the use of the case study data (Blumberg, Cooper, & Schindler, 2011).

To ensure the relevance, rigorousness, and validity of the research and artifact, the above described guidelines and the insights of Wieringa (2013) are taken into account to create the research model. The research model is shown in Figure 1.6 on the next page, which is based on the theory on developing a research model by Verschuren and Doorewaard (2007). The research model contains the overall research process, depicted in different phases, and the deliverables of the research. In addition, the research model indicates the sections of the thesis in which the deliverables are presented. The first phase in the research model is the 'Literature Gathering' during which theory on the following five areas is obtained: Business Rules Management, Derivation Business Rules, Controlled Natural Languages, Formal grammars, and Patterns. During the second phase 'Literature Analysis', the obtained literature is analyzed to identify the fundamental constructs and grammar rules of the precise implementation independent language (i.e. controlled natural language). These artifacts are validated during the 'Preliminary Validation' phase by traversing three rounds, all from a different point of view. This third phase corresponds to the 'Conceptual Validation phase' of the research of Wieringa (2013). In the fourth phase '(Re)-design and development', the precise implementation independent language (i.e. controlled natural language) including fundamental constructs and grammar rules are revised by means of the validation results. Furthermore, the pattern catalogue specified in the precise implementation independent language (i.e. controlled natural language) is developed during this phase based on the revised artifacts. After that, the 'Validation' phase is performed which corresponds to the 'Modeling phase' of Wieringa (2013). In this fifth phase, the devised pattern catalogue is validated by means of real-life case study data from the Dutch Tax and Customs Administration. Only the last phase of Wieringa (2013) is not incorporated, since field testing requires that the artifact will be tested by applying it in practice. This phase is not included in this research due to time constraints. Subsequently, the 'Refinement' phase is initiated which entails the revision of the pattern catalogue based on the outcome of the validation in the preceding phase. The last phase of the research model 'Communication' addresses guideline 7, *Communication of research*, of Hevner et al. (2004). This guideline indicates that it is important to distribute the research findings, which is achieved by writing and presenting the artifacts, a thesis and research paper.

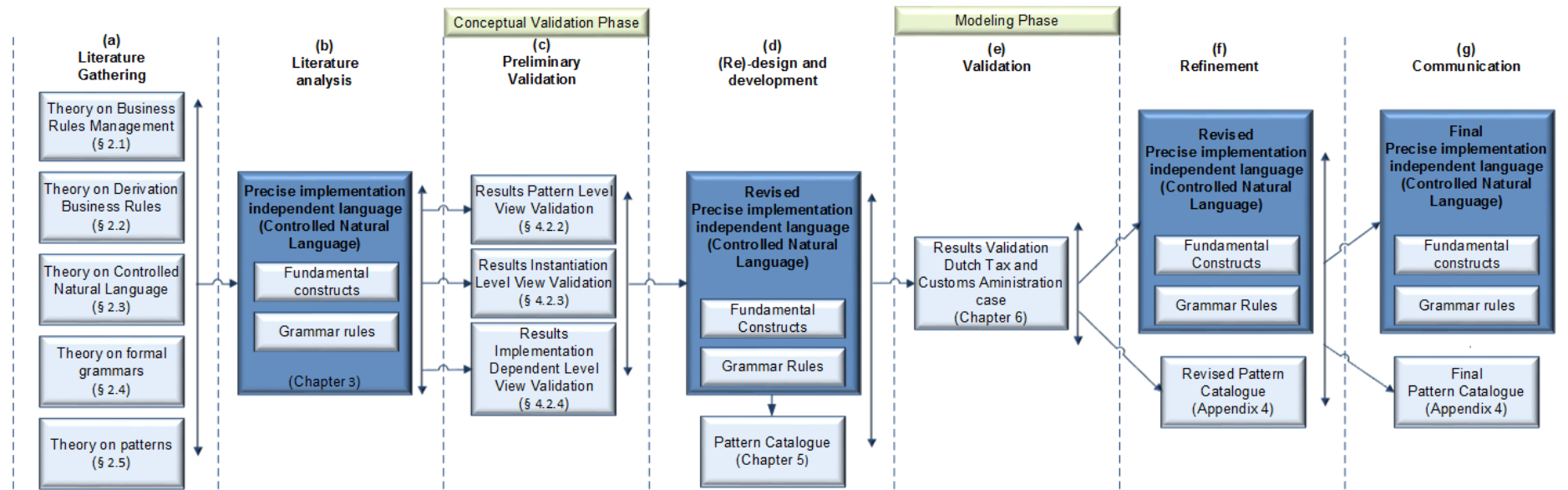


Figure 1.6: Research model

## 2 Literature Review

Previous sections presented the triggers and problems to perform this research. Furthermore, the research questions were listed and the research approach is discussed. In the next five sections, the individual concepts related to the research problem are elaborated on. To ground the literature review, the full conceptual overview of the concepts and their relationships is already presented in Figure 2.1. Each rectangle contains a number that represents the section of this thesis in which the mentioned concept will be discussed.

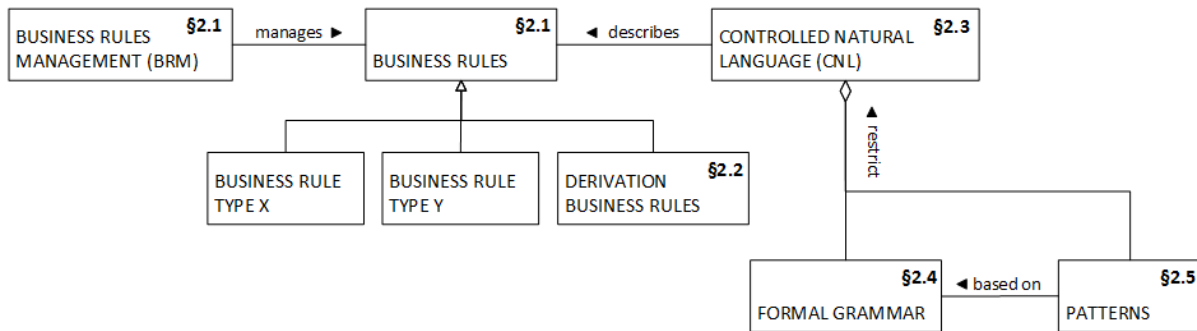


Figure 2.1: Full Conceptual Overview

### 2.1 Business Rules Management and Business Rules

In this section the concepts Business Rules Management (BRM) and business rules will be explained (see blue colored rectangles in Figure 2.2). With regard to these two concepts, their position within literature is shown and also their definitions are provided.

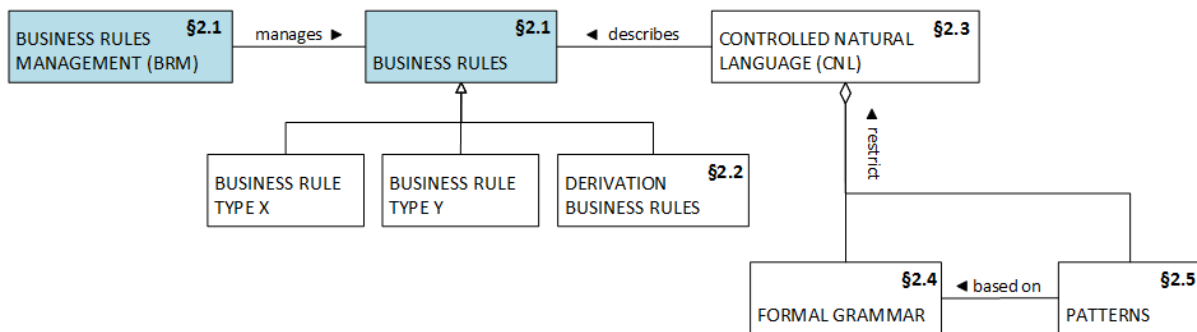


Figure 2.2: Conceptual Overview - BRM and Business rules

Organizations have to deal with a lot of rules that can be either established by the organization itself or imposed by the external environment (e.g. by legislation). These rules exist to guide or constrain organizations' information system including their business operations, human actors (employees) and information technology (Boyer & Mili, 2011). Rules can be viewed from different perspectives, on a high abstraction level two main perspectives can be distinguished: the business perspective and the information system perspective (Hay & Healy, 2000). From a business perspective, a rule provides guidance regarding human actors for instance in the form of conduct or procedures (Hay & Healy, 2000). With regard to the information system perspective, rules are focused on the behavior and the structure of the business and the data that is captured by information systems during the performance of business processes (Boyer & Mili, 2011). Rules that provide guidance for either of the above mentioned perspectives can be considered as 'business rules' (Boyer & Mili, 2011; Herbst, 1996). This view is also supported by the Object Management Group (OMG) which defines a rule as "a proposition

that is a claim of obligation or of necessity” and a business rule as “a rule that is under business jurisdiction” (Object Management Group, 2013).

In literature, a “business rule” is defined in a variety of ways which is emphasized by a statement of Von Halle (1994) “depending on whom you ask, business rules may encompass some or all relationship verbs, mathematical calculations, inference rules, step-by-step instructions, database constraints, business goals and policies, and business definitions”. To illustrate these different perspectives, several definitions will be discussed below.

From the perspective of Von Halle (2001), business rules are “the set of conditions that govern a business event so that it occurs in a way that is acceptable to the business”. In contrast, Kramer (1997) proposed the following definition to describe business rules: “programmatically implemented policies and practices of a business organization”. According to Ceri and Fraternal (1997): “business rules respond to application needs; they model the reaction to events which occur in the real world, with tangible side effects on the database content, so as to encapsulate the application’s reactive behavior to such events”. Selfridge, Waters, and Chikofsky (1993) describe a business rule as: “a requirement on the conditions or manipulation of data expressed in terms of the business enterprise or application domain”. A more general definition is found in Ross (1987), defining a business rule as “a rule or policy that governs the behavior of the enterprise and distinguishes it from others”. Moreover, Herbst (1997) defines business rules as follows: “statements about how the business is done, i.e. about guidelines and restrictions with respect to states and processes in an organization”. The definition of Rosca, Greenspan, Febowitz, and Wild (1997) resembles the previous one: “business rules are statements about the enterprise’s way of doing business. They reflect policies, procedures or other constraints on ways to satisfy customers, make good use of resources”.

A more detailed definition of a business rule is the one of Morgan (2002), who defines a business rule as “a compact statement about an aspect of a business [that] can be expressed in terms that can be directly related to the business, using simple, unambiguous language that’s accessible to all interested parties: business owner, business analyst, technical architect, and so on. It’s a constraint, in the sense that a business rule lays down what must or must not be the case”. A definition which is frequently cited is the one originated from the GUIDE project: “A statement that defines or constrains some aspect of the business, intending to assert business structure or to control the behavior of the business” (Hay & Healy, 2000). This latter definition will be adopted for this research given the fact that it is used very often in literature (Morgan, 2002).

All above provided definitions are listed in Table 2.1 below to give an overall overview.

Source	Definition Business Rule
OMG (2013)	“a rule that is under business jurisdiction”
Von Halle (2002)	“the set of conditions that govern a business event so that it occurs in a way that is acceptable to the business”
Kramer (1997)	“programmatically implemented policies and practices of a business organization”
Ceri and Fraternal (1997)	“business rules respond to application needs; they model the reaction to events which occur in the real world, with tangible side effects on the database content, so as to encapsulate the application’s reactive behavior to such events”

Selfridge, Waters and Chikofski (1993)	"a requirement on the conditions or manipulation of data expressed in terms of the business enterprise or application domain"
R. G. Ross (1987)	"a rule or policy that governs the behavior of the enterprise and distinguishes it from others"
Herbst (1997)	"statements about how the business is done, i.e. about guidelines and restrictions with respect to states and processes in an organization"
Rosca et al. (1997)	"business rules are statements about the enterprise's way of doing business. They reflect policies, procedures or other constraints on ways to satisfy customers, make good use of resources"
Morgan (2002)	"a compact statement about an aspect of a business [that] can be expressed in terms that can be directly related to the business, using simple, unambiguous language that's accessible to all interested parties: business owner, business analyst, technical architect, and so on. It's a constraint, in the sense that a business rule lays down what must or must not be the case"
Hay & Healy (1997)	"A statement that defines or constrains some aspect of the business, intending to assert business structure or to control the behavior of the business"

**Table 2.1:** Definitions of a business rule

From literature it emerges that organizations have different issues with managing their business rules. These issues are mostly caused by the way in which the business rules are implemented, namely decentralized and embedded in the information systems (Boyer & Mili, 2011):

1. A first issue regards *consistency*; many organizations do not have insight into which business rules are deployed for which business service. As a result, organizations often deploy different business rule sets with the possibility of operating under conflicting business rules.
2. A second issue concerns *traceability*; organizations should be able to show which business rules have been applied at which moment in time. This traceability, or in other words transparency, is important to provide justification of the followed procedures and taken decisions towards the stakeholders.
3. A third issue is related to *agility*; organizations should be able to quickly respond to the changing business environment. These changes also influence the business rules and their current implementation, it is mostly hard to find and change the implemented business rules. An organization has to cope with this impact in an agile way.

The three above described issues can be addressed by Business Rules Management (BRM), which can be seen as the discipline comprising the representation, organizational structure, techniques, methods and tools to manage business rules (Von Halle, 2001; Zoet, 2014; Zur Muehlen & Indulska, 2010). The application of BRM can provide several benefits: 1) improving the alignment between information systems and the business, 2) enhancing the transparency of the business operations, and 3) increasing the business agility (Boyer & Mili, 2011).

## 2.2 Derivation Business Rules

In previous section, the concepts BRM and business rules are explained. This section deals with the different types of business rules that exist. Firstly, a general view about the various business rule types will be discussed. Subsequently, the specific business rule type at which this research is targeted will be described namely *derivation business rules* (see Figure 2.3).

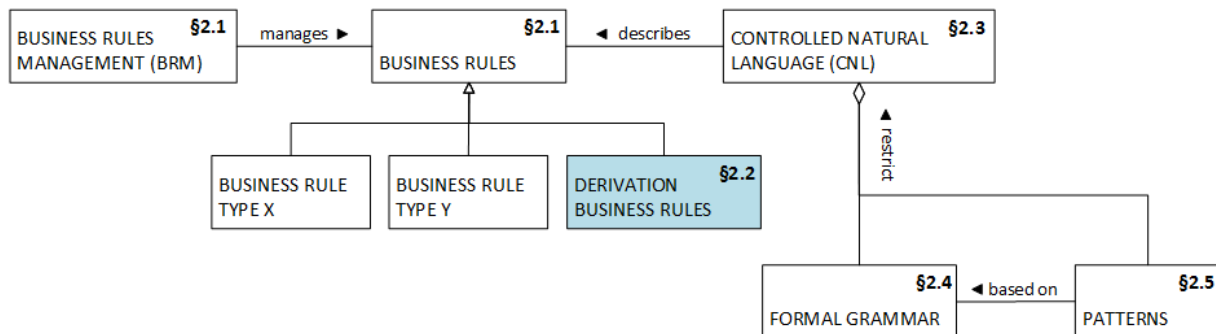


Figure 2.3: Conceptual Overview - Derivation Business Rules

Currently, not one commonly accepted way to classify business rules exists. A frequently used distinction in literature is the separation of business rules into two main types: structural (definitional) and behavioral (operational) business rules. Where structural business rules define some aspect of the structure of the organization, in other words they define the relationships between entities (objects) in a business information model (Boyer & Mili, 2011). An example of a structural business rule is (Object Management Group, 2008): “*Each rental always has exactly one requested car group*” which refers to the Rental entity. In contrast, behavioral business rules are evaluated (for example by a rule engine) to determine a decision result; they implement business decision logic (Boyer & Mili, 2011). An example of a behavioral business rule is (Object Management Group, 2008): “*The rental duration of a rental must be considered as expired, if the rental duration is more than 90 rental days*”.

To delimit this research, the focus will lie on one specific type of behavioral (operational) business rules namely **derivation business rules**. A derivation business rule can be defined as: “*an expression that evaluates facts, by means of a calculation or classification, leading to a new fact (i.e. conclusion)*” (Hay & Healy, 2000; Von Halle & Goldberg, 2009). Looking at this definition into more detail, facts can be divided into two types: base facts and derived facts. A base fact is defined as “*a fact given in the world that is stored in an information system*” (Hay & Healy, 2000). A derived fact is specified as “*a fact that is created by a mathematical calculation or an inference from other facts*” (Hay & Healy, 2000). In this case, an inference corresponds to a classification, as included in the definition, which produces a derived fact by means of reasoning about premises (i.e. arguments) to reach a conclusion.

From literature, ten different classification schemes to classify business rules emerged which each cover several business rule categories (types) (Boyer & Mili, 2011; Caron et al., 2013; do Prado Leite & Leonardi, 1998; Hay & Healy, 2000; Object Management Group, 2008, 2013; Sangere-van Cappellen, 2014; Von Halle, 2001; Wan-Kadir & Loucopoulos, 2004; Zoet, 2014). In Appendix 1, the found classification schemes are listed along with an explanation of each category and source to provide more insight. Among the ten classification schemes, different names are used to refer to either similar or dissimilar business rule categories. To position the type of business rule on which this research focuses, *derivation business rules*, this type is compared to the categories included in the ten found classification schemes.

This comparison showed that derivation business rules correspond to the following twelve categories of the found classification schemes:

1. Inference rules - from the classification scheme of Boyer and Mili (2011)
2. Computation rules - from the classification scheme of Boyer and Mili (2011)
3. Computation rules - from the classification scheme of Von Halle (2001)
4. Inference rules - from the classification scheme of Von Halle (2001)
5. Derivation rules - from the classification scheme of Hay and Healy (2000)
6. Computation rules - from the classification scheme of Wan-Kadir and Loucopoulos (2004)
7. Inference rules - from the classification scheme of Wan-Kadir and Loucopoulos (2004)
8. Classification rules - from the classification scheme of Morgan (2002)
9. Computation rules - from the classification scheme of Morgan (2002)
10. Decision rules - from the classification scheme of Sangers-van Cappellen (2014)
11. Calculation rules - from the classification scheme of Sangers-van Cappellen (2014)
12. Rounding rules - from the classification scheme of Sangers-van Cappellen (2014)

To demonstrate the similarity between the twelve categories and a derivation business rule, Table 2.2 lists seven example business rules from the above mentioned sources. The left column shows the name of the business rule category along with the authors from which the category originates. In the right column, an example of each category adapted from the authors is provided. The similarity between these examples and a derivation business rule is indicated by denoting the following aspects from the adopted definition of a derivation business rule as follows:

1. a **calculation** or **classification**;
2. the **new fact**.

Business Rule Category (source)		Example Business rule
1	Inference (Boyer & Mili, 2011)	If the age of the driving license <b>is below 3</b> , then <b>add a risk factor of 50 to <u>the total risk score</u></b> .
2	Computation (Boyer & Mili, 2011)	<b><u>A risk factor variable</u> can be computed as the possession time of driving license minus the number of years without claims.</b>
3	Computation (Von Halle, 2001)	<b><u>The total-amount-due for an order</u> is computed as the sum of the line-item amount(s) for the order plus tax.</b>
4	Inference (Von Halle, 2001)	If a customer <b>has no outstanding invoices</b> , then <b><u>the customer</u> is of preferred status.</b>
5	Derivation (Hay & Healy, 2000)	<b><u>The insurance amount in Rental</u> is calculated from the rental insurance rate multiplied by its number of days.</b>
6	Computation (Wan-Kadir & Loucopoulos, 2004)	<b><u>The amount of bill item</u> is computed as the unit amount multiplied by the quantity.</b>
7	Inference (Wan-Kadir & Loucopoulos, 2004)	If a patient's condition <b>is critical</b> then <b><u>the patient</u> is an emergency patient.</b>

Table 2.2: Example business rules per category

The overall aim of executing the example business rules in Table 2.2 is the creation of new information (i.e. new facts). In case of the example business rules provided above, the first business rule creates new information by calculating the value of subject “total risk score” when the condition is met. The second, third, fifth, and sixth example also create new information by means of a calculation but without evaluating a condition. Examples four and seven fill in a specific value for the subjects “customer” and “patient”.

The aforementioned aim is also reflected in the classification scheme of Von Halle (2001), which classifies business rules by means of their intention (see Figure 2.4). This classification scheme is very applicable to show what derivation business rules comprise and which business rules can be classified as such and which not. Therefore, this classification scheme is adopted for this research. In Figure 2.4, the grey colored ellipses are the types that correspond to derivation business rules on which the focus lies. According to Von Halle (2001), “the conclusion for an inference is a new piece of information and the conclusion for a computation is a computed value”.

Both the ‘Mandatory Constraint’ and ‘Guideline’ category of Von Halle (2001) do not correspond to derivation business rules. Although both categories also evaluate facts like derivation business rules, the aim of the evaluation is different. Mandatory Constraints and guidelines evaluate these facts to constrain subjects by including a condition that must be true or not. Derivation business rules, on the other hand, evaluate facts to create new information (a new fact) as denoted with light blue in the examples in Table 2.2.

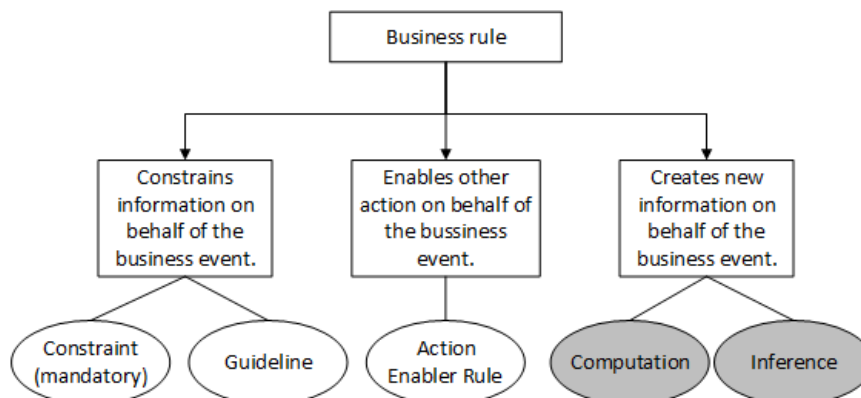


Figure 2.4: Rule Classification (adapted from Von Halle, 2001)

Mandatory Constrains differ from Guidelines in the way a business rule is imposed. A Mandatory Constraint prescribes that a condition must or must not be met, and a Guideline only suggests the compliance to a condition. An example of a Mandatory Constraint is: “The total dollar amount of a customer order must not be greater than the customer’s single order credit limit amount” (Von Halle, 2001). This example shows that the business rule only checks if a subject (*total dollar amount*) is not greater than another subject (*customer’s single order credit limit amount*). However, no new information or fact is calculated or derived. The following rule categories, found in literature, are similar to the Mandatory Constraint category and are therefore also not applicable for this research:

- Non-functional rules (do Prado Leite & Leonardi, 1998);
- Mandatory constraint (Wan-Kadir & Loucopoulos, 2004);
- Basic Constraint (Morgan, 2002);
- List Constraint (Morgan, 2002);
- Enumeration (Morgan, 2002);
- Constraint (Boyer & Mili, 2011).



An example of a Guideline is: *“A customer should not have more than 10 open orders at one time”* (Von Halle, 2011). The following rule categories, found in literature, are similar to the Guideline category and are therefore also not applicable for this research:

- Functional rules (do Prado Leite & Leonardi, 1998);
- Guideline (Wan-Kadir & Loucopoulos, 2004);
- Guidelines (Boyer & Mili, 2011).

In addition, the **‘Action Enabler category’** of Von Halle (2001) does not correspond to derivation business rules. Although this rule category also evaluates facts just like derivation business rules, action enabler rules have a very different purpose of doing this. An Action Enabler business rule evaluates these facts in order to initiate some kind of action (e.g. triggering a business event, an activity or a message). In contrast to a derivation business rule which creates new information, action enabler business rules check conditions and based on the outcome of the evaluation it determines an appropriate action. An example of an Action Enabler rule is: *“If a customer is high risk, then notify the customer services manager”* (Von Halle, 2001). The following rule categories, found in literature, are similar to the Action Enabler category and are therefore also not applicable for this research:

- Action-enablers (Boyer & Mili, 2011);
- Event Condition Action (Boyer & Mili, 2011);
- Action assertion (Wan-Kadir & Loucopoulos, 2004);
- Action assertion (Hay & Healy, 2000).

## 2.3 Controlled Natural Language (CNL)

Previous two sections explained what BRM and business rules are, which types of business rules exist and on which type of business rules this research is focusing. This section will cover how business rules can be captured or in other words be specified in a precise implementation independent way by means of a Controlled Natural Language (see Figure 2.5).

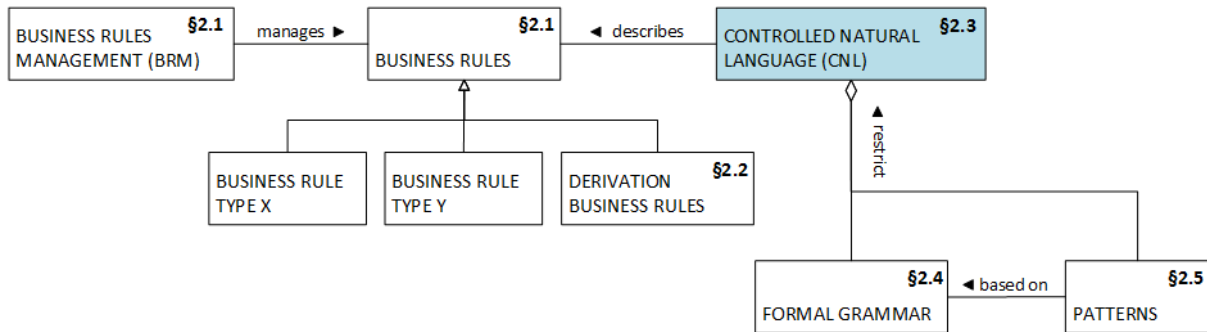


Figure 2.5: Conceptual Overview - Controlled Natural Language

As can be concluded from section 2.2, a lot of different business rule types exist. All of these business rules have to be captured by specifying them using a business rule language. Multiple different business rule languages are devised as mentioned in the Introduction chapter. These languages have different forms of expression. Von Halle (2001) distinguishes four forms of expressing business rules: 1) as a business conversation piece, 2) with a natural language version, 3) with a rule specification language version, and 4) with a rule implementation language version. The latter two are already mentioned earlier in this thesis, only referred to with a different name, namely: implementation independent and implementation dependent languages.

The first form, business conversation piece, specifies a business rule in a very informal way. It can be considered as a note of an employee which makes a first attempt to capture the business rule. The second form, a natural language, is especially used in order to specify business rules that are readable for a business audience. The usage of these first two forms can have different disadvantages, namely: the business rules may become imprecise, incomplete, redundant and inconsistent (Von Halle, 2001). The third form, a rule specification language or implementation independent language, is already slightly more restricted and precise compared to the first two forms. Such a language is created to express a business rule in a declarative way, which means that it specifies what the business rule should do but not how (Von Halle, 2001). Some examples of implementation independent languages are: TDM (Von Halle & Goldberg, 2009), RuleSpeak (Object Management Group, 2008), and SBVR (Object Management Group, 2013). These languages are implementation independent, since they comply with a certain level of naturalness but have a delimited predefined expressiveness, and are not tailored to be applicable for a specific information system (Zoet & Versendaal, 2013). The fourth form, rule implementation languages or implementation dependent languages, are languages that are directly executable for example in a rule engine (Von Halle, 2001). The majority of the Business Rule engine vendors has devised its own language, some examples of these languages are: Corticon, Be Informed, Pega, Berkeley Bridge, Drools and Visual Rules. These languages are implementation dependent as they have a specific grammar which can only be interpreted by a particular information system (Zoet & Versendaal, 2013).

Considering the four forms of expressing business rules, only the third form is applicable for this research. To recall the research goal: creating a pattern catalogue written in a precise implementation independent language which ensures that a business rule set only has to be specified once and can

automatically be deployed to multiple environments. This goal cannot be achieved by creating a pattern catalogue in a language which complies with the first or second expression form, due to the fact that both are not precise enough to be transformed automatically into different executable forms (Von Halle, 2001). Furthermore, the fourth expression form is not applicable since business rules specified in an implementation dependent language can only be deployed in one environment. The third form, implementation independent, could realize transformation to multiple environments and is therefore an applicable expression form for the pattern catalogue. Although various implementation independent business rules languages exist, those current languages are not precise enough to ensure automatic transformation as will be explained later on in this section (Kuhn, 2013).

A promising solution to bridge this gap is the creation of a controlled natural language (CNL). On the one hand, a CNL can adhere to a level of precision which is necessary for a system to interpret the business rule, but in such a way that the language is not restricted to be readable by one specific system (Kuhn, 2010, 2013). On the other hand, a CNL can resemble a natural language making it understandable for humans (Kuhn, 2010, 2013). So, the use of a CNL can positively affect the specification and verification of business rules by humans who mostly lack knowledge about formal notations. A lot of research has been done in the field of CNLs which supports that a CNL can provide the following advantages: 1) improve *communication* among humans, 2) improve machine-assisted *translation* and reduce overall translation costs, and 3) provide an intuitive representation for *formal notations* which makes it easier for humans to use and understand (Aikawa, Schwartz, King, Corston-Oliver, & Lozano, 2007; Chervak, Drury, & Ouellette, 1996; Hallett, Scott, & Power, 2007; O'Brien & Roturier, 2007; Ruffino, 1982; Shubert, Spyridakis, Holmback, & Coney, 1995; Temnikova, 2010).

Kuhn (2010, 2013) studied many different languages which appeared during the last four decades, and concluded that a lot of these languages could be considered as CNL practices. Although a lot of different names are used in literature and practice to describe these CNLs, Kuhn (2013) discovered that these languages share important properties and therefore categorizes them all the same namely as CNL. Kuhn (2013) states that several other terms should not be confused with a CNL even though they are related to CNLs like: sublanguages, fragments of languages, style guides, phraseologies, and controlled vocabularies. These terms are different for instance because: some emerge naturally (e.g. sublanguage) in contrast to a CNL, or some give advice on how to use an existing language (e.g. style guide) instead of describing a new language. On the other hand, CNLs can be considered as a sort of sub-class of three other well-known terms namely Constructed Languages, Artificial Languages or Planned Languages (Kuhn, 2013).

To establish a common understanding and terminology, Kuhn (2013) provides the following elaborate definition: "A language is called a controlled natural language if and only if it has all of the following four properties:

1. *It is based on exactly one natural language (its 'base language');*
2. *The most important difference between it and its base language (but not necessarily the only one) is that it is more restrictive concerning lexicon, syntax, and/or semantics;*
3. *It preserves most of the natural properties of its base language, so that speakers of the base language can intuitively and correctly understand texts in the controlled natural language, at least to a substantial degree;*
4. *It is a constructed language, which means that it is explicitly and consciously defined, and is not the product of an implicit and natural process (even though it is based on a natural language that is the product of an implicit and natural process)."*

To summarize, Kuhn (2013) also gives a more shorter definition: "A controlled natural language (CNL) is a constructed language that is based on a certain natural language, being more restrictive concerning lexicon, syntax and/or semantics while preserving most of its natural properties".

Due to the diversity of languages that can be seen as CNL, it is hard to get a clear view of the fundamental properties of a CNL. Therefore, Kuhn (2013) identified two main categories of properties to classify and characterize CNLs: environmental properties and language properties. Below, both categories are explained in detail.

### ENVIRONMENTAL PROPERTIES

Nine different environmental properties of CNLs are identified, which tell something about the goal and the form of the CNL. The environmental properties are explained and denoted with a specific letter code in Table 2.3.

Code	Property
c	The goal of the CNL is to improve <i>comprehensibility / communication</i> among humans (e.g. speakers of different native languages).
t	The goal of the CNL is to improve <i>translation</i> and reduce overall translation costs.
f	The goal of the CNL is to provide an intuitive representation for <i>formal notations</i> which makes it easier for humans to use and understand formal formalisms.
w	The CNL is intended to be written.
s	The CNL is intended to be spoken.
d	The CNL is designed for a specific narrow domain.
a	The CNL is originated from academia.
i	The CNL is originated from industry.
g	The CNL is originated from government.

**Table 2.3:** Environmental properties of CNLs

CNLs which comply with property ‘c’ are commonly called *human-oriented CNLs*. These CNLs are especially designed to enhance the communication among humans, and can also enhance the comprehensibility of technical documentation. Human-oriented CNLs are devised for sufficient understandability by humans, not towards processability by a system (Kuhn, 2013). Examples of human-oriented CNLs are: Caterpillar Fundamental English (CFE), FAA Air Traffic Control Phraseology (FAA), and Basic English (Kuhn, 2013).

CNLs which comply with property ‘t’ and/ or ‘f’ are commonly referred to as *machine-oriented CNLs*. These CNLs are particularly designed to enhance the communication between humans and information systems. Machine-oriented CNLs have two prominent characteristics: 1) complete unambiguousness and 2) the possibility to be defined by formal grammars with a direct mapping to formal logic. Examples of machine-oriented CNLs are: Attempto Controlled English (ACE), Processable English (PENG), KANT Controlled English (KCE), and Controlled Language Optimized for Uniform Translation (CLOUT) (Kuhn, 2013).

## LANGUAGE PROPERTIES

Based on many language properties found in existing literature, Kuhn (2013) derived the following four fundamental language properties, abbreviated as PENS. A detailed description of each property can be found in (Kuhn, 2013).

- **Precision:**  
This property indicates the degree to which the meaning of a text, defined with a CNL, is directly clear from its textual form. This implies that the CNL has to be unambiguous.
- **Expressiveness:**  
This property indicates the range of statements that a certain CNL is able to express. In other words, the degree to which communication can be captured.
- **Naturalness:**  
This property indicates the degree to which a text, defined with a CNL, resembles a natural language. In other words, the statements of a CNL should be understandable and readable for speakers of the concerned natural language.
- **Simplicity:**  
This property refers to the degree of simplicity to define the language (CNL) in terms of syntax and semantics. Furthermore, this property covers the effort needed to implement the language in a computer program. The indicator which is used for simplicity is: the number of pages needed to describe the language in an exact and comprehensive way.

The four properties are also called dimensions, since there is a large variety in the degree to which a CNL adheres to one of the four properties. CNLs can be positioned somewhere between a natural language (high expressiveness and naturalness, but low precision and simplicity) and a formal language (high precision and simplicity, but low expressiveness and naturalness). To be more accurate in the classification and identification of the nature of a CNL, Kuhn (2013) constructed PENS as a classification scheme including a five-tier ranking (1 - 5) for each of the four dimensions where:

1. P1 = an imprecise language and P5 = a language with fixed syntactic and semantics;
2. E1 = an inexpressive language and E5 = a language with maximal expressiveness;
3. N1 = an unnatural language and N5 = a language with natural texts;
4. S1 = a very complex language and S5 = a language with very short descriptions.

An extensive explanation of each rank per dimension is provided in Appendix 2. It should be noted that this ranking does not indicate the quality or usefulness of a CNL. The PENS classification scheme is useful to identify the nature of a language in order to select a CNL which is applicable for a specific application domain and purpose. In addition, the classification scheme is useful when devising a new CNL because some tradeoffs have to be made between the properties. According to Kuhn (2010, 2013), a language cannot entirely comply with all four properties since they are frequently in conflict. Most of these conflicts are very obvious but also supported by statistical proof. Kuhn (2013) found different correlations, both positively and negatively, between some of the properties. A negative correlation corresponds to a pair that is in conflict, which are the following: 1) *precision* and *expressiveness* (Spearman's rank correlation coefficient  $p = -0.66$ ), 2) *precision* and *naturalness* ( $p = -0.67$ ), 3) *expressiveness* and *simplicity* ( $p = -0.82$ ), and 4) *naturalness* and *simplicity* ( $p = -0.76$ ). For the first pair it means that how higher the precision level of a language, the lower the expressiveness of the language. This is also how the correlations between the remaining three pairs can be interpreted.

## CNL APPLICATION DOMAINS

Many different CNLs are created for several domains, examples of these domains are: computer science, philosophy, and linguistics (Kuhn, 2010, 2013; Pool, 2006). Pool (2006) investigated 41 CNLs and found that the majority of these CNLs were designed to be applicable for a single domain. Only four CNLs were created to be applied in multiple domains. Taking this finding into account, CNLs can be related to the concept: ‘Domain Specific Languages’ (Pool, 2006; Ranta, 2014; Sun, Demirezen, Mernik, Gray, & Bryant, 2008). A domain-specific language (DSL) is defined as: “a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain” (Pool, 2006; Ranta, 2014; Sun et al., 2008; Van Deursen & Klint, 2002; Van Deursen, Klint, & Visser, 2000).

In some cases, a language can be considered both a CNL and a DSL at the same time. This is for example true for the Structured Query Language (SQL), which is a language specifically applicable for the relational database domain and complies with the four requirements of a CNL (Kuhn, 2013; Van Deursen & Klint, 2002; Van Deursen et al., 2000):

1. *It is based on exactly one natural language (its base language “English”);*
2. *It is more restrictive concerning lexicon, syntax, and/or semantics than its base language (i.e. based on relational theory);*
3. *It can intuitively be understood, at least to a substantial degree;*
4. *It is a constructed language.*

These above four language requirements distinguish a CNL from a regular DSL.

In addition, Pool (2006) found that CNLs are created based on different natural languages (e.g. German, Chinese, French). Kuhn (2013) identified hundred CNLs with as natural language English and also provides a list of specific application areas of CNLs along with some examples (see Table 2.4):

Application Area	Examples of CNLs per Area
<b>Semantic Web</b>	▪ <i>OWL Simplified English</i> (Power, 2012)
<b>Technical Documentation</b>	▪ <i>KANT Controlled English</i> (Mitamura & Nyberg, 1995)
<b>General-Purpose Knowledge Representation</b>	▪ <i>Computer Processable Language (CPL)</i> (Clark, Harrison, Jenkins, Thompson, & Wojcik, 2005) ▪ <i>Controlled English to Logic Translation (CELT)</i> (Pease & Li, 2010)
<b>Personal Rules and Scripts</b>	▪ <i>Voice Actions</i> (Google, 2015)
<b>Emergency Instructions</b>	▪ <i>Controlled Language for Crisis Management (CLCM)</i> (Temnikova, 2010)
<b>Query Interfaces</b>	▪ <i>Structured Query Language (SQL)</i> (Chamberlin & Boyce, 1974)
<b>International Communication</b>	▪ <i>FAA Air Traffic Control Phraseology</i> (FAA, 2014) ▪ <i>PoliceSpeak</i> (Johnson, 2000)
<b>Mathematical Texts</b>	▪ <i>Controlled Language of Mathematics (CLM)</i> (Humayoun & Raffalli, 2010)
<b>Software Specifications</b>	▪ <i>Gherkin</i> (Necas, 2011)
<b>Legislation/Government Documents</b>	▪ <i>Massachusetts Legislative Drafting Language</i> (Massachusetts Senate, 2003)
<b>Policies / Business Rules</b>	▪ <i>PERMIS Controlled Natural Language</i> (Inglesant, Sasse, Chadwick, & Shi, 2008) ▪ <i>SBVR Structured English</i> (Object Management Group, 2013) ▪ <i>RuleSpeak</i> (Object Management Group, 2008)

**Table 2.4:** CNL Application Areas along with Examples of CNLs

From the hundred English-based CNLs Kuhn (2013) identified, only three were tailored to the ‘Policies / Business rules’ application area. One of the three CNLs is established to define policies, namely the CNL called “PERMIS Controlled Natural Language”. The PERMIS Controlled Natural Language is especially used for access control policies of which examples are shown in Figure 2.6. A business policy in general is defined by Object Management Group (2008) as: “A non-actionable directive whose

purpose is to govern or guide the enterprise.” Furthermore, Object Management Group (2008) states that a business rule is derived from business policy. Taking these statements into account, PERMIS is not applicable as CNL for this research as it focuses on specifying policies instead of behavioral (operational) business rules like derivation business rules.

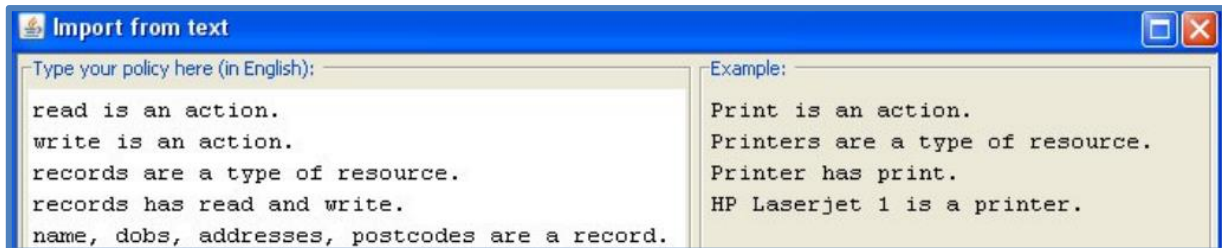


Figure 2.6: Examples access control policies adapted from Kuhn (2013)

The remaining two CNLs that Kuhn identified in the ‘Policies/ Business rules’ application area are focused on defining business rules, namely: RuleSpeak and SBVR Structured English. For the purpose of this research, solely these two CNLs are interesting to consider in more detail. RuleSpeak is introduced in 1994 and developed by Ronald G. Ross (Kuhn, 2013). In 2005, SBVR Structured English is introduced which is very similar to RuleSpeak (Kuhn, 2013). The similarity can be explained by the fact that both CNLs are compliant with the formal semantics defined in the SBVR standard (Object Management Group, 2008). The vocabulary of both languages consists of fixed sentence constituents (i.e. ‘building blocks’) which are divided into four types and denoted in a specific format:

- **terms** (i.e. concepts)
- **names** (i.e. individuals)
- **verbs** (i.e. relations)
- **keywords** (i.e. fixed phrases, quantifiers and determiners)

Although the permitted sentence constituents are provided, the order in which they can be placed is not enforced. So, both SBVR as RuleSpeak do not include a formal grammar and syntax which allows the specification of ambiguous business rules (Kuhn, 2013). This is illustrated in Table 2.5, where the same business rule is specified in two different ways by applying SBVR Structured English:

SBVR 1	It is obligatory that the <b>country of the return branch of each international inward rental is the country of registration of the rented car of the rental.</b>
SBVR 2	"If the <b>country of the pick-up branch of a rental is not the country of registration of the rented car of the rental then it is obligatory that the country of the return branch of the rental is the country of registration of the rented car.</b> "

Table 2.5: Example of an SBVR business rule with different syntax

Another example business rule is provided Table 2.6, which shows how the same business rule is specified with SBVR and subsequently with RuleSpeak. Considering this example, the possibility exists that two people will interpret this same business rule in a different way. For example, does this business rule imply “**each in-country rental AND each international inward rental**” or “**an in-country rental OR international inward rental**”. So, both specifications are not precise and leave room for interpretation.

SBVR	It is obligatory that <u>at the actual return date/time of each in-country rental and each international inward rental the local area that includes the return branch of the rental owns the rented car of the rental.</u>
RuleSpeak	The <u>local area that includes the return branch of an in-country rental or international inward rental must own the rented car of the rental at the actual return date/time of the rental.</u>

**Table 2.6:** Example of a business rule specified in an ambiguous way

Taking all previous statements and examples into account, both CNLs are classified in the exact same way by means of the PENS classification scheme and by the environmental properties:  $P^3E^4N^4S^2$ , c f w i (Kuhn, 2013). For an explanation of the letter codes, see Table 2.3 and Appendix 2.

As described in Chapter 1, the pattern catalogue that will be created during this research should allow automatic transformation by means of a parser. To realize this, the language in which the pattern catalogue is specified (i.e. CNL) should minimally comply with a precision level of P4 (see Appendix 2) which is not the case for SBVR and RuleSpeak. To reach this level, one of the requirements is a formal grammar underlying the CNL. The approach that will be pursued within this research is to create a formal grammar, which will be explained in the next sub-section.



## 2.4 Formal Grammar

In the preceding three sections, information is provided about BRM and business rules in general. Furthermore, insight into derivation business rules and CNLs is provided. Current section examines what formal grammars are (see Figure 2.7).

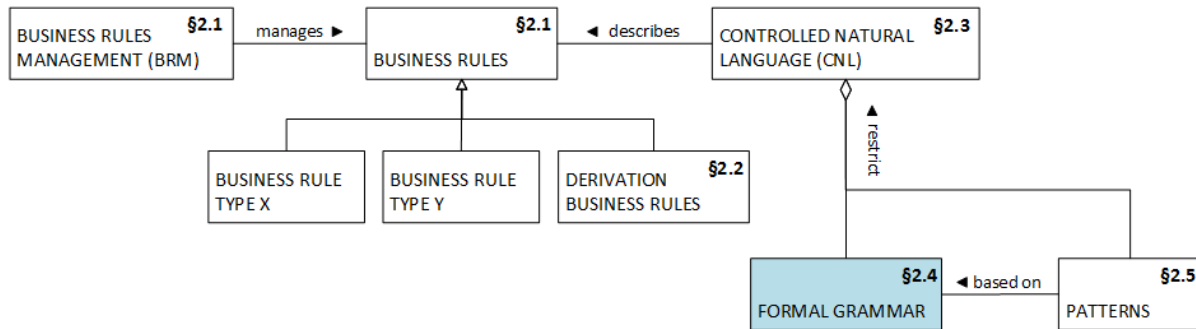


Figure 2.7: Conceptual Overview – Formal grammar

A language, whether it is a natural or formal one, is mostly defined by a grammar (Kuhn, 2010). According to Kuhn, CNLs have specific requirements with regard to their grammars that differ from those of natural or other formal languages. The grammar of a CNL has to comply with the following three requirements in order to be able to define, implement, use, and reuse the CNL efficiently (Kuhn, 2010): 1) Concreteness, 2) Declarativeness, and 3) Implementability. Concreteness implies that the grammar of the CNL is fully formalized and can be interpreted by automated information systems. This first requirement corresponds to *precision*, one of the language properties of a CNL, as explained earlier. The second requirement, declarativeness, is essential in order for a CNL to be used by different tools. A grammar can be called declarative if it is independent from a concrete implementation. In other words, the grammar is separated from the parser that will process it. This ensures reusability of the grammar and makes it possible to change or replace the parser without the need of changing the grammar. From a more practical point of view, implementability is important indicating that the grammar is easy to implement. Implementability is closely related to the usability of the CNL, appropriate implementation of the grammar can ensure sufficient usage of the CNL.

A grammar that fulfills the three described requirements can be called a *formal grammar* (Kuhn, 2010). A formal grammar in the context of this research is defined as a set of rules for specifying the syntax of strings (i.e. sequence of characters or words) (Gallier, 2011). Rules that define the syntax of strings are called *grammar rules* (Gallier, 2011). To rewrite (transform) these strings, there also exist rules that are considered as *production rules* in the information science domain (Gallier, 2011). Each production rule has at the left-hand side an *input* (i.e. source) which is the string that can be replaced, and at the right-hand side an *output* (i.e. target) which is a string that should replace it. A production rule is mostly expressed in the form: *input* → *output*. The input and output part of a production rule are composed of non-terminal and terminal symbols. Where terminals are defined as “the symbols which cannot be changed using the rules of the grammar”, and non-terminals as “the symbols which act like variables” (Gallier, 2011). With regard to the grammar rules of a formal grammar, those restrict the language in the set of fundamental constructs it can use and also to some extent the order in which they can be placed. In terms of the CNL, the grammar rules specify the syntax for specifying the source business rules.

A formal grammar can be used for parsing (Earley, 1970; Gallier, 2011), which is the intention of devising a formal grammar for the envisioned CNL of this research. What is meant by the word *parsing* can differ per discipline. In the context of this research, parsing is the automated process that consists of two main streams:

- 1) Firstly, decomposing a string (i.e. business rule specified in the controlled natural language) into its constituents (i.e. fundamental constructs) by analyzing it with the grammar rules of the formal grammar of the source language (i.e. CNL). This first step results in a parse tree that depicts the syntactic relations among the fundamental constructs;
- 2) Secondly, transforming this parse tree to a string (i.e. business rule specified in target language) that complies with the syntactic rules of the target language by means of the production rules of the formal grammar.

Formal grammars are often applied when a precise description of a language is required, such as for: manuals, communication protocols, and programming languages (Earley, 1970; Kuhn, 2010). Given the precision requirement of the envisioned CNL, the choice is made to create a formal grammar underlying the CNL. This formal grammar will impose the syntax of the fundamental constructs of the CNL by means of the grammar rules. The production rules, which specify how business rules specified with the CNL can be transformed into business rules in a target language, will not be devised during this research. Due to the fact that this research focuses on the fundamental constructs and patterns as a result of time constraints. The grammar rules and the fundamental constructs will be described in Chapter 3.

## 2.5 Patterns and pattern catalogues

The former four sections covered the concepts BRM, business rules, derivation business rules, CNLs, and formal grammars. Besides the envisioned CNL and grammar rules, the aim is to provide the business rule authors with a mechanism to consistently specify business rules in a proper way. Patterns are considered as such a mechanism, since they can be used as enforcements for business rule authors when specifying the business rules. In the context of this research, patterns will be fixed combinations of fundamental constructs adhering to the grammar rules of the formal grammar. In this way, patterns can make a language (i.e. the CNL) even more restrictive and precise. The advantage of applying patterns for the business rule authors is that they can decrease the duration of the design process and enhance the consistency (i.e. standardization). This section will define what (design) patterns and pattern catalogues are (see Figure 2.8). Moreover, current existing pattern catalogues will be reviewed and compared.

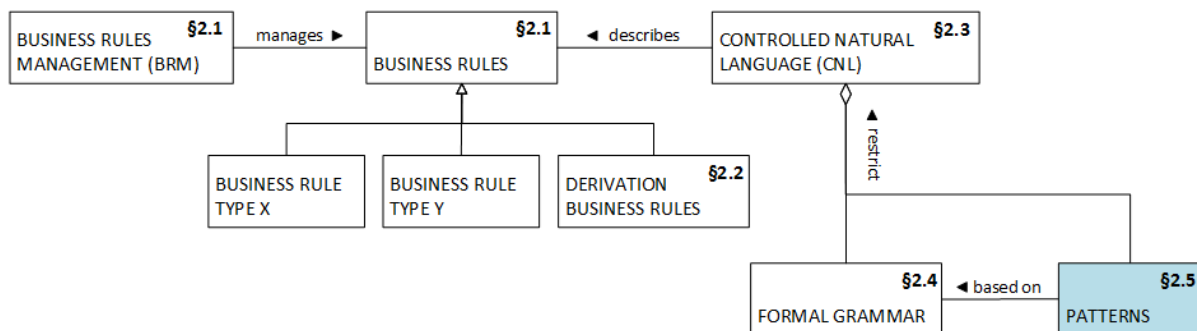


Figure 2.8: Conceptual Overview - Patterns

Design patterns, or patterns in general, were first brought to the attention by the work of Alexander, Ishikiwa, and Silverstein (1977) in the architecture domain for constructing buildings (Graham, 2006; Iacob, Lankhorst, & Schrier, 2012). Alexander et al. (1977) stated the following: “A pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.”

Later on, patterns were established for many other disciplines (Graham, 2006; Iacob et al., 2012). Gamma, Helm, Johnson, and Vlissides (1994), also known as the Gang of Four (GoF), made patterns familiar in the software development domain. From a software development view, they defined a design pattern as “a generally applicable solution to a common design problem, codified in a standardized form providing a configuration of elements that together solve the problem”.

Morgan (2002) views patterns from an information system perspective and defines a pattern as “a collection of model elements relating to a particular situation. It is a fragment of a model that's already available, ready for you to incorporate - either as it stands or with modifications - into your own complete model.” Another definition is found in Zoet (2014): “A pattern is a structured way of presenting key elements which can be used to create or identify statements”. Von Halle (2004) uses the word *rule template* to refer to a pattern, and defines rule templates as “disciplined patterns by which a business rule is expressed as a combination of rule clauses”. This latter definition is already more in line with this research since it is specifically tailored to the business rule domain. Regarding this research, *key elements* (Zoet, 2014) and *rule clauses* (Von Halle, 2001) are considered as the fundamental constructs (i.e. building blocks or business rule sentence parts) for specifying business rules. Previous definitions are combined to arrive at the following definition of a pattern, which will be applied for this research: “a structured combination of fundamental constructs to specify business rules” (Ghose & Koliadist, 2007; Morgan, 2002; Von Halle, 2001; Zoet, 2014).

Although each above provided definition originates from a different domain, in the core they all state the same namely that patterns are standard solutions (templates) to a recurring problem in a context. Due to this generic nature of patterns, they can be applied in all practices to reuse knowledge. Iacob et al. (2012) identified the following pattern types, each for a specific application domain: object-oriented design patterns, enterprise software application design patterns, enterprise integration patterns, organizational patterns, enterprise architecture management (EAM) patterns, workflow patterns, e-business patterns, software oriented architecture (SOA) patterns, ontology design patterns, user interface design patterns, rule patterns and multichannel management patterns. An explanation of these types of patterns can be found in (Iacob et al., 2012).

The application of patterns can have several advantages (Graham, 2006; Iacob et al., 2012): 1) simplify the communication about design, 2) educate new employees (designers) how to properly design, 3) enhance standardization of design which can prevent re-design, and 4) reuse of (design) knowledge in different contexts which can decrease the design time.

### CURRENT RULE PATTERN CATALOGUES

As described above, various pattern catalogues exist for different domains and/or focus on different granularity levels. Merely the pattern catalogues established for the BRM domain and which are focused on the level of specifying business rules will be considered into more detail. So even though the pattern catalogue of Graham (2006) is created for the BRM domain (i.e. rule catalogue), these patterns are very high-level and comprise advices for all kind of activities in a BRM project. However, from literature also a number of rule pattern catalogues emerged which are relevant. These catalogues are described below.

**do Prado Leite and Leonardi (1998)** proposed a taxonomy to categorize business rules into *functional* and *non-functional* business rules. Non-functional business rules are further subdivided into *macrosystem* and *quality* rules. A description of these categories can be found in Appendix 1. For every category in the taxonomy, a rule pattern is created which contains a combination of the following fundamental constructs: property, non-verb phrase, relation, verb phrase, should, should not, must, must not, because and cause. For an explanation of these pattern elements and patterns see Leite & Leonardi (1998).

The pattern catalogue of **Von Halle (2001)** is established with a higher granularity level. She distinguishes business rules based on their intention which lead to the following five rule categories and equal pattern types: 1) mandatory constraints, 2) guidelines, 3) action enabler rules, 4) computations and 5) inferences. For an explanation of these categories, see Appendix 1. The five patterns differ to a great extent with regard to their fundamental constructs, therefore only the most frequently applied and basic ones are provided here: <term>, <formula>, <value list>, <at least, at most, exactly n of>, <comparison>, <value>, <operator>, <rule phrase(s)>, IS COMPUTED AS, MUST HAVE, MUST BE, IF, THEN, MUST NOT and BE IN LIST. To view the patterns including fundamental constructs, see Von Halle (2011).

**Morgan (2002)** established a pattern catalogue also including five different patterns: 1) basic constraint, 2) list constraint, 3) classification, 4) computation, 5) enumeration. These patterns are an elaborated version of the basic form defined by Morgan: <subject> must <constraint>. The remaining elements that a pattern can contain are: <det>, <characteristic>, <fact>, <fact-list>, <m>, <n>, <result>, <algorithm>, <classification>, and <enum-list>. For an explanation of these fundamental constructs and patterns, see Morgan (2002).

In 2003, **Wan-Kadir and Loucopoulos** published a pattern catalogue based on the following typology to categorize business rules and their corresponding patterns: 1) constraint, 2) guideline, 3) action assertion, 4) computation, and 5) inference. For each category an associated pattern is provided. One

year later, they published a new version of this pattern catalogue along with their Business Rule Model to capture and specify business rules (**Wan-Kadir & Loucopoulos, 2004**). Although pattern four (i.e. computation) and five (i.e. inference) have remained exactly the same, several differences can be appointed between the two catalogue versions. As first, the elements of the first three patterns are substantively changed. Furthermore, the first and third pattern provide multiple options for the same pattern instead of only one. Moreover, the first pattern is renamed from constraint to *mandatory constraint*. In addition, the new pattern catalogue has an additional layer on top of the five patterns which clusters them into three main categories: 1) *Constraint* which includes mandatory constraint and guideline, 2) *Action assertion* which includes action assertion along with three options (enabler, copier and trigger), and 3) *Derivation* which includes computation and inference. A description of each business rule category from the latest version can be found in Appendix 1. Only this version is considered since it is more extensive and up to date. Since the five patterns differ to a great extent with regard to their fundamental constructs, not all the fundamental constructs will be listed here but only the most common used ones: <subject>, <value>, <condition>, <algorithm>, <fact>, <event>, <condition>, <action>, IS COMPUTED AS, MUST [NOT], MAY, IF, and THEN. For an explanation of the fundamental constructs and patterns, see Wan-Kadir and Loucopoulos (2004).

**RuleSpeak** is a business rule language for which **Hoppenbrouwers (2011)** devised a pattern catalogue including eighteen different rule patterns. Using this pattern catalogue to specify business rules ensures that the business rules comply with the RuleSpeak requirements. Each pattern in the catalogue is composed of the following fixed sequential parts: First part, Keyword(s), Second part, Keyword(s), and Third part. The first part always includes a subject and the third part always includes a condition. Especially the keyword(s) and second part cause the differences between the patterns, as they state if something 'must', 'need not' or 'may' be done and what 'should' be done (e.g. computation). For an explanation of these fundamental constructs (i.e. pattern parts) and patterns see (Hoppenbrouwers, 2011). Although RuleSpeak makes a distinction between structural and operational business rules, the created rule patterns are not specifically applicable for a certain business rule type. For a definition of the two rule types, see Appendix 1.

In contrast to the previous five catalogues, **Caron et al. (2013)** devised a very extensive rule taxonomy which is entirely centered around business rules with the aim to constrain or guide business processes. The taxonomy has two dimensions: a *process mining perspective* dimension and a *rule restriction focus* dimension. The first dimension provides four main groups (perspectives) to cluster the patterns:

1. The functional process perspective;
2. The control-flow process perspective;
3. The organizational process perspective;
4. The data process perspective.

All four perspectives deal with one of the following aspects, all related to business processes: the process elements (e.g. activities) that are being performed, the process behavior (i.e. when process elements can be performed), the performers of the business process (e.g. the actors), or the information elements (e.g. data) that are used, produced or changed.

For each of the four perspectives, the second dimension provides a subdivision for the patterns into the following five sub-groups:

1. Cardinality-based rules;
2. Coexistence rules;
3. Dynamic data-driven rules;
4. Relative time rules;
5. Static property rules.

Appendix 1 provides a definition of the five business rule types. All these business rule types are aimed at restricting or specifying the dynamic or static properties of process elements of a specific process instance. For example, the first type restricts the number of allowed instances of a specific process element (e.g. an activity of type a1 must be performed at least once).

In total, the rule catalogue of Caron et al. (2013) consists of twenty categories (4 main groups x 5 sub-groups). For every category, several patterns are created. Given the large amount of patterns and variation of fundamental constructs, these patterns including pattern elements can be viewed in Caron et al. (2013).

Besides the above six business rule pattern catalogues that were found in literature, also a pattern catalogue devised and provided by the case company is taken into account for this research. This pattern catalogue is called “**RegelSprak**” (Sangers-van Cappellen, 2014) and is established in Dutch, since the business rules of the Dutch Tax and Customs Administration are specified in this language. RegelSprak deviates from the other pattern catalogues with regard to the clustering of patterns. The catalogue includes individual patterns for specifying the conclusion part (THEN part) of a business rule, and individual patterns for specifying the condition part (IF-part) of a business rule. In contrast, all of the other six pattern catalogues created patterns including both parts. In total, the RegelSprak pattern catalogue includes 31 patterns. For the condition part, seven different patterns are created. For the conclusion part, 24 patterns are established which are clustered into several subcategories of three main categories as follows:

1. Derivation rules:
  - Decision rule patterns;
  - Calculation rule patterns.
2. Constraint rules:
  - Value Range rule patterns;
  - Consistency Control rule patterns;
  - Rounding rule patterns.
3. Process rule patterns.

As explained in section 2.2, this research focuses on derivation business rules. Taking all the previous described business rule pattern catalogues into account, none of these catalogues is completely focused on derivation business rules. Therefore, during this research a pattern catalogue solely focused on the specification of derivation business rules will be created.

## 2.6 Summary literature review

In the previous sections the following concepts were described: BRM, business rules, derivation business rules, CNL, formal grammar, and patterns (see Figure 2.9). This section will relate all previous sections to the primary research goal: Creating a pattern catalogue specified in a precise implementation independent language, which can be used to specify a set of derivation business rules once, and which allows automatic transformation of the business rule set to be applicable for multiple business rule engines.

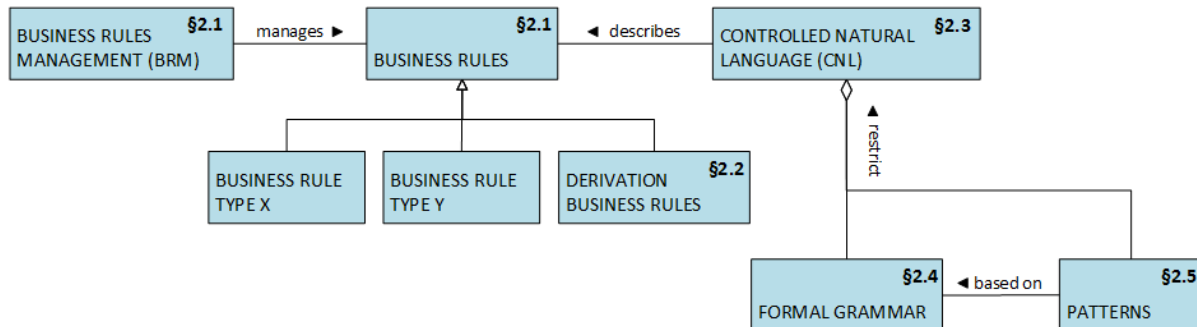


Figure 2.9: Summary Literature Review

Given the above stated research goal, the implementation independent language in which the pattern catalogue will be specified should be:

1. Precise (P) in order for an automated information system to parse the business rules specified with the language;
2. Expressive (E) enough to be able to capture derivation business rules;
3. Natural (N) to some extent to be understandable for a human which will specify and verify the business rules;
4. Simple (S) to be defined with a formal grammar in terms of syntax (Kuhn, 2013).

A machine-oriented CNL complies with the four language requirements listed above as it is designed to improve the communication between humans and information systems (satisfying 2 & 3). Furthermore, these CNLs can be created along with a formal grammar with a direct mapping to formal logic (satisfying 1 & 4). Furthermore, the formal grammar can be used as an interlingua which enables transformation by parsing the source language into different target languages (Ranta, 2014). In this way, a CNL can also implicitly comply with an implementation independent language.

In summary, a CNL is highly applicable to achieve the research goal. At this moment, merely two CNLs for business rules specification are found by Kuhn (i.e. RuleSpeak and SBVR Structured English). Since those two CNLs are not strictly defined by means of a formal grammar as explained in section 2.3 and have a higher expressiveness than necessary by providing the possibility to express multiple different types of rules, they are not suitable to address the research goal. In addition, the pattern catalogue of Caron et al. (2013) can also be considered as CNL by taking the language properties of Kuhn (2013) into account. Although this CNL is precise enough, the expressive power of Kuhn's CNL is only applicable for specifying process rules instead of derivation business rules.

To bridge this gap, a machine-oriented CNL targeted at specifying derivation business rules will be created during this research. The underlying natural language of the CNL will be English given the fact that English is widely spoken and is the common language in the academic world (Kuhn, 2010). As mentioned in section 2.3, it is important to assess the tradeoffs between the four language properties (PENS) when creating a new CNL. Given the fact that this positioning will provide a clear view for the creation process (Kuhn, 2013). For that reason, the initial idea of the CNL that will be created during this research is mapped onto the PENS classification scheme of Kuhn (2013). Moreover, the

environmental properties are taken into account. These will be explained first. In terms of environmental properties, the envisioned CNL will have the following properties: it will be a language with as goal to improve translation (**t**) and to provide a representation of the formal notation which remains understandable for humans (**f**), it will be a written (**w**) language, originating from government (**g**) and be used in a specific domain (**d**) namely for business rules.

The four language properties of the envisioned CNL have already been briefly discussed above. Now, each property will be considered into more detail by describing the specific required level for the CNL by means of the five-tier ranking and related criteria of Kuhn (2013). The **precision (P)** level of the CNL should at least be equal to 4 in order to be able to parse the language with an automated information system. Level 4 requires that a language is fully formal on a syntactic level, which can be reached by formulating a formal grammar. The highest precision level, level 5, is not desired for the envisioned CNL since it requires that the language is also fully specified on a semantic level. A specification of the semantics can only be achieved when every subject of the business rule set is defined prior to the business rules specification. The high level of precision will restrict both the level of **expressiveness (E)** and **naturalness (N)** of the CNL to a certain extent, since only specific language structures can be used. However, the aim is to have a minimal level of 3 for expressiveness to be able to capture general rule structures (if / then) which is otherwise not possible. In addition, also a minimal level of 3 for naturalness is required to ensure that the language is understandable for a human to write and verify the business rules. When a language corresponds to a lower level (1 or 2), the language looks very unnatural mostly due to the heavy use of symbols and is therefore not considered as a CNL according to Kuhn's definition. For the last property of the CNL, **simplicity (S)**, it is desired to comply with level 4 which means that the CNL can be defined in an exact and comprehensible way requiring between one and ten pages. This exact level of simplicity is chosen since a simplicity level of 3 implies more than ten pages, and a simplicity level of 5 implies that the descriptions fits on a single page. Altogether, the aim is to devise a CNL with the following PENS levels: P= 4, E= 3, N =3, S=4. For a broader explanation of these levels, see Appendix 2. The levels of this new CNL are graphically compared to these of SBVR and RuleSpeak, see Figure 2.10.

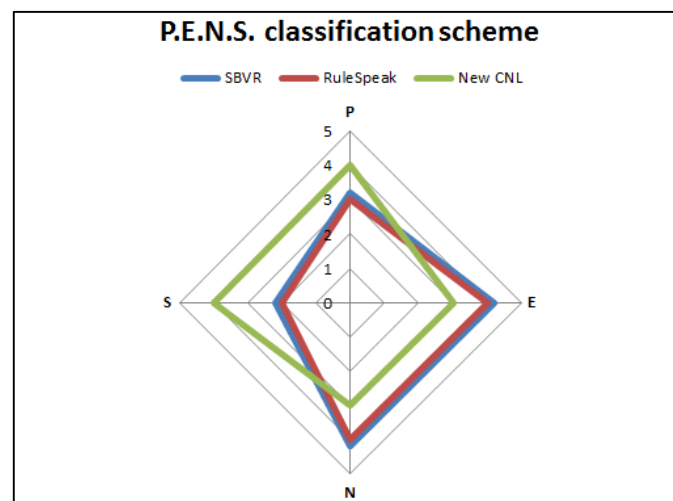


Figure 2.10: PENS classification scheme for several CNLs

As mentioned above, a formal grammar needs to be formulated underlying the CNL to reach a high precision level. The grammar rules will restrict the syntax of the fundamental constructs of the CNL. When the fundamental constructs and grammar rules are devised, the patterns will be created and specified by means of the CNL. These patterns will make the CNL even more restrictive by providing a set of fixed combinations of fundamental constructs that comply with the grammar rules.



### 3 CNL Creation

The previous chapter identified all the components that are necessary to create or to support a CNL that complies with the purpose of this research. These components are: fundamental constructs, a formal grammar (i.e. the grammar rules), and patterns. In this chapter, the fundamental constructs will be identified for specifying derivation business rules along with the grammar rules that restrict and/or impose the application of these fundamental constructs. Together, the fundamental constructs and grammar rules constitute the meta-model to which the CNL has to conform. This meta-model is included at the end of this chapter.

#### BUSINESS RULE

From the literature study, information with regard to specifying business rules emerged. In general, it became clear that on the highest level a business rule is composed of two parts: the conclusion part and condition part (Von Halle & Goldberg, 2009; Zoet et al., 2011). In the following example business rule, the conclusion part is denoted by an orange border and the condition part by a green border:

*The tax amount of a taxpayer must be calculated as the sum of the salary of each current employment minus the tax rebate if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.*

Although for this research these two fundamental constructs are designated as ‘conclusion part’ and ‘condition part’, different alternatives can be found in literature. For instance, the conclusion part is also referred to as ‘conclusion assertion’, ‘consequent’ or ‘then-part’, and the condition part as ‘if-part’ ‘antecedent’ or ‘when-part’ (Von Halle & Goldberg, 2009; Wong, Whitney, & Thomas, 1999; Zoet et al., 2011). The relationship between both fundamental constructs and the number of times they occur in one business rule can also differ per source, for example due to personal choice of the business rule modeler or the business rule language that is used (Zoet et al., 2011). Most languages allow the exclusion of a condition or the inclusion of one or multiple condition parts (conditions) and allow only one conclusion part (conclusion). Meeting these requirements ensures the creation of atomic or normalized business rules (Boyer & Mili, 2011; Von Halle & Goldberg, 2009; Zoet et al., 2011). Atomic business rules are “*business rules that cannot be further decomposed without losing meaning*” (Boyer & Mili, 2011). In contrast, other languages allow multiple conclusion parts (conclusions).

Enforcing only one conclusion part is desirable since it can provide several advantages: 1) it eliminates ambiguity of meaning, 2) enhances understandability, maintainability, execution efficiency and manageability, 3) increases ease of validation and implementation, and 4) can eventually prevent redundant or overlapping business rules (Boyer & Mili, 2011; Von Halle & Goldberg, 2009; Zoet et al., 2011). Given these advantages, the creation of atomic business rules will be enforced by the envisioned CNL which corresponds to the following grammar rules:

- ✓ A Derivation business rule consists of **exactly one** Conclusion Part;
- ✓ A Derivation business rule consists of **zero or more** Condition part(s);
- ✓ A Conclusion part belongs to **exactly one** Derivation business rule;
- ✓ A Condition Part belongs to **exactly one** Derivation business rule.

The relationships and cardinalities of the two fundamental constructs of a derivation business rule are shown in Figure 3.1.

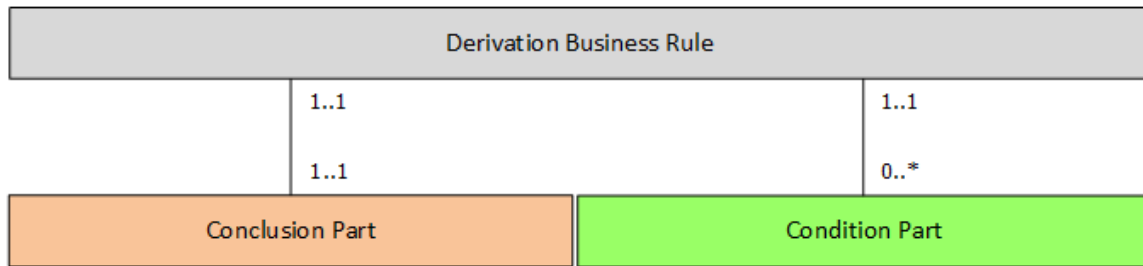


Figure 3.1: Relationships and cardinalities

### 3.1 Equivalent Underlying Fundamental Constructs

The conclusion part and condition part have many similar underlying fundamental constructs, which were determined by examining literature and business rule catalogues (Caron et al., 2013; do Prado Leite & Leonardi, 1998; Hay & Healy, 2000; Hoppenbrouwers, 2011; Morgan, 2002; Object Management Group, 2013; Sangers-van Cappellen, 2014; Von Halle, 2001; Von Halle & Goldberg, 2009; Wan-Kadir & Loucopoulos, 2004). Only a few differences with regard to the underlying fundamental constructs and grammar rules are found between both parts. These differences will be explained later on in section 3.2 and 3.3. First, the discovered equivalent underlying fundamental constructs will be described in succession. This will be done by taking the definition of a derivation business rule into account again:

*“an expression that evaluates facts, by means of a calculation or classification, leading to a new fact (i.e. conclusion)”*

#### FUNDAMENTAL CONSTRUCT: SUBJECT

Looking at the definition above, for this research facts are seen as values, pieces of information or data, that can be filled in for a specific business concept incorporated in a business rule (Von Halle & Goldberg, 2009). In general, a business concept is considered as *“a noun, a thing with an agreed-upon definition, a recognizable business entity”* (Morgan, 2002; Von Halle, 2001; Von Halle & Goldberg, 2009). With regard to business rule specification, a business concept is seen as one of the fundamental constructs only many different names are used to refer to a business concept. In the rule pattern catalogues, the following expressions of a business concept were found: Von Halle (2001) uses <term>, Morgan (2002) uses <subject> and <result>, Wan-Kadir and Loucopoulos (2004) use <subject> and <value>, and Hoppenbrouwers (2011) uses SUBJ.

Above mentioned expressions and some additional synonyms to refer to a business concept were also found in literature:

- Term, which can either be an object or a role to define a person or a thing (Hay & Healy, 2000; Object Management Group, 2013);
- Concept (Boyer & Mili, 2011; Object Management Group, 2013; Von Halle, 2001);
- Property of a concept (Von Halle, 2001);
- Subject (Von Halle, 2001);
- Entity (Von Halle, 2001);
- Attribute (Von Halle, 2001).

For this research, all previous listed alternatives found in the pattern catalogues and literature are referred to with the word *subject* as fundamental construct. The word *subject* is chosen since the majority of the other options, except term, are derived from specific fields like for example the database field. The choice to only include one fundamental construct to refer to these different levels of concepts is made in order to keep the amount of fundamental constructs of the CNL limited. In this way, the CNL will adhere to simplicity. To clarify the meaning of a subject further, the example business rule from above will be considered again. In this example, each subject in the conclusion and condition part is denoted by a blue border:

The tax amount of a taxpayer must be calculated as the sum of the salary of each current employment minus the tax rebate if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.

As can be concluded from this example, both the conclusion part as condition part can consist of multiple subjects. To make the business rule meaningful, it has to include at least one subject to reason about. This leads to the following grammar rules for the CNL:

- ✓ A Conclusion Part consists of **one or more** Subject(s);
- ✓ A Subject belongs to **exactly one** Conclusion Part;
- ✓ A Condition Part consists of **one or more** Subject(s);
- ✓ A Subject belongs to **exactly one** Condition Part.

The relationships and cardinalities between these fundamental constructs are shown in Figure 3.2.

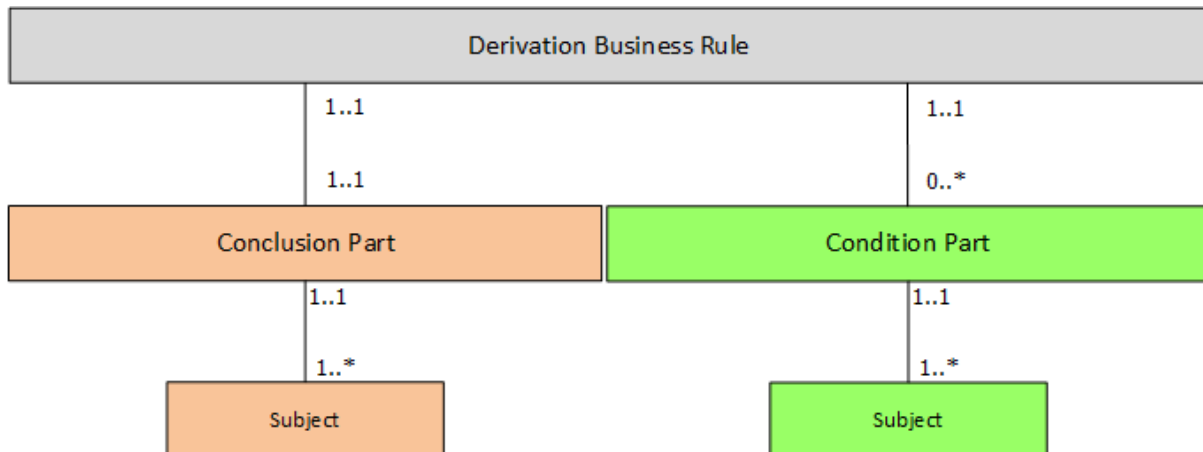


Figure 3.2: Relationships and cardinalities

### FUNDAMENTAL CONSTRUCT: QUANTIFIER

In two cases, a fundamental construct is found which denotes if a business rule involves: a specific subject (e.g. **the** subject), one subject (e.g. **a/an** subject) or more subjects (e.g. **each/every** subject). In the rule pattern catalogue of Morgan (2002), this fundamental construct is called a *determiner* and is expressed by <det>. Furthermore, in the business rule language SBVR this is called a *keyword*. For the CNL, this fundamental construct is also included but with another name namely *quantifier*. The reason to use a different name is because the term *keyword* is a bit vague. Furthermore, the word *determiner* is used in the English linguistics domain comprising many more instantiations (e.g. which, another, what) than the fundamental construct for this research can comprise (British Council, 2015). In linguistics, quantifiers are seen as a specific sub-group of determiners (British Council, 2015). The difference between the application of quantifiers in linguistics and this research is that for this research quantifiers also comprise articles (i.e. the, an, a). Another reason to choose the name ‘quantifier’ is because this concept is applied across the linguistics and logic domain bringing both domains together (Peters & Westerståhl, 2006).

To be more specific about the meaning of a quantifier, the example business rule from above will be considered again. In this example, each quantifier in the conclusion and condition part is denoted by a red border:

The tax amount of a taxpayer must be calculated as the sum of the salary of each current employment minus the tax rebate if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.

As can be concluded from this example, each quantifier has a direct association with one subject (see blue borders) and vice versa. The quantifier makes a business rule more precise and unambiguous (Object Management Group, 2013). In practice, not many business rule languages or rule pattern catalogues obligate this relation. However since precision is important for the CNL, this association will be included by the following grammar rules:

- ✓ A Subject is associated with **exactly one** Quantifier;
- ✓ A Quantifier is associated with **exactly one** Subject.

The relationships and cardinalities between these fundamental constructs are shown in Figure 3.3.

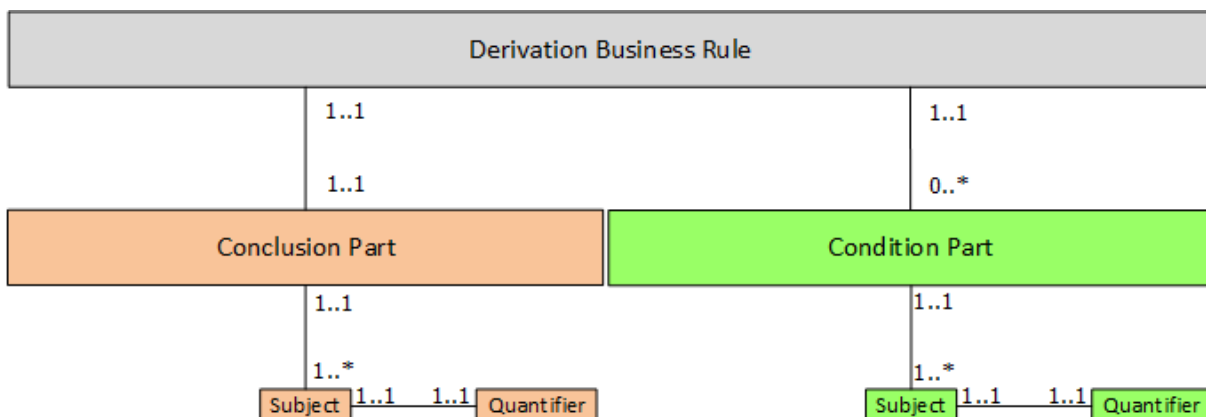


Figure 3.3: Relationships and cardinalities

## FUNDAMENTAL CONSTRUCT: RELATION

The choice to include *subject* as single fundamental construct in the CNL to refer to both “concepts (i.e. entities)” and “properties of concepts (i.e. attributes)”, can have a disadvantage. Although it keeps the CNL simple with regard to the amount of fundamental constructs, it can also make the business rule ambiguous. Therefore, some practitioners choose to use the common names and others use a kind of fundamental construct which addresses this disadvantage. This fundamental construct specifies the relation/association between subjects. By means of this relation, the different granularity levels between subjects can be made clear again. This relation is shown by means of a black border in the example below:

The tax amount of a taxpayer must be calculated as the sum of the salary of each current employment minus the tax rebate if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.

Below, the found examples in literature of a fundamental construct that specifies the relation/association between subjects will be discussed.

Hay and Healy (2000) call the fundamental construct a ‘*fact*’ and distinguish three types of facts: 1) a fact that associates a subject (i.e. business concept) with its ‘attribute’, 2) a fact that associates a subject as a generalization (i.e. supertype) of another subject, and 3) a fact that associates several subjects. Hay and Healy (2000) allow a lot of freedom in the usage of a fact since it can comprise a whole sentence and can cover multiple subjects instead of only a binary relation. For instance, the sentence “*a customer may request a model of car from a rental branch on a date*” is one fact according to the BRG that comprises four subjects: customer, car model, rental branch and date (Hay & Healy, 2000).

In addition, Von Halle (2001) defines a fact as “*a statement that connects terms, through prepositions and verb phrases, into sensible, business-relevant observations*”. She also provides a definition from a database point of view “*a fact is a relationship among entities or the association of an attribute to an entity*”.

Moreover, Von Halle and Goldberg (2009) propose the fundamental construct “*fact type*” which specifies the relation among a business concept and a property of this concept. An example of a fact type that Von Halle and Goldberg (2009) provide is “*Employment history of Person*”. So, it should be noted that this fundamental construct includes two subjects (i.e. Employment history and Person) and the relation.

Furthermore, Wan-Kadir and Loucopoulos (2004) describe a fact as “*a statement that asserts a relationship (attribute/ preposition / verb / generalization /aggregation) between two subjects*”. They state that this is often expressed in the form <subject> <relationship> [of] <subject>.

In contrast to previous examples, Object Management Group (2013) does not provide a fundamental construct to refer to a whole sentence part including the subjects and relationships. Object Management Group (2013) proposes “*(is) of*” as specific fundamental construct to specify the relationship between exactly two subjects (e.g. Engine size of car model).

For this research, a fundamental construct as explained above will be included for the CNL. The reason to include this fundamental construct is to be able to precisely specify the relation between subjects in a business rule, ensuring an unambiguous business rule set. In the context of this research, this fundamental construct will be called *relation* since it only specifies the relationship and will not refer to a whole sentence. With regard to the CNL, this fundamental construct may only be applied to show

that exactly two subjects are related (i.e. binary relationship). This corresponds to the following grammar rules:

- ✓ A Subject is associated with **zero or one** Relation;
- ✓ A Relation is associated with **exactly two** Subjects.

The relationships and cardinalities between these fundamental constructs are shown in Figure 3.4.

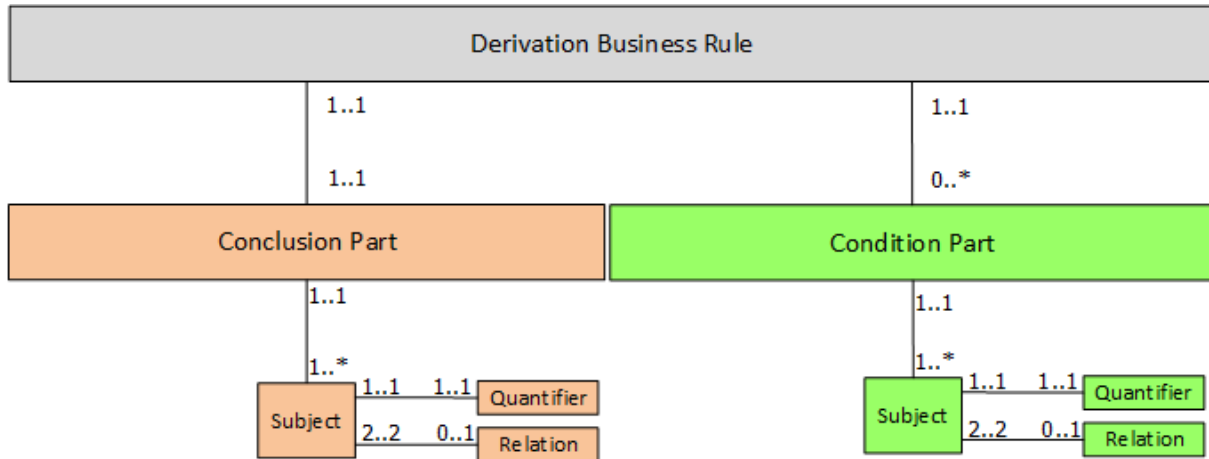


Figure 3.4: Relationships and cardinalities

## FUNDAMENTAL CONSTRUCT: EXPRESSION

Looking once again at the definition of a derivation business rule: “an expression that evaluates facts, by means of a calculation or classification, leading to a new fact (i.e. conclusion)” (Hay & Healy, 2000; Von Halle & Goldberg, 2009). Considering this definition, two statements can be made: 1) facts in a derivation business rule are evaluated by means of a calculation or classification, and 2) a new fact (i.e. conclusion) of a derivation business rule is either determined by a calculation or a classification. Both the calculation and the classification are seen as a separate fundamental construct of a derivation business rule, where the calculation is called a *ground* with regard to the CNL. The fundamental construct is called a ground since it has several underlying fundamental constructs, therefore the names computation or calculation are considered as too narrow. The substantiation for this view about both the fundamental construct ground and classification will be explained by means of an example. The example business rule from above is considered again, in which each occurring classification is depicted with a pink border and each occurring ground is depicted with a light blue border:

The tax amount of a taxpayer must be calculated as the sum of the salary of each current employment minus the tax rebate if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.

As can be seen in the example, the conclusion part (orange border) consists of one ground and the condition part (green border) consists of both a classification and a ground. By means of this example the following properties of the conclusion and condition part can be demonstrated:

- A conclusion part should include exactly one classification or ground:
  - a. without any of these two fundamental constructs (zero), no conclusion can be drawn about ‘the tax amount of a taxpayer’ and the business rule becomes meaningless;
  - b. it cannot include both fundamental constructs since it is logically not possible to say that ‘the tax amount of a taxpayer’ must be calculated and classified at the same time.
- A condition part should include one or more classifications or grounds:
  - a. without any of these two fundamental constructs (zero), a condition becomes meaningless. In this case, it must be checked if ‘the nationality of the taxpayer’ is equal to the classification Dutch. Besides this classification, it must be calculated if ‘the age of the taxpayer’ is higher than 18 (i.e. the ground).
  - b. it can include both fundamental constructs since a condition part can cover more than one condition and each condition should include one classification or ground. The inclusion of more conditions will be explained in more detail later.

This distinction between a ground and classification is supported by different sources. First of all, Hay and Healy (2000) consider two kinds of derivations: by a mathematical calculation (i.e. ground) or by an inference (i.e. classification). Secondly, Morgan (2002) also identifies these two fundamental constructs in his pattern catalogue only designates a ground by *<algorithm>*. Thirdly, in the patterns of Wan-Kadir and Loucopoulos (2004) also both fundamental constructs emerge. They include ground as *<algorithm>* and classification is covered by the more extensive fundamental construct *<fact>*. Lastly, the RuleSpeak pattern catalogue of Hoppenbrouwers (2011) uses both *STATE* and *TYPE* as fundamental construct for a classification and *COMP* as fundamental construct for a ground.



Taking the definition of a derivation business rule into account , “an expression that evaluates facts, by means of a calculation or classification, leading to a new fact (i.e. conclusion)” (Hay & Healy, 2000; Von Halle & Goldberg, 2009), both the ground and classification fundamental construct are seen as a specific type of *expression*. To clarify if a derivation business rule comprises a calculation expression or a classification expression, ‘expression’ is also included as a fundamental construct of the CNL. These premises correspond to the following grammar rules:

- ✓ A Conclusion Part consists of **exactly one** Expression;
- ✓ An Expression belongs to **exactly one** Conclusion Part;
- ✓ A Condition Part consists of **one or more** Expression(s);
- ✓ An Expression belongs to **exactly one** Condition Part;
- ✓ An Expression is **either** a Ground or a Classification.

The relationships and cardinalities between these fundamental constructs are shown in Figure 3.5.

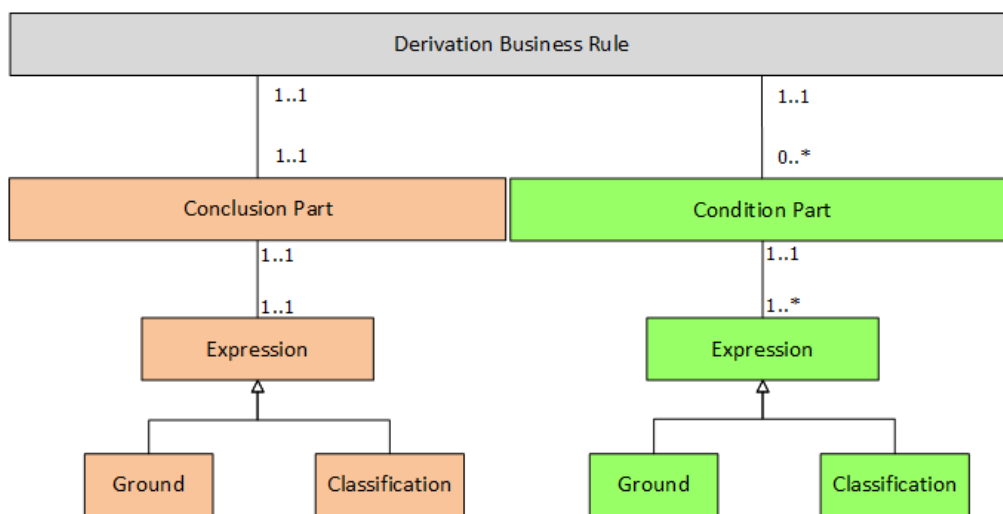


Figure 3.5: Relationships and cardinalities

### FUNDAMENTAL CONSTRUCT: CLASSIFICATION

Now, the fundamental construct *classification* will be considered into more detail by examining the following example business rules below:

- 1) The tax period must be equated to Dutch tax period....if (classification in a conclusion part);
- 2) The customer must be equated to gold-member...if (classification in a conclusion part);
- 3) The debt must be equated to 0 ....if (classification in a conclusion part).
- 4) If the tax period is equal to Dutch tax period (classification in a condition part);
- 5) If the customer is equal to gold-member (classification in a condition part);
- 6) If the debt is equal to 0 (classification in a condition part);
- 7) If the membership is equal to one of the following values: silver, gold, platinum (classification in a condition part).

The first three examples show that a classification, in the **conclusion part** of a derivation business rule, can equate a subject with:

- a. another subject (i.e. Dutch tax period);
- b. a value which can be for example a String, Date, Boolean, Number (i.e. gold-member, 0).

Examples four to seven show that a classification, in the **condition part** of a derivation business rule, can check the consistency between a subject and:

- a. another subject (i.e. Dutch tax period);
- b. a value which can be for example a String, Date, Boolean, Number (i.e. gold-member, 0).

Example seven shows that a business rule can also specify more than one option to choose from.

In short, a classification in a derivation business rule can:

- 1) **equate** a subject **with** another subject or a value – in the conclusion part;
- 2) or **check the consistency** between a subject and another subject or a value – in the condition part.

To be able to make the difference between the two classification options (i.e. *equate with* or *check the consistency*) clear, a fundamental construct will be included and made obligatory for the CNL. This fundamental construct will be called a *propositional operator*, which is underlined in the example business rules above. In the business rule language SBVR (Object Management Group, 2013), this is called an *instantiation formulation* which classifies things. From reviewing multiple different business rules from different sources, it became clear that instantiations of this propositional operator occur repeatedly. However, no overall name or individual fundamental construct could be found in the rule pattern catalogues.

A fundamental construct that did emerge from literature is *value*. Von Halle and Goldberg (2009) refer to a value by the word ‘fact’ or ‘fact value’. In the rule pattern catalogue of Von Halle (2001), <value> is included as fundamental construct to define some kind of value. Due to these sources, value is also seen as a separate fundamental construct. Altogether, this corresponds to the following grammar rules:

- ✓ A Classification consists of **exactly one** Propositional Operator;
- ✓ A Propositional Operator belongs to **exactly one** Classification;
- ✓ A Classification consists of **zero or more** Value(s);
- ✓ A Value belongs to **exactly one** Classification;
- ✓ A Classification consists of **zero or more** Subject(s);
- ✓ A Subject belongs to **exactly one** Classification;
- ✓ A Classification consists of **at least one** Value or of **at least one** Subject;

The relationships and cardinalities between these fundamental constructs are shown in Figure 3.6.

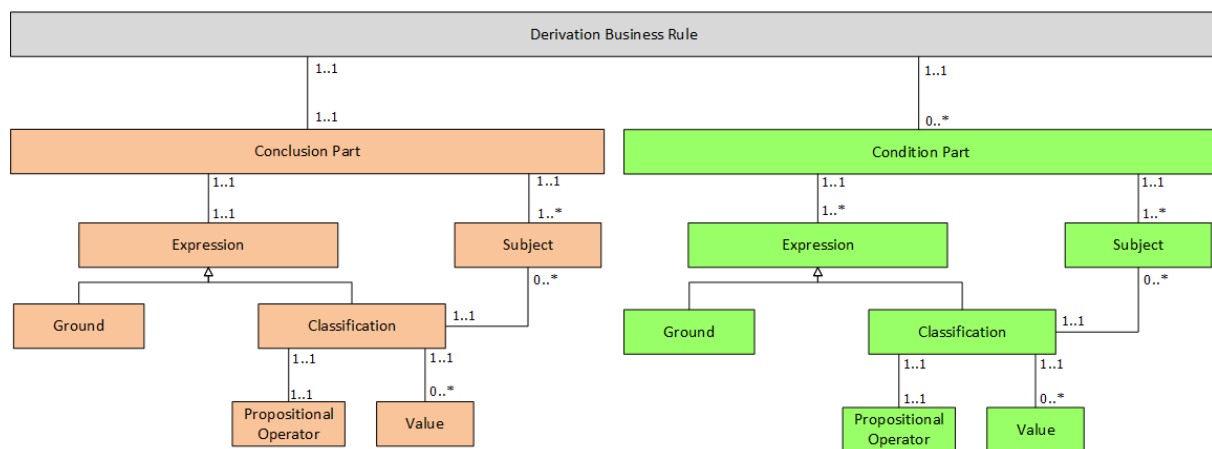


Figure 3.6: Relationships and cardinalities

## FUNDAMENTAL CONSTRUCT: GROUND

Now, the fundamental construct *ground* will be considered into more detail by examining the following example business rules below:

- 1) The tax amount must be calculated as the basic tax amount minus the tax rebate plus 5 percent ...if (basic ground in a conclusion part);
- 2) The tax amount must be calculated as the sum of the salary of each current employment ...if (mathematical function in a conclusion part);
- 3) If the tax amount is more than /less than/ more than or equal to/ less than or equal to the basic tax amount (ground in a condition part);
- 4) If the tax amount is more than /less than/ more than or equal to/ less than or equal to 0 (ground in a condition part);
- 5) If the tax amount is more than /less than/ more than or equal to/ less than or equal to the basic tax amount minus tax rebate (ground in a condition part);
- 6) If the tax amount is more than /less than/ more than or equal to/ less than or equal to the sum of the salary of each current employment (ground in a condition part).

The first two examples show that a ground, in the **conclusion part** of a derivation business rule, can equate a subject with:

- a. a basic ground (i.e. minus, plus, the sum of);

Examples three to six show that a ground, in the **condition part** of a derivation business rule, can compare a subject with:

- a. another subject (i.e. basic tax amount);
- b. a value (i.e. 0);
- c. a basic ground (i.e. minus, the sum of);

In short, a *ground* in a derivation business rule can either:

- 1) **equate** a subject **with** a basic ground – in the conclusion part;
- 2) or **compare** a subject **with** another subject, a value, or a basic ground – in the condition part.

To be able to make the difference between the two *ground* options (i.e. *equate with* and *compare with*) clear, a fundamental construct will be included and made obligatory for the CNL. This fundamental construct will be called a *mathematical operator*, which is underlined in the example business rules above. From reviewing multiple different business rules of different sources, it became clear that instantiations of this mathematical operator occur repeatedly. However, no overall name or individual fundamental construct could be found in the rule pattern catalogues or literature.

The mathematical operator, as fundamental construct in the CNL, also covers the following instantiations: +, -, /, \*. So, in case a derivation business rule equates or compares a subject with a basic ground, multiple mathematical operators have to be included (one to capture the equation or comparison and one or more to capture the other operators). To include a mathematical function in a business rule, an additional fundamental construct is incorporated in the CNL called *mathematical function*. In the examples above, only simple calculations are included. When more sophisticated calculation have to be made, the derivation business rule can include more than one mathematical function. Altogether, this corresponds to the following grammar rules:

- ✓ A Ground consists of **one or more** Mathematical Operator(s);
- ✓ A Mathematical Operator belongs to **exactly one** Ground;
- ✓ A Ground consists of **zero or more** Mathematical Function(s);
- ✓ A Mathematical Function belongs to **exactly one** Ground;
- ✓ A Ground consists of **zero or more** Value(s);
- ✓ A Value belongs to **exactly one** Ground;

- ✓ A Ground consists of **zero or more** Subject(s);
- ✓ A Subject belongs to **exactly one** Ground;
- ✓ A Ground consists of **at least one** Subject or of **at least one** Value.

The relationships and cardinalities between these fundamental constructs are shown in Figure 3.7.

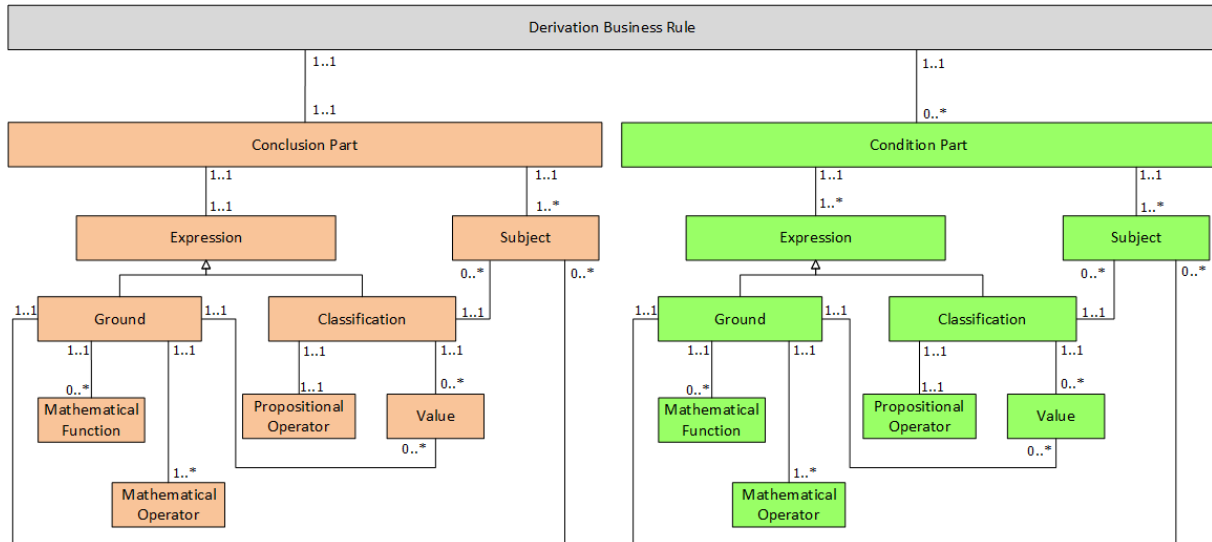


Figure 3.7: Relationships and cardinalities

## 3.2 Unique Fundamental Constructs Conclusion Part

After discussing the equivalent fundamental constructs for the conclusion and condition part, now the deviating fundamental constructs of both parts will be considered.

One fundamental construct is found that is only applicable for the conclusion part and not for the condition part. This fundamental construct determines how the derivation business rule is imposed. In other words, this fundamental construct defines the modality of the business rule. A *modality* can be considered as a type of proposition that asserts or denies the permissibility or obligation of some content (The Free Dictionary, 2015).

In all the seven business rule pattern catalogues, that are discussed in section 2.5, modality is included (Caron et al., 2013; do Prado Leite & Leonardi, 1998; Hoppenbrouwers, 2011; Morgan, 2002; Sangers-van Cappellen, 2014; Von Halle, 2001; Wan-Kadir & Loucopoulos, 2004). In most of the patterns, modality is included by means of providing one or more specific options that can be chosen. Examples of these modality options are: “*must*” to formulate an obligation or “*may*” to formulate permission. So, these pattern catalogues did not choose a common name to refer to the fundamental construct. In contrast, the business rule language SBVR uses the word *Keyword* or *modal operator* as umbrella to define the modality options (Object Management Group, 2013). Below, the example business rule is stated again to show the modality in a specific context. The modality is denoted by a purple border:

The tax amount of a taxpayer **must** be calculated as the sum of the salary of each current employment minus the tax rebate if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.

By explicitly specifying the modality of a business rule, the intention of the business rule becomes clearer for humans. For instance, does the business rule impose a requirement (i.e. *must*) or does the business rule impose an advice (i.e. *may*). However, excluding the modality will not change the logic of the business rule. When ‘*must*’ is excluded from the example business rule above, only the representation will change to: “*The tax amount of a taxpayer is calculated as the sum of the salary of each current employment minus the tax rebate, if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.*”

The choice has been made to include the modality as fundamental construct for the CNL, so that business rule authors have the choice to include it for enhancing the readability of the business rules for humans. To refer to this fundamental construct, the name *Modal Claim Type* is chosen. Taking previous premises into account, the following grammar rules can be established for the CNL:

- ✓ A Conclusion part consists of **zero or one** Modal Claim Type;
- ✓ A Modal Claim Type belongs to **exactly one** Conclusion Part.

The relationship and cardinalities between these fundamental constructs are shown in Figure 3.8.

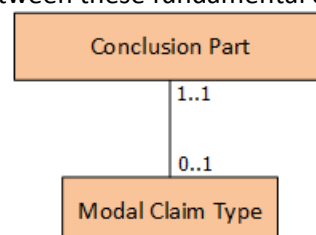


Figure 3.8: Relationships and cardinalities

### 3.3 Unique Fundamental Constructs Condition Part

Besides the *Modal Claim Type* that is included in the conclusion part, also two specific fundamental constructs are found for the condition part.

One of these fundamental constructs is used to indicate a condition part of the business rule which is repeatedly found in business rule catalogues or languages. Same as for the *Modal Claim Type*, most pattern catalogues only include specific instantiations for this fundamental construct and no overall name. For instance, Morgan (2002) includes the instantiations ‘if or unless’ to indicate the condition part. In his pattern catalogue, this is included with the following fundamental construct: (if | unless). Where the parentheses are used to show that it is one coherent pattern part (i.e. fundamental construct), and the vertical bar separates the two alternatives. Furthermore, in the catalogue of Wan-Kadir and Loucopoulos (2004) two alternatives for the fundamental construct are included in capitals as follows: IF and ONLY IF. Von Halle (2001) and Caron et al. (2013) have a very similar approach as the latter catalogue, they only include one option for the fundamental construct: if. Solely the RuleSpeak pattern catalogue of Hoppenbrouwers (2011) provides an overall name for such instantiations namely *keywords*, which covers the following three: if, when and only if. In computer science, or more specifically with regard to programming languages, the above provided instantiations are commonly referred to as *constructs* (2015). Therefore, the word ‘construct’ is adopted as name for the fundamental construct of the CNL.

The example business rule is used again to show the *construct* in a specific business rule, see the yellow border:

The tax amount of a taxpayer must be calculated as the sum of the salary of each current employment minus the tax rebate if the nationality of the taxpayer is Dutch and the age of the taxpayer is higher than 18.

This example includes only one condition part, which results in one *construct*. However, considering the relationship and cardinalities between a derivation business rule and the condition part again (see Figure 3.1), a derivation business rule can include several condition parts. The choice has been made to make the *construct* obligatory in relation to a condition part. In this way, a clear separation between condition parts can be ensured by using the CNL. This corresponds to the following grammar rules:

- ✓ A Condition Part consists of **one or more** Construct(s);
- ✓ A Construct belongs to **exactly one** Condition Part.

The relationship and cardinalities between these fundamental constructs are shown in Figure 3.9.

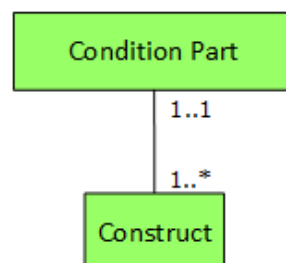


Figure 3.9: Relationships and cardinalities

The above provided example shows that a business rule can include one condition part covering multiple conditions, namely: 1) the nationality of the taxpayer is Dutch, and 2) the age of the taxpayer is higher than 18. In this case, both conditions have to be met (i.e. be true) in order for the 'tax amount of a taxpayer' to be calculated in that particular way. This becomes clear by means of the word 'and' between both conditions (see grey border). However, sometimes multiple conditions are formulated in a business rule of which only one has to be met, or a few of them, or maximal one.

So in case a derivation business rule includes more than one condition, the connection between these conditions has to be made clear. From the example above, but also from the reviewed business rules catalogues, it emerged that this can be done by using an additional fundament construct. Similar to the example above, Von Halle (2001) specifies the fundamental construct AND in her patterns to connect conditions in a binary way. By using the patterns of Von Halle, the fundamental construct will be included multiple times when specifying more than two conditions. This last observation is not the case when using the patterns of Morgan (2002). He applies the fundamental construct as follows: at least  $\langle m \rangle$  [and not more than  $\langle n \rangle$ ] of the following is true. Thus, when more than two conditions are included it is not necessary to include the fundamental construct two times.

Besides mentioned pattern catalogues, SBVR also specifies the connection between conditions and calls this fundamental part a *logical operation* (Object Management Group, 2013). The SBVR language provides more options than the rule catalogues to choose from, it includes the following *logical operations*: Conjunction ( $p$  and  $q$ ), Disjunction ( $p$  or  $q$ ), Exclusive disjunction ( $p$  or  $q$  but not both), Nand formulation (not both  $p$  and  $q$ ), Nor formulation (neither  $p$  nor  $q$ ), and the Whether-or-not formulation ( $p$  whether or not  $q$ ). The  $p$  and  $q$  correspond to conditions (Object Management Group, 2013).

Since the fundamental construct connects two or more conditions, the choice has been made to call it a *connective* with regard to the CNL. This fundamental construct is only necessary when more than one condition is included in the condition part. Furthermore, the way in which the *connective* is used will not be imposed by the grammar rules of the CNL. Both ways as described above are allowed. So, either the fundamental construct can be used once in a business rule which specifies the relation between several conditions. Or using the fundamental construct multiple times, between each condition which corresponds to a binary relation. This results in the following grammar rules:

- ✓ A Condition Part consists of **zero or more** Connective(s);
- ✓ A Connective belongs to **exactly one** Condition Part;
- ✓ A Connective must be included to connect **two or more** Conditions.

In the example business rule on the previous page, the *connective* is denoted by a grey border. The relationship and cardinalities between the *connective* and *condition part* are shown in Figure 3.10.

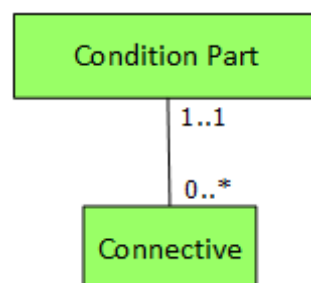


Figure 3.10: Relationships and cardinalities



### **3.4 Summary**

In previous sections, all fundamental constructs of the CNL to specify a derivation business rule are indicated and described along with the corresponding grammar rules. Here, these fifteen fundamental constructs will be listed again for clarity reasons: Conclusion part, Condition part, Modal Claim Type, Construct, Connective, Expression, Subject, Quantifier, Relation, Ground, Classification, Propositional Operator, Value, Mathematical Operator, and Mathematical Function. In Appendix 3, all grammar rules of the formal grammar underlying the CNL are listed.

By aggregating the fundamental constructs and their interrelationships, the whole meta-model of the CNL is created. This meta-model is shown in Figure 3.11 on the next page.



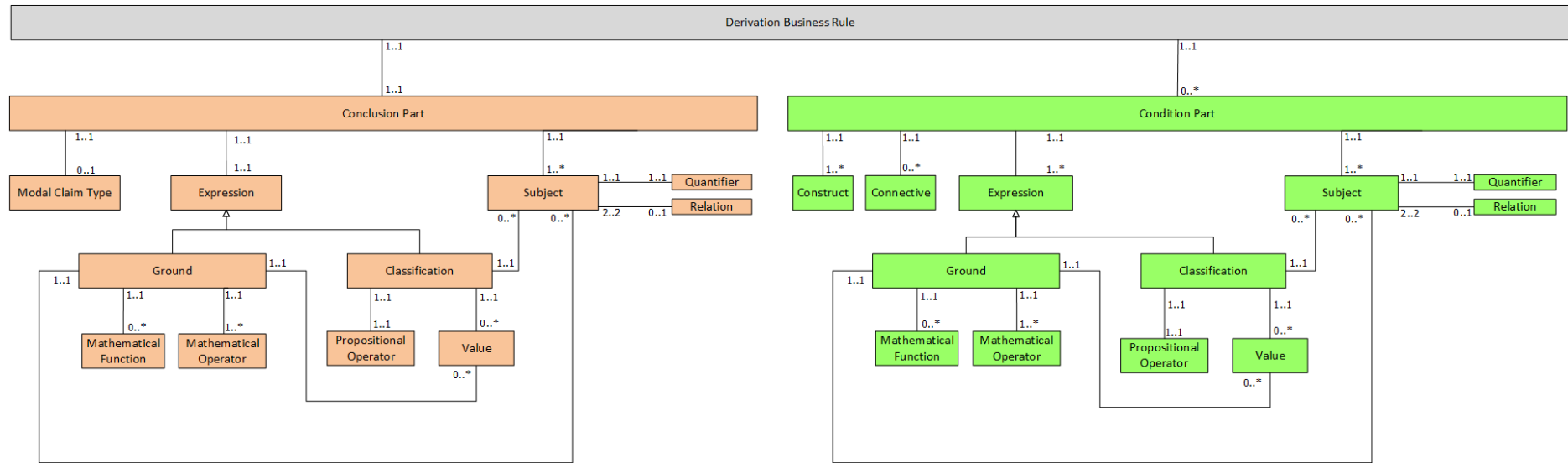


Figure 3.11: Meta-model

## 4 Preliminary Validation

In previous chapter, the fundamental constructs to specify derivation business rules were identified. This chapter describes the validation process of the fundamental constructs, which is done prior to the pattern set creation, in order to ensure that the patterns will be composed of necessary building blocks. In this way, the fundamental constructs are the unit of analysis of the validation and serve as dependent variable in the experiments. Three validation rounds are performed, each from a different point of view and using a different independent variable to analyze the fundamental constructs (see Figure 4.1):

1. from a pattern level view using existing pattern catalogues as independent variable;
2. from an instance level view using specified business rule sets as independent variable;
3. from an implementation dependent level view using business rules management systems as independent variable.



Figure 4.1: Validation Point of Views

For all three of these validation rounds, the data collection and data analysis process are explained below in two separate sub-sections: data collection and data analysis.

### 4.1 Data Collection

Below, the data collection process per validation round is described including the applied sampling strategy and selection criteria.

#### 4.1.1 Validation Round 1 – Pattern Level View

For the first validation round, the pattern level view, current existing business rule pattern catalogues were collected by means of a broad literature search (see section 2.5). These pattern catalogues were collected to map the existing patterns onto the identified fundamental constructs of this research. This mapping was performed in order to investigate if all the existing patterns could be captured with the identified fundamental constructs of the envisioned CNL. In this way, this first validation round could already indicate on a high level if essential fundamental constructs were absent. When this validation round would demonstrate that all the patterns of the existing business rule pattern catalogues could be mapped, it would also mean that business rules specified with these patterns could be captured with the fundamental constructs of the CNL.

The first practical selection criterion to collect the pattern catalogues corresponds to *access*. Access to the pattern catalogues was required to use them for this research. Besides this practical selection criterion, also two theoretical selection criteria are applied. The first theoretical criterion corresponds to *relevance*. As described in section 2.5, seven business rule pattern catalogues were found which are



relevant for this research. These catalogues are relevant for two reasons: they are established for the BRM domain and they are focused on specifying business rules. Based on this second criterion, the following seven business rule pattern catalogues were selected: 1) do Prado Leite and Leonardi (1998), 2) Morgan (2002), 3) RuleSpeak of Hoppenbrouwers (2011), 4) Wan-Kadir and Loucopoulos (2004), 5) Von Halle (2001), 6) Caron et al. (2013), and 7) RegelSpraak of Sangers-van Cappellen (2014). Due to the last applied theoretical selection criterion *representativeness*, two pattern catalogues (number one and six) were excluded from the validation since they did not include any pattern to specify derivation business rules.

As a result, five pattern catalogues were used for the first validation round. In total, these five remaining catalogues comprised 66 patterns of which 29 patterns were discarded. Similar to the exclusion reason of the pattern catalogues, these 29 patterns were not targeted at capturing the correct type of business rules (i.e. derivation business rules). Eventually, this resulted in the use of 37 patterns for the validation.

#### **4.1.2 Validation Round 2 – Instance Level View**

For the second validation round, the instance level view, a set of business rules was collected. This business rule set has been composed of business rules which were randomly selected from different business rule cases originated from both literature and practice. This sampling strategy was followed in order to cover a wide range of domains where business rules are applied. During this second validation round, these collected business rules were mapped onto the identified fundamental constructs of the CNL to investigate if specific instantiations could be captured by the fundamental constructs.

The business rule set for the validation has been composed of business rules that derive from eleven different cases. To select these business rules, the theoretical criteria *relevancy* and *representativeness* were applied. In order to comply with these two criteria, only derivation business rules were included in the set. Besides these two theoretical criteria, also a practical selection criterion was applied namely *access*. Access to the business rule cases was required.

Five of the cases correspond to a pattern catalogue from the first validation round, namely: Morgan (2002), Wan-Kadir and Loucopoulos (2004), Von Halle (2001), RegelSpraak of Sangers-van Cappellen (2014), and RuleSpeak of Hoppenbrouwers (2011). However, in this second validation round not the patterns were used for the mapping but the business rules they provided. The choice has been made to include the sources again as an additional validation to increase the reliability of the validation (Lee & Baskerville, 2003). One remark has to be made, RuleSpeak of Hoppenbrouwers (2011) did not provide business rules along with the patterns. Therefore, a business rule set defined with RuleSpeak from an anonymous Dutch government organization was utilized.

In contrast to the business rules from the pattern catalogues, also six additional business rule cases were gathered to be able to further generalize the outcome of the validation. One of these cases correspond to a business rule case study which was published online by the Decision Management Community (DM Community, 2015) for public use, namely a business rule set to determine the risk of meeting a werewolf. Moreover, one business rule set to assess the risk for diabetic patients is derived from (Parish, 2014) and one business rule set that deals with vehicle insurances called “UServ Product Derby” is derived from Building Business Capability (BBC, 2015). Two other cases are derived from (Feldman, 2011, 2014): 1) a business rule set to determine the required therapy for a patient, and 2) a business rule set to calculate tax returns. Lastly, a case from an anonymous Dutch government organization was utilized which is a business rule set to determine the eligibility to be au pair.

In total, the eleven cases contained 313 business rules which differed from each other in terms of their type (e.g. process rules, calculation rules, structural rules). Therefore, the business rules were first evaluated and sorted by type. Only the derivation business rules were eligible. From the remaining set of derivation business rules, a random selection of 150 business rules is made due to time constraints.

#### **4.1.3 Validation Round 3 – Implementation Dependent Level View**

For the third validation round, the implementation dependent level view is taken into account by means of analyzing business rules in an implementation dependent environment. This directly results in the first theoretical criterion to select these business rules, namely that they were *implemented in a specific Business Rules Management System (BRMS)*. The second theoretical selection criterion corresponded to the fact that the business rule set was *representative*, and the third that it was a *relevant* business rule set. Besides these three theoretical criteria, also two practical selection criteria are applied. Firstly, *access* to the BRM systems or documentation about these business rule set implementations was needed. Secondly, it was required that the business rule set (i.e. use case) was identical for each implementation in a different BRMS.

Based on these theoretical and practical criteria, the use case called “UServ Product Derby” derived from the Building Business Capability (BBC, 2015) was selected. This use case was published online on the Decision Management Community website (DM Community, 2015). Every month, the Decision Management Community posts a decision modelling challenge on their website and they invite practitioners from all enterprise levels (e.g. business analysts, developers, technology vendors, consultants) to share their solutions. In other words, these practitioners show how the business rules from the use case can be implemented by using their BRMS. The online availability aspect triggers the practitioners to deliver high quality documentation in order to preserve their reputation. This ensured that the documentation about the use case implementations complied to the *representativeness* criterion. Furthermore, the use case was *relevant* as it dealt with insurance issues and the insurance industry is an industry which process a high volume of business rules. Moreover, the use case was *representative* for this research as it comprised 69 derivation business rules from the total amount of business rules.

With regard to the “UServ” use case, six solutions were submitted on the Decision Management Community website by the following BRMS vendors: 1) Blueriq (Schadd, 2015), 2) Corticon (Parish, 2015), 3) IBM ODM (Ortiguela, 2015), 4) Sapiens (Segal, 2015), 5) OpenRules (Feldman, 2015), and 6) OpenL Tablets (Bastun, 2015). The documentation of the UServ business rule set implementations into the six BRM systems is used as input for validation round three. If this documentation was not sufficient enough to perform the mapping of the implementation components onto the fundamental constructs, manuals or the actual BRMSs were analyzed.

## 4.2 Data Analysis

Section 4.1 stated which input data was collected for each validation round, from where this data was collected, how this data was selected, and for which reasons. This section will explain how this data is analyzed by describing the ‘overall data analysis process’ and applied method for the entire preliminary validation. Subsequently, the specific process of conducting each validation round and the results thereof are described.

### 4.2.1 Overall Data Analysis Process and Method

During this research, the collected data is analyzed by means of nominal comparisons across cases. Mahoney (1999) states that “*nominal comparison involves the use of categories that are mutually exclusive to locate the causes of an outcome*”. Methods that are based on nominal comparison are Mill’s methods (Mahoney, 1999; Mill, 1906). One of the five Mill’s methods (Mill, 1906), “*The Joint Method of Agreement and Difference*”, is applied as the overall data analysis method for the preliminary validation of this research. The intention of this method is to identify similarities and differences between cases (Mahoney, 1999). These observations can then be analyzed and interpreted to draw conclusions (Mahoney, 1999). In general, Mill’s methods are used to draw conclusions about causal relationships by analyzing the data (i.e. effects) and find a common denominator (i.e. cause) (Mill, 1906). However, for this research the method is applied in a kind of reversed way. It is not applied to find causal relationships but to validate the already drawn conclusion of which fundamental constructs are the common denominators required to specify derivation business rules. This conclusion was drawn by means of analyzing literature as described in Chapter 3.

More specifically, Mill’s method is used during the validation to conduct the mappings of the data onto the identified fundamental constructs as mentioned in section 4.1. Mapping means that each data item in a data set (e.g. a single business rule, pattern or implementation component) was disassembled in smaller parts (if necessary) and these parts were tried to match onto a fundamental construct. A match was indicated by a cell filled with a corresponding data item part, and when a fundamental construct was not found the cell remained empty. In this way, the number of similarities and differences could be identified efficiently. Table 4.1 shows an excerpt of how this mapping was done, where the grey cells correspond to fundamental constructs and the blue cells to data item parts (i.e. business rule parts).

The mapping showed which fundamental constructs occurred in each data set. As a result, an indication about the minimal set of fundamental constructs required to specify derivation business rules emerged. In addition, the mapping could indicate that a specific fundamental construct may be superfluous to include (i.e. empty cells). From this validation, it became clear which fundamental constructs are important and which are not necessary to include in the CNL. In this way, the application of the Mill’s method could provide a further substantiation of including the identified fundamental constructs in the CNL besides the already found support from literature.

	<b>Quantifier</b>	<b>Subject</b>	<b>Relation</b>	<b>Modal Claim Type</b>
<b>A customer must...</b>	A	customer		must
<b>The member’s volume discount amount...</b>	The	member	’s	
		volume discount amount		

**Table 4.1:** Example Mapping with Mill’s Method

The next three *result* sub-sections are organized as follows. First, the eligible check of the selected data is described into more detail compared to the *data collection* section. Then, the results of the mapping by means of the selected Mill’s method are provided.

#### 4.2.2 Results Validation Round 1 – Pattern Level View

To recall, five pattern catalogues were selected for the first validation round by means of the data collection process. These five pattern catalogues were: Morgan (2002), RuleSpeak of Hoppenbrouwers (2011), Wan-Kadir and Loucopoulos (2004), Von Halle (2001), and RegelSprak of Sangers-van Cappellen (2014). Altogether, these catalogues provided 66 patterns of which 29 patterns were out of scope due to the fact that they were not devised for derivation business rules. As a result, 37 patterns were used for the validation round (See Table 4.2).

Case	Total Amount of Patterns Available	Amount Out of scope	Amount In scope
Morgan	5	3	2
RuleSpeak	18	6	12
Wan Kadir & Loucopoulos	5	3	2
Von Halle	5	3	2
RegelSprak	33	14	19
<b>TOTAL</b>	<b>66</b>	<b>29</b>	<b>37</b>

Table 4.2: Figures Data Collection Process for Validation Round 1

To decide which patterns were eligible to use for the validation, the patterns were analyzed to investigate if they were established to capture derivation business rules. As explained in section 2.2, the aim of derivation business rules is to create new information (i.e. new fact) by means of a classification or calculation. In the rest of this subsection, the stated figures in Table 4.2 will be considered into more detail per catalogue.

##### **PATTERN CATALOGUE: MORGAN**

Morgan (2002) provides five patterns in his catalogue: 1) Basic Constraint, 2) List Constraint, 3) Classification, 4) Computation, and 5) Enumeration. These patterns were checked for eligibility to be used for the validation. This resulted in omitting three patterns since they were established to define business rules that constrain a subject on behalf of a business event. With regard to the Basic Constraint and the List Constraint, this aim is reflected in the pattern by the inclusion of the fundamental construct <characteristic>. This fundamental construct implies “*the business behavior that must take place or a relationship that must be enforced*” (Morgan, 2002). With regard to the Enumeration pattern, the constraining part is indicated by the following fundamental parts: ‘*must be chosen from the following [ open | closed ] enumeration*’ and <enum-list>. These parts prescribe the range of values that a subject can adopt. In this way, these three patterns capture business rules that constrain information and correspond to the excluded Mandatory Constraint category of Von Halle (2001). Therefore, these three patterns were out of scope for this research (see Table 4.3). In contrast, the Classification and Computation pattern lie within the scope of this research, since they are applicable for specifying derivation business rules. As a result, these two pattern were eligible for the validation (see Table 4.3).

Case	Total Amount of Patterns Available	Amount Out of scope	Amount In scope
Morgan	5	3	2

Table 4.3: Figures of Eligibility Check - Pattern Catalogue Morgan

### **PATTERN CATALOGUE: RULESPEAK**

The RuleSpeak pattern catalogue of Hoppenbrouwers (2011) comprised 18 patterns. In contrast to Morgan (2002), these patterns were not classified or given a specific name. Before the 18 patterns were used for the validation, they were also analyzed for eligibility which resulted in the exclusion of six patterns. Two of these six patterns included the fundamental constructs ‘*must*’, ‘*be performed*’, ‘*when*’, indicating that both patterns are used to specify business rules that enable other action on behalf of the business event. Such business rules can be classified as Action Enablers according to the classification scheme of Von Halle (2001), which are out of scope for this research. The other four out of six discarded patterns included the fundamental construct ‘*May/Need not*’ meaning that business rules specified with these patterns are not compulsory. These business rules are covered by the Guideline business rule category of Von Halle (2001), that is also out of scope. In summary, the other twelve patterns were eligible for the validation (See Table 4.4).

Case	Total Amount of Patterns Available	Amount Out of scope	Amount In scope
RuleSpeak	18	6	12

Table 4.4: Figures of Eligibility Check - Pattern Catalogue RuleSpeak

### **PATTERN CATALOGUE: WAN-KADIR & LOUCOPOULOS**

Wan-Kadir and Loucopoulos (2004) created a catalogue with the following five patterns: 1) Mandatory Constraint, 2) Guideline, 3) Action Assertion, 4) Computation, and 5) Inference. The first two patterns are focused on specifying business rules that constrain information, and the third pattern aims at capturing business rules that enable action. So, the first three patterns are similar to the three excluded categories of Von Halle’s classification (2001) (i.e. Mandatory Constraint, Guideline, and Action Enabler). Therefore, these three patterns were not eligible to use for the validation. On the other hand, the latter two patterns are established to capture business rules that create new information by means of a calculation or classification (i.e. derivation business rules). So, these two patterns were incorporated in the validation (See Table 4.5).

Case	Total Amount of Patterns Available	Amount Out of scope	Amount In scope
Wan-Kadir & Loucopoulos	5	3	2

Table 4.5: Figures of Eligibility Check - Pattern Catalogue Wan-Kadir & Loucopoulos

### **PATTERN CATALOGUE: VON HALLE**

Von Halle (2001) established five patterns, one for each of the business rule categories she defines in her classification scheme: 1) Mandatory Constraint, 2) Guideline, 3) Action Enabler, 4) Computation, and 5) Inference. As described in section 2.2, the first three patterns are established for specifying a different type of business rule than derivation business rules and are therefore out of scope for this research. So, only the latter two patterns are considered as eligible for the validation (See Table 4.6).

Case	Total Amount of Patterns Available	Amount Out of scope	Amount In scope
Von Halle	5	3	2

Table 4.6: Figures of Eligibility Check - Pattern Catalogue Von Halle

### **PATTERN CATALOGUE: REGELSPRAAK**

RegelSprak of Sangers-van Cappellen (2014) has 31 patterns in total which are divided into patterns for each of the following rule categories: 1) Decision Rules, 2) Calculation Rules, 3) Value Range Rules, 4) Consistency Control Rules, 5) Rounding Rules, 6) Process Rules, and 7) the If-part of Rules. Each category consists of a different amount of patterns.

Three of the seven categories were considered as out of scope. As first, the Value Range Rules category including one pattern was not eligible. Given the fact that the aim of this pattern is to capture business rules that constrain the range of values that an attribute can have. Secondly, the eleven patterns for specifying Consistency Control Rules were discarded. These business rules also constrain the value of an attribute, but this time by means of verifying if the value of this attribute is consistent with a predetermined value. Taking these statements into account, both categories are similar to the excluded Mandatory Constrain category of Von Halle (2001). The third omitted category, Process Rules, included two patterns which are used to define business rules that prescribe the order in which other rules should be executed. These patterns are also out of scope as they resemble the Action Enabler Rules of Von Halle (2001). So, 14 patterns were excluded in total.

The other four categories were considered as in scope. Firstly, two Decision Rule patterns were provided to specify business rules aimed at determining the value of an attribute (result) by equating it with another value. So, no calculation is made and the result is not numerical but categorical. Secondly, eight Calculation Rule patterns were included that are also applicable for business rules that determine the value of an attribute but in this case by means of a calculation. So, both Decision and Calculation patterns capture business rules that create new information on behalf of the business event. Thirdly, two Rounding Rule patterns are included which are associated with the Calculation Rule patterns since they round the calculated value of an attribute. Lastly, seven patterns were available that are used to define the IF-part of a business rule. These patterns were not established for one specific type of business rule and were formulated on a very high-level, making them eligible to specify the condition part of derivation business rules. Eventually, 19 patterns were eligible and used for the validation (See Table 4.7).

Case	Total Amount of Patterns Available	Amount Out of scope	Amount In scope
RegelSprak	33	14	19

**Table 4.7:** Figures of Eligibility Check - Pattern Catalogue RegelSprak



## RESULTS

After the eligible check of all the collected data was performed, the 37 patterns in scope were mapped by means of the Mill's method. This entire mapping can be found in Appendix 5. On page 67, two examples of this mapping are provided to give an impression. The examples are divided over two different tables for readability reasons. Table 4.9 shows the mapping of the patterns onto the Conclusion part, and Table 4.10 shows the mapping onto the Condition part.

The mapping showed that the overall granularity level of the building blocks of the existing patterns was lower than the fundamental constructs of the CNL. In other words, the level of detail of the patterns was a lot lower; a building block of an existing pattern represented a much larger part of a business rule than a fundamental construct. As a result, the building blocks (i.e. pattern parts) of the existing patterns did not always corresponded directly to a fundamental construct. Therefore, sometimes the same building block was repeatedly mapped onto different fundamental constructs.

To make this finding more clear, consider the example in Table 4.8 below. The left column shows the computation pattern of Morgan (2002) and the right column shows a business rule, provided by Morgan (2002), defined with this pattern. From this example, it became clear that Morgan (2002) uses the building block **<algorithm>** to capture the entire calculation sentence part “**total item value plus sales tax**”. In contrast, when using the CNL to define a calculation part, six different fundamental constructs are established namely: Mathematical Operator, Mathematical Function, Value, Quantifier, Subject, and Relation. Altogether, these six fundamental constructs represent the higher level *ground* fundamental construct. Therefore, when mapping this pattern of Morgan, the **<algorithm>** building block was mapped onto all different fundamental constructs in the ground part. This is denoted in a cell by “*falls under <algorithm>*” (see row nr. 1 in Table 4.9 on page 67), to indicate that these fundamental constructs were found but are represented by Morgan in a different way.

Computation pattern Morgan (2002)	Instantiation of pattern
<b>&lt;det&gt;</b> <b>&lt;result&gt;</b> <i>is defined as</i> <b>&lt;algorithm&gt;</b>	<i>The total sale value is defined as total item value plus sales tax.</i>

**Table 4.8:** Example Granularity Level Building Blocks of Existing Pattern

Considering the mapping of the remaining pattern parts of Morgan's computation pattern (2002), **<det>** corresponds directly to the fundamental construct *Quantifier* (i.e. **the**). Furthermore, **<result>** corresponds directly to the fundamental construct *Subject* (i.e. **total sale value**). Moreover, Morgan states that he uses *is defined as* instead of the imperative “must be computed as” to avoid a very procedural style. As a result, this pattern part corresponds to two separate fundamental constructs. In terms of the CNL, **is** corresponds to the fundamental construct *Modal Claim Type* (i.e. **must**), and **defined as** to the fundamental construct *Mathematical Operator* (i.e. **be computed as**). The cells of the other fundamental constructs in the conclusion part remained empty, indicated in the example by a dash (i.e. -). Since this pattern is solely targeted at specifying the conclusion part of a derivation business rule, nothing could be mapped onto the fundamental constructs of the condition part. To view how this pattern of Morgan was mapped, see row 1 of Table 4.9 on page 67.

The second example (see row nr. 2 in Table 4.9) corresponds to the mapping of the following derivation pattern from RegelSprak:

De/het ...attribuut... van een (voorkomen van een ...object... van een) ...object/rol... moet gesteld worden op ja/nee.

In this pattern, the fundamental construct *Quantifier* is included by different instantiations instead of a placeholder: De/het (i.e. the) and een (i.e. a/an). For the fundamental construct *Subject*, multiple placeholders are incorporated: ...attribuut... (i.e. attribute), ...object... (i.e. object), and ...object/rol... (i.e. object/role). The pattern parts van (i.e. of) and voorkomen van (i.e. occurrence of) are equal to the fundamental construct *Relation*. Furthermore, the pattern part moet (i.e. must) corresponds to the fundamental construct *Modal Claim Type* and gesteld worden op (i.e. be equated to) corresponds to the *Propositional Operator*. Lastly, the pattern part ja/nee indicates two different options to choose from as instantiation of the fundamental construct *Value*. To view how this pattern of RegelSprak was mapped, see row 2 of Table 4.9 on the next page.

As already described earlier, the RegelSprak pattern catalogue includes individual patterns solely targeted at specifying the condition part. The following condition pattern is mapped in Table 4.10 to provide an example:

Indien hij / een/ het/ elk voorkomen van zijn ...object... aan alle volgende voorwaarden voldoet: ...

In this pattern, the pattern part Indien (i.e. If) is equal to the fundamental construct *Construct*. Same as for the conclusion part: the orange pattern parts correspond to the fundamental construct *Quantifier*, the green pattern parts to *Subject*, and pink pattern parts to *Relation*. Moreover, the pattern part aan alle volgende voorwaarden voldoet: corresponds to the fundamental construct *Connective*. As became clear from the provided example of RegelSprak along with this pattern, the pattern part “...” is used as placeholder to specify the entire *Expression* part of the conditions. Therefore, the cells of all the fundamental constructs related to the *Expression* part were filled with *Falls under “...”*. The forward slash (i.e. /) in this pattern separates the pattern parts which are mutually exclusive. To view how this pattern of RegelSprak was mapped, see row 2 of Table 4.10 on the next page.



Derivation Business Rule															
Conclusion Part															
Quantifier	Subject	Relation	Modal Claim Type	Expression											
				Classification					Ground						
				Propositional Operator	Value	Quantifier	Subject	Relation	Mathematical Operator	Mathematical Function	Value	Quantifier	Subject	Relation	
1	<det>	<subject>	-	is	-	-	-	-	-	defined as	Falls under <algorithm>	Falls under <algorithm>	Falls under <algorithm>	Falls under <algorithm>	Falls under <algorithm>
										Falls under <algorithm>					
2	De / het	attribuut	van	moet	gesteld worden op	ja/nee..	-	-	-	-	-	-	-	-	-
	een	object/ rol	(voorkomen van)												
	(een)	(object)	(van)												
	(een)														

Table 4.9: Example Mapping of Patterns on Conclusion Part

Condition Part															
Construct	Quantifier	Subject	Relation	Connective	Expression										
					Classification					Ground					
					Propositional Operator	Value	Quantifier	Subject	Relation	Mathematical Operator	Mathematical Function	Value	Quantifier	Subject	Relation
1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
2	Indien		hij /		aan alle volgende voorwaarden en voldoet:	Falls under "..."	Falls under "..."	Falls under "..."	Falls under "..."	Falls under "..."	Falls under "..."	Falls under "..."	Falls under "..."	Falls under "..."	Falls under "..."
		een/het/ elk	voorkomen van												
		zijn	object												

Table 4.10: Example Mapping of Patterns on Condition Part



The complete mapping showed that it was possible to map all 37 patterns, but some cells remained empty as can be seen in the previous examples. These empty cells were further investigated, because they could indicate that the establishment of the fundamental constructs for the CNL was not correct. For instance, that a fundamental construct is superfluous.

To figure out why these cells remained empty, the mapping was analyzed from which two overall reasons emerged. Firstly, some entire high level fundamental constructs with their underlying fundamental constructs remained empty. In case a pattern intended for a *ground* business rule was mapped all cells of the fundamental constructs related to a *classification* remained empty. This is shown by a thick border in example pattern 1, mapped in Table 4.9. The other way around, mapping a *classification* business rule pattern resulted in empty cells for the fundamental constructs related to a *ground* business rule. This is shown in example pattern 2, mapped in Table 4.9. In addition, some patterns were solely established for the conclusion part omitting the condition part and vice versa. As a result, all fundamental constructs associated with one of these main parts remained empty (see row nr 1 of pattern 1 in Table 4.10). This first finding shows high cohesiveness between specific fundamental constructs, which provides substantiation for the established meta-model in which the fundamental constructs are subdivided and connected. Furthermore, this finding provides a logical explanation for the fact that many cells remained empty and contradicts the assumption that these fundamental constructs could be superfluous.

The second overall finding was that for some patterns, individual fundamental constructs remained empty. For these empty cells, no direct explanation could be found in contrast to the ones as described above. To obtain insight into these individual empty cells, the amount of empty cells per fundamental construct was calculated. Table 4.11 shows the result of this calculation, where the figures should be placed in context by considering the total amount of filled cells which is 37.

Fundamental Constructs		Amount of empty cells
Quantifier	(Conclusion Part)	0 / 37
Subject	(Conclusion Part)	0 / 37
Relation	(Conclusion Part)	2 / 37
Modal Claim Type	(Conclusion Part)	3 / 37
Propositional Operator	(Conclusion Part - Classification)	0 / 37
Value	(Conclusion Part - Classification)	4 / 37
Quantifier	(Conclusion Part - Classification)	3 / 37
Subject	(Conclusion Part - Classification)	2 / 37
Relation	(Conclusion Part - Classification)	7 / 37
Mathematical Operator	(Conclusion Part - Ground)	0 / 37
Mathematical Function	(Conclusion Part - Ground)	1 / 37
Value	(Conclusion Part - Ground)	4 / 37
Quantifier	(Conclusion Part - Ground)	2 / 37
Subject	(Conclusion Part - Ground)	2 / 37
Relation	(Conclusion Part - Ground)	5 / 37
Construct	(Condition Part)	0 / 37
Quantifier	(Condition Part)	0 / 37
Subject	(Condition Part)	0 / 37
Relation	(Condition Part)	0 / 37
Connective	(Condition Part)	1 / 37
Propositional Operator	(Condition Part - Classification)	0 / 37
Value	(Condition Part - Classification)	0 / 37
Quantifier	(Condition Part - Classification)	0 / 37
Subject	(Condition Part - Classification)	0 / 37

<b>Relation</b>	(Condition Part - Classification)	0 / 37
<b>Mathematical Operator</b>	(Condition Part - Ground)	0 / 37
<b>Mathematical Function</b>	(Condition Part - Ground)	0 / 37
<b>Value</b>	(Condition Part - Ground)	0 / 37
<b>Quantifier</b>	(Condition Part - Ground)	0 / 37
<b>Subject</b>	(Condition Part - Ground)	0 / 37
<b>Relation</b>	(Condition Part - Ground)	0 / 37

**Table 4.11:** Amount of Empty Cells per Fundamental Construct

Considering Table 4.11 above, the fundamental construct *Relation* remained 2 out of 37 times empty in the conclusion part and the Modal Claim Type 3 out of 37 times. These numbers are based on the given patterns, in these cases no pattern part could be indicated as *Relation* or *Modal Claim Type*. An explanation for the finding with regard to the fundamental construct *Relation* is that this fundamental construct is only necessary to indicate the relation between two subjects. However, the two patterns for which the cells remained empty, were targeted at specifying only one subject. With regard to the fundamental construct *Modal Claim Type*, this fundamental construct is not important for the logic of a business rule but it can be included for readability and understandability as explained in section 3.2. However, both fundamental constructs appear in practice as can be concluded from the other patterns for which the cells did not remain empty. Furthermore, the fundamental construct *Relation* is significant to be able to precisely specify the relation between subjects in a business rule with the CNL as explained in section 3.1. In addition, the *Modal Claim Type* can be included to enhance the readability and understandability of the intention of the business rules as described in section 3.2. Therefore, these fundamental constructs are considered as necessary and will stay included in the CNL.

In the conclusion part, the following fundamental constructs related to a classification remained empty more often: *Value*, *Quantifier*, *Subject* and *Relation*. This can be explained by the fact that two types of classifications exist, which are mutually exclusive, since it is impossible to classify ‘something’ as a Value and a Subject at the same time. When a pattern was targeted at classifying ‘something’ as a Value, the cells of the Quantifier, Subject and Relation stayed empty. It should be noted that when a classification did not include a Subject, also the Quantifier and Relation remained simultaneously empty, see for an explanation section 3.1. In addition, the fundamental constructs *Value*, *Quantifier*, *Subject*, *Relation* and *Mathematical Function* remained also empty a few times with regard to a ground. This can be explained by the fact that these patterns only specified a very simple or specific calculation.

Lastly, the *Connective* remained empty one time which can be explained by the fact that this pattern only allowed the specification of one condition.

In summary, validation round one revealed that all the 37 patterns could be mapped, which ensured that no fundamental constructs were lacking. Furthermore, a logical explanation could be found when a cell of a fundamental construct remained empty. In addition, no fundamental construct remained empty for all 37 patterns. Therefore, it can be concluded that all the selected fundamental constructs are significant to include in the CNL.

### 4.2.3 Results Validation round 2 – Instance Level View

To recall, a set of 150 business rules is randomly selected from eleven different business rule cases by means of the data collection process. The following cases were used: 1) Morgan (2002), 2) RuleSpeak patterns of Hoppenbrouwers (2011), 3) Wan-Kadir and Loucopoulos (2004), 4) Von Halle (2001), 5) RegelSpraak of Sangers-van Cappellen (2014), 6) WereWolf (DM Community, 2015), 7) Diabetic Patient Monitoring (Parish, 2014), 8) Patient Therapy (Feldman, 2014), 9) Tax Return (Feldman, 2011), 10) Au Pair (Anonymous Dutch government organization), and 11) UServ Product Derby (BBC, 2015).

Altogether, these eleven business rule cases provided 313 business rules which differed from each other with regard to their type. Since only derivation business rules were eligible for the validation, the business rules from each case were first evaluated. After that, 150 derivation business rules were randomly selected for the validation. Table 4.12 shows the amount of selected business rules per case that were in scope and the total amount of business rules that a case contained. By following this approach, a mixed set of 150 business rules extracted from all eleven cases was gained.

Case	Total Amount of Business Rules Available	Amount of Selected Business Rules In Scope
Morgan	9	4
RuleSpeak	125	11
Wan Kadir & Loucopoulos	13	4
Von Halle	21	9
RegelSpraak	33	19
WereWolf	8	8
Diabetic Patient Monitoring	3	3
Patient Therapy	6	6
Tax Return	16	16
Au pair	1	1
UServ Product Derby	71	69
<b>TOTAL</b>	<b>313</b>	<b>150</b>

**Table 4.12:** Figures Data Collection Process for Validation Round 2

## RESULTS

After the eligibility check of the collected data was performed, the 150 business rules in scope were mapped by means of the Mill's method to validate the fundamental constructs from an instantiation point of view. The majority of the used business rules (i.e. 119 of the 150) was formulated in English. To investigate if the language aspect could have an effect on the logic of the specified business rules, and in this way on the outcome of the mapping, all the business rules from five of the eleven cases are translated into Dutch prior to the mapping. These cases comprised 102 business rules in total, which are mapped in English and in Dutch. This choice has been made since the case company, the Dutch Tax and Customs Organization, formulates their business rules in Dutch.

The entire mapping, including both Dutch and English business rules, was also performed by a second researcher, which acted as reliability coder. The validation is performed two times because the possibility exists that the results of the validation are influenced by the mindset and convention of the researcher. Involving a reliability coder could reduce this effect and could enhance the reliability of the validation (Mays & Pope, 1995). To ensure that the mapping was performed in exactly the same way by the reliability coder as the researcher, a mapping procedure was documented in detail. Furthermore, the researcher mapped and explained a few example business rules to the reliability coder in advance. After both mappings were conducted, the results were compared to investigate the agreement between the two researchers. The comparison showed that the two mappings corresponded exactly in terms of logic, only some deviations occurred in terms of representation. This can be explained by the fact that some business rules could not be mapped directly, due to the fact that these business rules resembled the human language to a large extent. In other words, these business rules included a lot of context (i.e. words) for readability and human understandability. These business rules had to be rewritten into a more atomic form to be able to map them. As stated in Chapter 3, atomic business rules are *“business rules that cannot be further decomposed without losing meaning”* (Boyer & Mili, 2011). When rewriting these business rules to atomic ones, the logic remained the same only the representation was altered.

To make this rewriting process more clear, consider the examples in Table 4.13 below. The left column shows the original business rule derived from a case, and the right column the rewritten version. To provide an example of a deviation between both researchers due to this rewriting process, consider the first rewritten business rule below again. This business rule is the rewritten version of the researcher. In contrast, the reliability coder rewrote this business rule as follows: *“If allergy of patient is penicillin, then the therapy choice is levofloxacin”*. The only difference between the two mappings is the instantiation for the fundamental construct **Relation** that was used (i.e. “.” or “of”). So, both researchers mapped the logic of the business rule the same (i.e. filled the Relation cell), only the representation differed.

	Original Business Rules	Rewritten Business Rules
1	If Patient is allergic to Penicillin, then the therapy choice is Levofloxacin.	If patient.allergy is penicillin, then the therapy choice is Levofloxacin.
2	If a customer has no outstanding invoices, then the customer is of preferred status.	If a customer.outstandinginvoices = FALSE Then the customer.status = preferred.
3	The member's volume discount amount is computed as the product of standard volume discount rate times the number of hours the member spent in a park over the volume discount threshold for that park.	The member's volume discount amount is computed as (standard volume discount rate) * (the volume discount threshold for that park – the number of hours the member spent in a park).

Table 4.13: Examples of Rewriting Process



Besides comparing the mapping of the researcher and the reliability coder, also the mapping of the original English business rules and the mapping of the translated version of these business rules are compared. This comparison is performed to investigate if the language aspect had an effect on the logic of the specified business rules and indirect on the mapping of the business rules. From the comparison emerged that although the representation changed (from English to Dutch), no difference occurred in the mapping of these business rules in terms of logic.

All business rules that are used for the validation are documented along with the rewritten version of the researcher. Appendix 6 includes the entire mapping of all these business rules. To provide more insight in how these business rules were mapped, two examples are provided on the next page. The example business rules are split up again for readability reasons into two different tables. Table 4.14 shows the mapping of the two business rules onto the Conclusion part, and Table 4.15 shows the mapping of these same business rules onto the Condition part.

The first example (see row nr. 1 of Table 4.14 and Table 4.15) corresponds to the following business rule from the use case "UServ Product Derby":

*"If the car is convertible, then the car's potential theft rating is high."*

The second example (see row nr. 2 of Table 4.14 and Table 4.15), corresponds to the following business rule from Wan-Kadir and Loucopoulos (2004):

*"The total amount of a bill is computed as the sum of the bill item amount."*

From the mapping of the entire business rule set, a specific finding occurred which could indicate that fundamental constructs were lacking. First of all, some business rules were specified as IF-THEN constructions. The word 'then' could not be mapped since it does not correspond to one of the fundamental constructs. This is shown in the mapping of example business rule one, see row nr 1 of Table 4.14 again. However, 'then' is only used for readability and when it would be omitted, the logic of the business rule will not change: *"If the car is a convertible, the car's potential theft rating is high."* Therefore, no additional fundamental construct will be included for the CNL. The mapping also revealed that some business rules explicitly specify the unit of measurement, for instance: *"If HbA1C-level is less than 7% then diabetes risk is Low"*. This example indicates that the HbA1C-level (i.e. subject) should be specified in percentages (i.e. unit of measurement). By mapping this business rule, it became clear that there is no fundamental construct to capture the unit of measurement (i.e. %). However, this issue will not be addressed by including an additional fundamental construct but it is recommended to establish a fact model. By means of this fact model, the subjects and the properties of these subjects will be recorded separately from the business logic. In this way, the system is aware of the fact that the value of the HbA1C-level subject represents percentages and the business rule author should not specify this in the business rule. In section 4.2.4, the concept *fact model* will be addressed in more detail when the technical implementation (i.e. implementation dependent view) is taken into account.





Derivation Business Rule														
Conclusion Part														
Quantifier	Subject	Relation	Modal Claim Type	Expression										
				Classification					Ground					
				Propositional Operator	Value	Quantifier	Subject	Relation	Mathematical Operator	Mathematical Function	Value	Quantifier	Subject	Relation
1	(then) the	car	's	-	is	high	-	-	-	-	-	-	-	-
		potential theft rating												
2	The	total amount	of	-	-	-	-	-	-	is computed as	the sum of	-	the	bill item amount
	a	bill												

Table 4.14: Example Mapping of Business Rules on Conclusion Part

Derivation Business Rule														
Condition Part														
Construct	Quantifier	Subject	Relation	Connective	Expression									
					Classification					Ground				
					Propositional Operator	Value	Quantifier	Subject	Relation	Mathematical Operator	Mathematical Function	Value	Quantifier	Subject
1	If	the	car	-	-	is	convertible	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 4.15: Example Mapping of Business Rules on Condition Part

The complete mapping, from both the researcher as the reliability coder, showed that it was possible to map all 150 business rules. However, also this validation resulted in cells that remained empty. These empty cells were investigated further because they could indicate that the selection of the fundamental constructs for the CNL was not correct. As mentioned previously, a part of the business rule set was translated into Dutch only to investigate if the language aspect could have an effect on the logic and mapping of the business rules. Since the comparison between the mapping of the original English business rules and the translated version showed no differences in terms of logic, the translated business rules were omitted from the mapping prior to the analysis of the empty cells.

To identify the possible causes for the empty cells, the mapping of the 150 original business rules was analyzed. From the analysis emerged the same two overall causes as came forward during validation round one. Firstly, in case a business rule comprised a *ground* all the fundamental constructs related to a *classification* remained empty. The other way around, mapping a *classification* business rule resulted in empty cells for the fundamental constructs related to a *ground* business rule. Moreover, some business rules merely included a conclusion part and did not specify conditions. As a result, all fundamental constructs of the condition part remained empty. On the other hand, RegelSprak provided example business rules that only specified the condition part resulting in empty cells for the fundamental constructs of the conclusion part. Since a logical reason is found for these empty cells, they were not analyzed further.

The second overall finding was that for some business rules, individual fundamental constructs remained empty. For these empty cells, no direct explanation could be found in contrast to the ones as described above. To obtain insight into these individual empty cells, the amount of empty cells per fundamental construct was calculated. Table 4.16 shows the result of this calculation.

Fundamental Constructs		Amount of empty cells
Quantifier	(Conclusion Part)	67
Subject	(Conclusion Part)	0
Relation	(Conclusion Part)	101
Modal Claim Type	(Conclusion Part)	119
Propositional Operator	(Conclusion Part - Classification)	0
Value	(Conclusion Part - Classification)	8
Quantifier	(Conclusion Part - Classification)	72
Subject	(Conclusion Part - Classification)	70
Relation	(Conclusion Part - Classification)	77
Mathematical Operator	(Conclusion Part - Ground)	0
Mathematical Function	(Conclusion Part - Ground)	47
Value	(Conclusion Part - Ground)	23
Quantifier	(Conclusion Part - Ground)	52
Subject	(Conclusion Part - Ground)	6
Relation	(Conclusion Part - Ground)	68
Construct	(Condition Part)	0
Quantifier	(Condition Part)	109
Subject	(Condition Part)	0
Relation	(Condition Part)	124
Connective	(Condition Part)	93
Propositional Operator	(Condition Part - Classification)	0
Value	(Condition Part - Classification)	11
Quantifier	(Condition Part - Classification)	152
Subject	(Condition Part - Classification)	149
Relation	(Condition Part - Classification)	160



<b>Mathematical Operator</b>	(Condition Part - Ground)	0
<b>Mathematical Function</b>	(Condition Part - Ground)	48
<b>Value</b>	(Condition Part - Ground)	4
<b>Quantifier</b>	(Condition Part - Ground)	47
<b>Subject</b>	(Condition Part - Ground)	43
<b>Relation</b>	(Condition Part - Ground)	47

Table 4.16: Amount of Empty Cells per Fundamental Construct

By analyzing the figures of the second validation round (see Table 4.16) and comparing it to the figures of the first validation round (see Table 4.11), it emerged that the reasons why the cells remained empty for an individual fundamental construct appeared to be similar for both validation rounds. These reasons are already extensively described in section 4.2.2. and therefore not explained in this section again. Only two overall deviations emerged by comparing the figures from the first and second validation, which are explained below.

## DEVIATIONS BETWEEN VALIDATION ROUND 1 AND VALIDATION ROUND 2

The *first deviation* that can be appointed between both validation rounds is that the amount of empty cells per fundamental construct is considerably higher for validation round 2. The *second deviation* between both validation rounds is that the following fundamental constructs remained empty several times (see Table 4.16) for validation round 2, which did not remain empty once for the first validation round:

- In the Conclusion part:
  - Quantifier
- In the Condition part:
  - Quantifier
  - Relation
  - Value (classification)
  - Quantifier (classification)
  - Subject (classification)
  - Relation (classification)
  - Mathematical Function (ground)
  - Value (ground)
  - Quantifier (ground)
  - Subject (ground)
  - Relation (ground)

The occurrence of the two deviations can be explained by the fact that during validation round 2 specific instantiations (i.e. business rules) are mapped, in contrast to the first validation round during which the patterns (i.e. templates to specify business rules) were mapped. Although a part of the mapped business rules was established by adhering to the specific patterns, still a lot of freedom in specifying the business rules retained for a business rule author. This freedom can be explained by the low granularity level of the building blocks of existing patterns. For instance, various patterns included one building block (e.g. "COND") to cover the entire condition part. So, during the first validation round the building block COND was mapped onto all fundamental constructs of the condition part and these fundamental constructs did not remain empty (see section 4.2.2). However, when the business rule author applied such a pattern to specify a specific instantiation he or she could choose every possible way to specify the conditions of the business rule. For example, the business rule author could write the following condition part: "if driver is young driver AND driver is married AND driver.locatedInCA\_NY\_VA is TRUE". When this business rule was mapped during validation round 2, it resulted in a lot of empty cells for the fundamental constructs in the condition part. Table 4.17 shows the mapping of this example in which the empty cells are indicated by a dash (-). The example business rule covers three individual conditions that all **check the consistency** between a subject and a value.

In this example, for none of the three *subjects* (i.e. driver, driver, driver.locatedInCa\_NY\_VA) a *quantifier* is specified resulting in already three empty cells for the *quantifier* in the second column. Furthermore, two of the *subjects* do not have a *relation* resulting in two empty cells. Since none of the conditions check the consistency between a subject and another subject, also the fundamental constructs *quantifier*, *subject* and *relation* below the expression remained three times empty. These figures (amount of empty cells) result from the mapping of solely one derivation business rule. This can also explain why the amount of empty cells for a single fundamental construct could even exceed the total amount of 150 business rules that were mapped (see Table 4.16).

Condition Part									
Construct	Quantifier	Subject	Relation	Connective	Expression				
					Classification				
					Propositional Operator	Value	Quantifier	Subject	Relation
if	-	driver	-	AND	is	young driver	-	-	-
	-	driver	-	AND	is	married	-	-	-
	-	driver	.		is	TRUE	-	-	-
		locatedInCA_NY_VA							

Table 4.17: Example Mapping Specific Instantiation of Condition Part

In conclusion, validation round two showed that all the 150 business rules could be mapped. This ensured that no fundamental constructs were lacking. Furthermore, a logical explanation could be found when a cell of a fundamental construct remained empty. Therefore, it can be concluded that all the selected fundamental constructs are significant to retain included in the CNL. In addition, the seven fundamental constructs that appeared to be zero times empty (see Table 4.16) correspond to the fundamental constructs that are made obligatory for the CNL. This finding provides substantiation for the establishment of the grammar rules of these fundamental constructs.

### 4.2.4 Results Validation round 3 – Implementation Dependent Level View

In previous two validation rounds, especially the implementation independent side is taken into account. In contrast, this third validation round focused on the implementation dependent level view. Recall that the documentation of the implementation of the “UServ Product Derby” business rule set into six different BRM systems is used as input for validation round three. When this documentation was not sufficient enough to perform the mapping of the implementation components onto the fundamental constructs, manuals or the actual BRMSs were analyzed. The implementation documentation was supplied by the following six different BRMS vendors: 1) Blueriq (Schadd, 2015), 2) Corticon (Parish, 2015), 3) IBM ODM (Ortiguela, 2015), 4) Sapiens (Segal, 2015), 5) OpenRules (Feldman, 2015), and 6) OpenL Tablets (Bastun, 2015).

#### RESULTS

The implemented version of the business rules from the implementation documentation were mapped by means of Mill’s method. This mapping can be found in Appendix 7. To provide more insight in how the mapping was done, two examples derived from the implementation documentation are provided on the next page. The example business rules are split up again for readability reasons into two different tables. Table 4.18 shows the mapping onto the Conclusion part, and Table 4.19 shows the mapping onto the Condition part.

The first example (see row nr. 1 of Table 4.18 and Table 4.19) corresponds to the following business rule from the use case “UServ Product Derby” implemented into the BRMS *Blueriq*:

- *If all of the following are true, then the car’s potential theft rating is moderate:*
  - *car’s price is between \$20,000 and \$45,000*
  - *car model is not on the list of “High Theft Probability Auto”*

How this business rule is implemented into the BRMS *Blueriq* is shown in Figure 4.2.

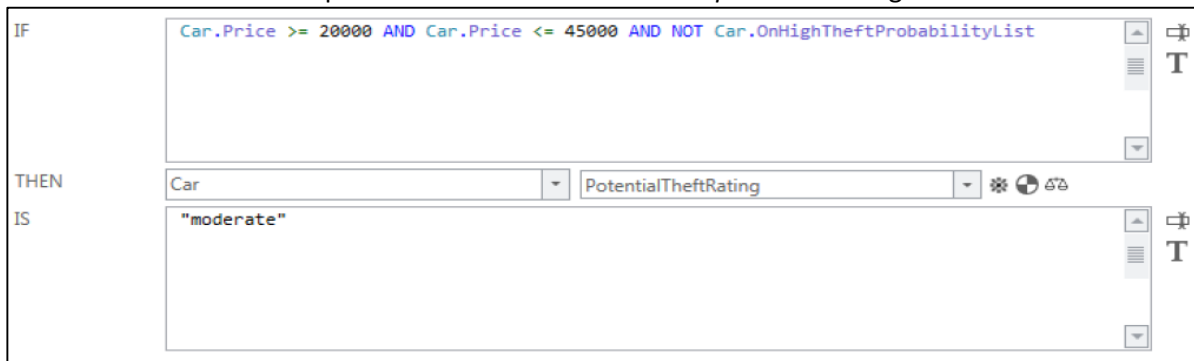


Figure 4.2: Example of Implemented Business Rule in Blueriq (adapted from Schadd, 2015)

The second example (see row nr. 2 of Table 4.18 and Table 4.19) corresponds to the following business rule from the use case “UServ Product Derby” implemented into the BRMS *OpenRules*:

- *If a preferred client, lower the premium by \$250.*

How this business rule is implemented into the BRMS *OpenRules* is shown in Figure 4.3.

DecisionTableMultiHit AdjustUsingClientSegment			
Condition		Conclusion	
Client Segment		Client Total Premium	
Is	Preferred	=	250

Figure 4.3: Example of Implemented Business Rule in OpenRules (adapted from Feldman, 2015)



Derivation Business Rule															
Conclusion Part															
Quantifier	Subject	Relation	Modal Claim Type	Expression						Ground					
				Classification					Ground	Mathematical Operator	Mathematical Function	Value	Quantifier	Subject	Relation
				Propositional Operator	Value		Quantifier	Subject	Relation						
1	-	THEN Car	.	-	IS	"moderate"		-	-	-	-	-	-	-	
		PotentialTheftRating													
2	-	Client Total Premium	-	-	-	-		-	-	-	=	-	250	-	

Table 4.18: Example Mapping of Business Rules on Conclusion Part

Derivation Business Rule																
Condition Part																
Construct	Quantifier	Subject	Relation	Connective	Expression					Ground						
					Classification					Ground	Mathematical Operator	Mathematical Function	Value	Quantifier	Subject	Relation
					Propositional Operator	Value		Quantifier	Subject	Relation						
1	IF	Car	.	AND	-	-		-	-	-	>=	-	20000	-		
		Price														
		Car	.	AND NOT	-	-		-	-	-	<=	-	45000	-		
		Price														
		Car	.		-	-		-	-	-	-	-	-	-		
		OnHighTheftProbability														
2	Condition	Client Segment	-	-	Is	Preferred		-	-	-	-	-	-	-		

Table 4.19: Example Mapping of Business Rules on Condition Part

The complete mapping showed that it was possible to map all the implemented business rules. However, two aspects emerged from the implementation documentation that could not be addressed which could indicate that fundamental constructs were lacking namely: 1) Context-structures and 2) Technical Implementation Information. These two aspects will be described below.

## CONTEXT-STRUCTURES

First of all, the implementation documentation included *context-structures*. In the documentation different ways to visualize and to refer to these context-structures are found (e.g. according to Sapiens it is a decision model, according to Corticon it is a Rule Flow Diagram, according to Blueriq it is a Decision Requirement Diagram). An explicit example of a context-structure is depicted in Figure 4.4 which is adapted from the implementation documentation of Blueriq. This context-structure shows how the contexts, with regard to determining the premium of a driver, are connected.

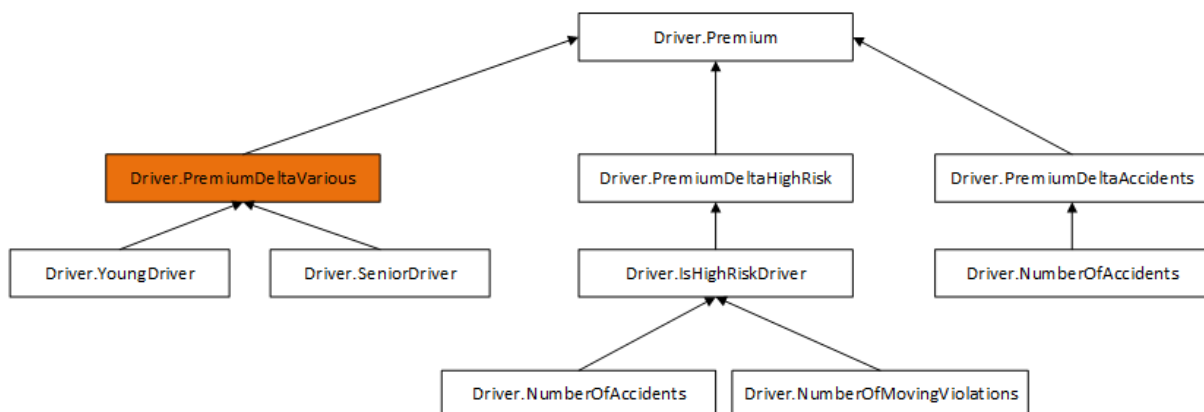


Figure 4.4: Explicit Example of Context-Structure in Blueriq (adapted from Schadd, 2015)

The orange colored context in Figure 4.4 called “Driver.PremiumDeltaVarious” includes the following six business rules which originate from the use case “UServ Product Derby”:

- *If young driver and married and located in CA, NY or VA, then increase premium by \$700.*
- *If young driver and single and located in CA, NY or VA, then increase premium by \$720.*
- *If young driver and married and not located in CA, NY or VA, then increase premium by \$300.*
- *If young driver and single and not located in CA, NY or VA, then increase premium by \$300.*
- *If senior driver and located in CA, NY or VA, then increase premium by \$500.*
- *If senior driver and not located in CA, NY or VA, then increase premium by \$200.*

The business rules from the context “Driver.PremiumDeltaVarious” are implemented in Blueriq with the decision table as shown in Figure 4.5. This decision table determines the change in the amount of premium a driver has to pay. Each of the six business rules determine the value of the decision “Driver.PremiumDeltaVarious” (i.e. conclusion). The table includes one conclusion (see orange border in Figure 4.5), and four conditions which are listed in the left-most column of Figure 4.5 (see green border). The table should be read from top to bottom like: if Driver.YoungDriver is TRUE and Driver.LocatedInCa\_NY\_VA is TRUE and Driver.Married is FALSE, then Driver.PremiumDeltaVarious is 720. In this example, the subject Driver.SeniorDriver is ignored due to the included asterisk (\*) which indicates that a subject in the most-left column is not included in that business rule. Furthermore, the two brackets ( [ ] ) indicate that any non-mentioned option matches. In this case, when Driver.YoungDriver would be FALSE the pathway of the right-most column would be followed.

Driver.YoungDriver	TRUE	[]		
Driver.SeniorDriver	*	TRUE		
Driver.LocatedInCA_NY_VA	TRUE	FALSE	TRUE	FALSE
Driver.Married	TRUE	FALSE	*	*
Driver.PremiumDeltaVarious	700	720	300	500
			200	

Figure 4.5: Example Implementation of the Business Rules from One Context in Blueriq (adapted from Schadd, 2015)

In summary, a context-structure connects multiple contexts and/or sub-contexts, where a *context* represents one *decision*. A context needs to be specified by means of a set of business rules. In other words, a context includes multiple *individual business rules* to determine one and the same decision. How these concepts are related is conceptually shown in Figure 4.6.

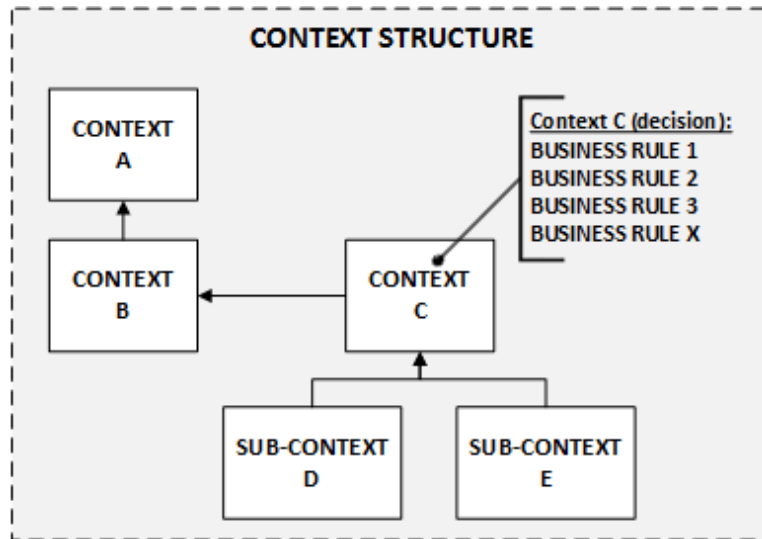


Figure 4.6: Relation between context structure, context (decision), and business rules in a context

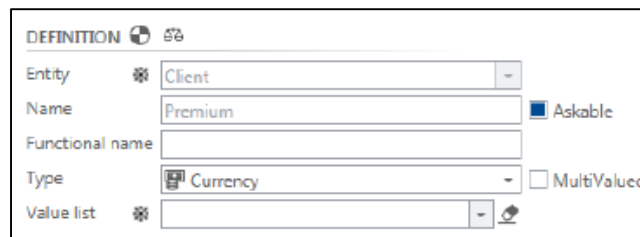
Business rule engines use these context-structures to establish the connections between different decisions and business rules, or in other words they use them to understand how a value of a specific subject in a business rule can be derived from another business rule. For example, the conclusion of **Decision/Context C** can be incorporated as a condition in **Decision/Context B**. Moreover, these context-structures can ensure that the organization keeps an overview of the implemented business rules and their connections. However considering the CNL of this research, the language and patterns will be devised to specify **individual business rules in a context**. Therefore, contexts and context-structures lie on a higher level than the level for which the CNL will be devised. In conclusion, the context-structures are not taken into account for the CNL.



### TECHNICAL IMPLEMENTATION INFORMATION

The analysis of the implementation documentation also revealed that a business rule engine requires detailed information about *where* it can retrieve the values of subjects (either variable or constant values) when executing a business rule. This information needs to be specified with enough detail for the business rule engine to fetch the data out of the correct database fields. Furthermore, it needs to be specified *how* the business rule engine should retrieve this information from the database. These two aspects imply precision on a technical implementation level. In case the *where* and *how* aspects would be taken into account for the patterns, the implementation in a specific database would already be specified within the business rules during specification time. Therefore, both aspects will not be included within (the patterns of) the CNL, otherwise the language will not be implementation independent. *How* and from *where* the business rule engine should retrieve the values of subjects (either variable or constant values) can be addressed by using an Application Programming Interface (API) for example JavaScript Object Notation (JSON) or Representational State Transfer (REST).

In addition, from the analysis emerged that the subjects, the relations between subjects and the properties of the subjects need to be precisely specified for a business rule engine (i.e. *What* it should retrieve). These properties comprise a specification of the subject itself and the data type (e.g. INTEGER, BOOLEAN) including the unit of measurement (e.g. PERCENTAGE). Consider for example the BRMS Blueriq, these properties are specified for every occurring subject in the business domain. In Figure 4.7, an interface of Blueriq is shown for specifying the data type of a subject. In this case, the data type *Currency* is selected for the subject *Client.Premium*.



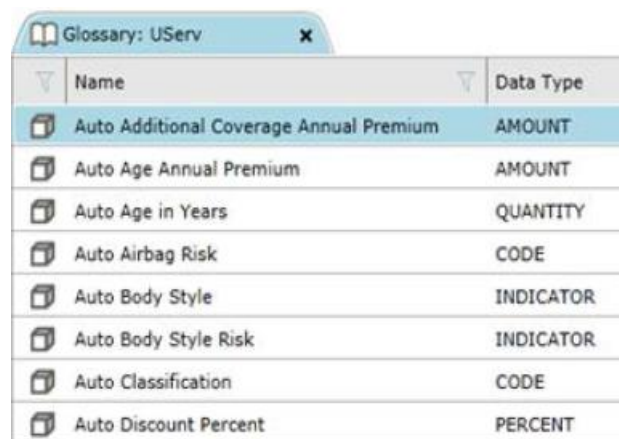
The screenshot shows a form titled 'DEFINITION' with the following fields:

- Entity: Client
- Name: Premium
- Functional name: (empty)
- Type: Currency
- Value list: (empty)

There are also checkboxes for 'Askable' and 'MultiValued'.

Figure 4.7: Example Specification of Subject in Blueriq (adapted from Schadd, 2015)

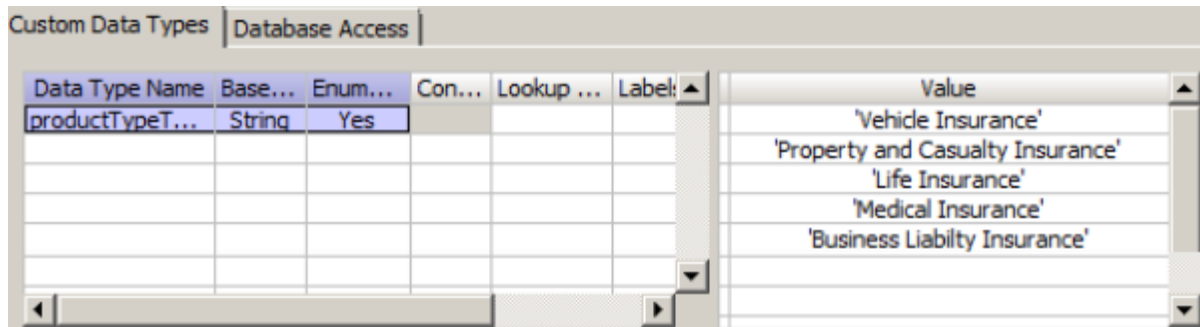
Another example is found in the BRMS Sapiens which uses a glossary to record all of the approved Fact Type names (i.e. subjects) and data types. Figure 4.8 shows a snapshot of the glossary of Sapiens.



Name	Data Type
Auto Additional Coverage Annual Premium	AMOUNT
Auto Age Annual Premium	AMOUNT
Auto Age in Years	QUANTITY
Auto Airbag Risk	CODE
Auto Body Style	INDICATOR
Auto Body Style Risk	INDICATOR
Auto Classification	CODE
Auto Discount Percent	PERCENT

Figure 4.8: Example Specification of Subjects in a glossary of Sapiens (adapted from Segal, 2015)

Similar is the approach of Corticon, this business rules management system includes a vocabulary (see Figure 4.9) which contains the business objects (i.e. subjects) and their relationships, their attributes (i.e. subjects), data types and possible values. This vocabulary can also comprise the mapping of business objects to tables and attributes to columns in a database.



Data Type Name	Base...	Enum...	Con...	Lookup ...	Label
productTypeT...	String	Yes			

Value
'Vehicle Insurance'
'Property and Casualty Insurance'
'Life Insurance'
'Medical Insurance'
'Business Liability Insurance'

Figure 4.9: Example Specification of Subjects in a vocabulary (adapted from Corticon (Parish, 2015))

In summary, the analyzed BRM systems manage the subjects, associated properties and related subjects separately from the business rules. At specification time, the subjects can be repeatedly retrieved from the glossary of the system to be incorporated in the business rules which ensures reusability. So, these systems consider both aspects as an individual *separation of concern*. Above provided examples show three specific implementation dependent ways to capture these two separation of concerns. To address this separation in an implementation independent way, the use of a *fact model* is proposed in literature. A Fact model is considered as a diagram that structures the business knowledge by means of specifying logical connections (called *facts*) between core concepts of the business (i.e. subjects) with the aim to standardize the business terminology in an implementation independent fashion (R. G. Ross, 2000; R. G. Ross, 2000).

The advantage of adhering to this separation of concerns is that when for example the properties of a subject change, only the fact model has to be altered and not the business rule set itself. Therefore, when devising the pattern set, it will be taken into account that the business rule engine has to know *what* it should retrieve by specifying the subjects and relations. However, not the properties of these subjects and relations between the subjects will be incorporated into the patterns. It is recommended to create a separate fact model from which this information can be retrieved.

## 5 Pattern Catalogue Creation

This chapter describes the process of creating the pattern set. Furthermore, the choices that have been made during the creation process are explained. The aim of this chapter is to make clear how and why these patterns are established.

### 5.1 Subdivision Patterns

As reminder, a pattern is defined as: “a structured combination of fundamental constructs to specify business rules” (Ghose & Koliadist, 2007; Morgan, 2002; Von Halle, 2001; Zoet, 2014). In other words, each pattern is composed of a set of fundamental constructs. In Chapter 3, an extensive explanation is provided for all identified fundamental constructs of a derivation business rule. Furthermore, their interrelations are described and visualized by means of a meta-model. This meta-model laid the foundation for the creation of the patterns, tailored to the specification of derivation business rules. To recall, in general a business rule can be composed of a conclusion and a condition part at the highest level. Regarding the CNL, it has been decided that a derivation business rule should always include exactly one conclusion part comprising either a classification or a ground. Additionally, a derivation business rule may contain one or more condition parts. A condition part can cover multiple conditions where each individual condition can also include either a classification or a ground.

These observations and requirements of a derivation business rule provided the basis for establishing the set of patterns. The justification of using this basis can be ensured by the outcome of the validation of the fundamental constructs as described in Chapter 4. The pattern set is established by traversing different phases of the research. From the literature study, in which current pattern catalogues including example business rules were investigated, recurring elements and their relations emerged. Moreover, during the first two validation rounds of the preliminary validation, 37 patterns and 150 business rules have been evaluated (see Chapter 4), from which similar and additional repeating elements and their connections became apparent. For the actual establishment of the patterns, it has been decided to create specific individual patterns for specifying specific parts of a derivation business rule. The rationale for this subdivision is twofold. Firstly, the patterns had to comply with the observations and requirements of a derivation business rule. Secondly, this subdivision can guide and support a business rule author by selecting the applicable pattern during the specification process. Two criteria for the subdivision were ‘*completeness*’ and ‘*mutual exclusiveness*’. This latter criterion implies that for the specification of a particular derivation business rule only one pattern should be applicable to choose. By adhering to this criterion, the required time for the specification process can be reduced and also the consistency between business rule authors can be enhanced. Eventually, this creation process led to 19 patterns in total.

The subdivision of patterns is depicted in Figure 5.1 as a ‘decision tree’. First of all, patterns are divided into patterns for the conclusion part and patterns for the condition part (see branch 1 & 2 in Figure 5.1). Unique patterns are devised for both parts, since the two parts comprise:

1. distinctive associated fundamental constructs. For example, the *Modal Claim Type* is only included in the conclusion part and the *Connective* only in the condition part;
2. different interrelations (i.e. cardinalities) between identical fundamental constructs which depend on the part they are included.

This separation is directly the first aspect to consider by a business rule author to shorten the pattern selection process: Does the business rule only include a conclusion part or also conditions? In this way, he or she can choose to follow the left branch (**number 1**) and/or the right branch (**number 2**) of the decision tree in Figure 5.1.

With regard to the **conclusion part (branch 1)**, the business rule author has to determine if it is a classification (1.1) or a ground (1.2) business rule (See Figure 5.1). Both the classification patterns as ground patterns have the aim to equate the value of a subject with ‘something’. This ‘something’ makes the difference between the patterns and results in a further subdivision. A *classification* can equate a subject with: (A) a value or (B) a subject. So, two different patterns (1.1 A – B) are available for specifying a classification in a conclusion part. A *ground* can equate a subject with: (A) a basic ground. Thus a business rule author has one pattern (1.2 A) to specify the conclusion part comprising a ground.

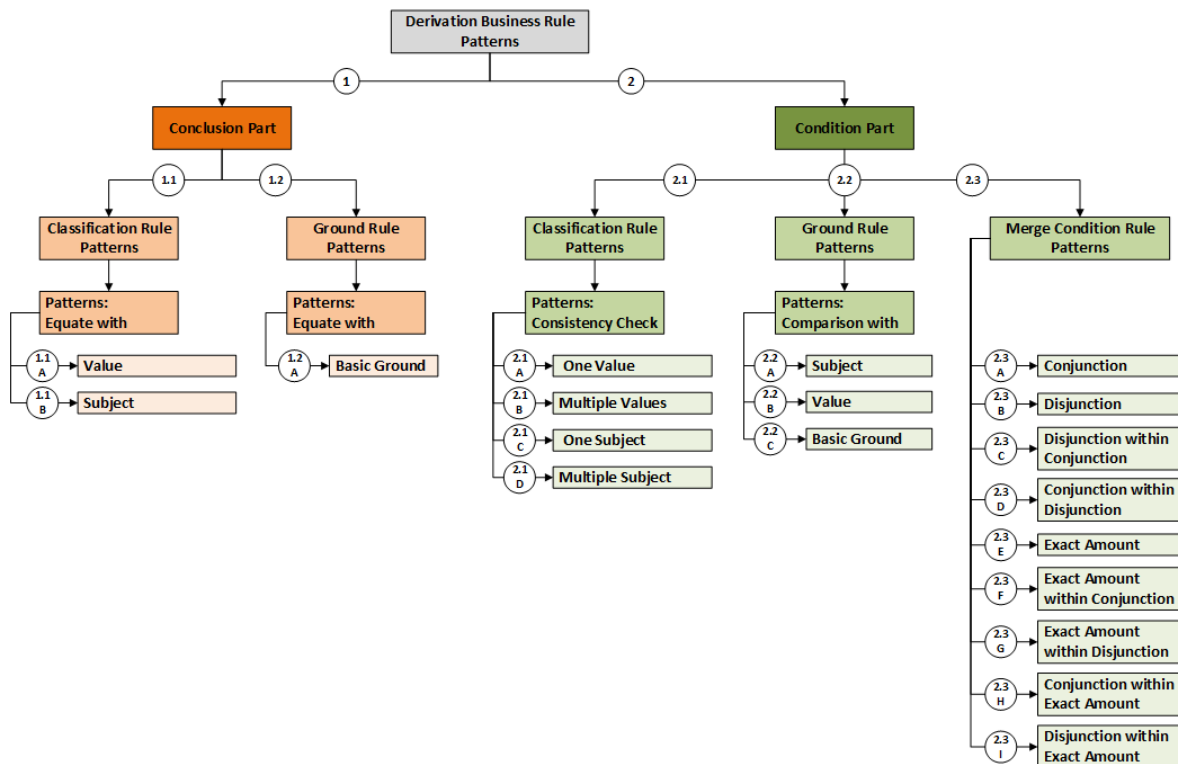


Figure 5.1: Subdivision of Patterns depicted as Decision Tree

With regard to the **condition part (branch 2)**, these patterns are subdivided into three types (see Figure 5.1): patterns to specify individual classification conditions (2.1), patterns to specify individual ground conditions (2.2), or patterns to merge multiple individual conditions (2.3). This subdivision emerged due to the fact that the condition part can include more than one individual condition. The first choice that a business rule author has to make for each individual occurring condition is whether the condition includes a classification (2.1) or a ground (2.2). A *classification* in a condition part can check the consistency between a subject and: (A) one value, (B) a value out of a list of multiple values, (C) one subject, or (D) a subject out of a list of multiple subjects. In this way, for the classification in the condition part four different patterns (2.1 A – D) are established. The consistency check in this part of the business rule means that the value of a subject should be exactly equal to another value in order for the condition to be true. A *ground* on the other hand, can compare a subject with: (A) another subject, (B) a value, or (C) a basic ground. For all these three options, a pattern is created (2.2 A – C).

As mentioned previously, the business rule author has to select a pattern for each occurring individual condition. In case multiple conditions are included in the business rule, the business rule author should also determine what the relation between these conditions is. For instance, should all the conditions be met (i.e. conjunction) or at least one/two/three etcetera conditions (i.e. disjunction). Nine different relations emerged from literature and the preliminary validation, which resulted into the creation of an equivalent number of patterns (see **2.3 A – I** in Figure 5.1). These nine patterns also include the placeholder <Individual pattern> multiple times, which can be replaced by the already selected individual patterns (see Table 5.1). All the other elements and symbols incorporated in the pattern will be explained in the remainder of this section. After the business rule author has determined all the patterns, the parts can be merged together to specify the entire business rule. An example of an entire merged business rule is shown in Table 5.1, where the Disjunction pattern is used along with two individual patterns.

NR	Patternname	Pattern
12.	<b>Disjunction</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] (meets at least <.. > of the following conditions) : <ul style="list-style-type: none"> <li>• &lt; Individual pattern &gt;</li> <li>• &lt; Individual pattern &gt;</li> <li>• [ &lt; Individual pattern &gt; ]</li> </ul>
	<b>Example:</b>	If the person (meets at least one of the following conditions) : <ul style="list-style-type: none"> <li>• If the driving license of the person ( is equal to ) TRUE</li> <li>• If the ID number of the person ( is equal to ) filled</li> </ul>

**Table 5.1:** Pattern example from Condition part – merge condition rule patterns

## 5.2 Modeling Choices

Besides the observations and requirements of a derivation business rule as described above, also some additional modeling choices are made during the pattern set creation. Most of these choices are made to adhere to the cardinalities of the fundamental constructs as incorporated in the meta-model shown in Chapter 3. Below, the modeling choices are categorized into choices with regard to: 1) the conclusion and condition part together, 2) the conclusion part, and 3) the condition part.

### CHOICES – CONCLUSION AND CONDITION PART

Firstly, it has been decided that the fundamental construct ‘**Subject**’ should be included at least once in both the conclusion and condition part. As a result, ‘Subject’ became a fixed pattern part. This pattern part is incorporated as placeholder, since it can contain an infinite number of different instantiations. A placeholder is indicated in the patterns by angle brackets (i.e. < >), see Table 5.2.

In addition, it is decided that in front of every occurring ‘Subject’ the fundamental construct ‘**Quantifier**’ is positioned (see Table 5.2). This choice has been made because the ‘Quantifier’ indicates about how many ‘Subjects’ a system should reason; it makes the business rule more precise and unambiguous as explained in Chapter 3. So, it is important for the automated processing of the patterns. Therefore, the ‘Quantifier’ is also included as fixed pattern part. Furthermore, the ‘Quantifier’ is incorporated as placeholder as it can differ per business rule what the instantiation is. However, for the ‘Quantifier’ a list of optional instantiations is provided in Appendix 4 since there only exist a limited number of quantifiers.

A fundamental construct that is included as optional pattern part is ‘**Relation**’, which indicates that two ‘Subjects’ are related. Since a ‘Relation’ connects two ‘Subjects’, it is always placed directly after

a ‘Subject’ and is succeeded by a ‘Quantifier’ and the other ‘Subject’. To show that these three fundamental constructs together comprise one optional pattern part, they are enclosed by square brackets and made italic as follows: [ <Relation> <Quantifier> <Subject> ]. In the patterns, the possibility is included to repeat this entire pattern part when specifying a derivation business rule which is denoted by *n\** (see Table 5.2). ‘Relation’ is also included as placeholder in the patterns, but a list of optional instantiations is provided in Appendix 4.

For the fundamental construct ‘Value’ it is decided to include this as placeholder in every pattern it occurs (see Table 5.2). As it can contain an infinite number of different instantiations, these will not be defined.

NR	Patternname	Pattern
1.	<b>Equate with VALUE</b>	<Quantifier> <Subject> [ <i>n*</i> <Relation> <Quantifier> <Subject> ] [ < Modal Claim Type > ] ( is equated to   be equated to ) < Value >
	<b>Example:</b> The status of the client must ( be equated to ) gold member  <b>Example 2:</b> The maximum amount of leave days must ( be equated to ) 26	

Table 5.2: Pattern example from Conclusion Part – Classification Rule patterns

With regard to the fundamental construct ‘Mathematical Function’, the choice is made to only include it as a placeholder in the patterns (see Table 5.3). Given the fact that there exist many different instantiations (e.g. SUM(), MAX(), MIN(), AVERAGE ()).

NR	Patternname	Pattern
3.	<b>Equate with Basic Ground</b>	<Quantifier> <Subject> [ <i>n*</i> <Relation> <Quantifier> <Subject> ] [ < Modal Claim Type > ] ( is computed as   be computed as ) <Quantifier> <Subject> [ <i>n*</i> <Relation> <Quantifier> <Subject> ]   <Value>   <Mathematical Function> <i>n*</i> <Quantifier> <Subject> [ <i>n*</i> <Relation> <Quantifier> <Subject> ] [ <i>n*</i> <Mathematical Operator> <Quantifier> <Subject> [ <i>n*</i> <Relation> <Quantifier> <Subject> ] ] <Value>   <Mathematical Function> <i>n*</i> <Quantifier> <Subject> [ <i>n*</i> <Relation> <Quantifier> <Subject> ] ]
	<b>Example:</b> The total amount of profit of a declarant must ( be computed as ) the total amount of income of the declarant minus the total amount of costs of the declarant  <b>Example 2:</b> The amount of income ( is computer as ) the SUM of the total amount of sold units of each order multiplied by the unit price	

Table 5.3: Pattern example from Conclusion Part – Ground Rule patterns

In some cases, it has been decided to include an additional pattern part which is not a fundamental construct namely < .. > (see the Disjunction pattern in Table 5.1). This pattern part is included as placeholder to make a pattern more flexible. A business rule author can replace this placeholder at its discretion (e.g. one, four, the last, the first).

### CHOICES – CONCLUSION PART

As explained in section 3.2, the fundamental construct '**Modal Claim Type**' is not compulsory for the conclusion part but it can be included to enhance the readability. Therefore, the '**Modal Claim Type**' is incorporated as optional pattern part which a business rule author can use (see Table 5.2). In addition, the instantiations of this fundamental construct can differ (e.g. must, may, could, it is permitted that, must not). As a result, the '**Modal Claim Type**' is included as placeholder in the conclusion part of the patterns.

When the conclusion part comprises a classification, the pattern should always encompass a '**Propositional Operator**'. This fixed part is not included as a placeholder, but a specific instantiation is provided in the patterns directly. This choice is made to make the aim of the patterns more clear and to contribute to the establishment of a consistent business rule set. A fixed instantiation of a fundamental construct can be recognized in the patterns by parentheses (i.e. ( ) ). For classifications that equate a value with another value, this fixed instantiation corresponds to: ( is equated to | be equated to ). This fixed instantiation comprises two options for the verb (i.e. *is* or *be*), which is indicated by a vertical bar in the patterns (see Table 5.2). From the separated alternative pattern parts, the business rule author can select one option. Which one the business rule author should select depends on the fact whether or not a '**Modal Claim Type**' is included (e.g. must *be* equated to).

On the other hand, when the conclusion part comprises a ground, the fundamental construct '**Mathematical Operator**' should always be in place according to the meta-model. Therefore, this is a fixed pattern part. For the ground rule pattern of the conclusion part applies that the '**Mathematical Operator**' is set to the following substantiation ( is computed as | be computed as ), see Table 5.3. This is done to show that the value of a Subject should be equated to a computed value. Also this fixed instantiation includes two options for the verb, which option the business rule author should select depends on the inclusion of the '**Modal Claim Type**'. Besides this instantiation, the '**Mathematical Operator**' is also enclosed as placeholder in the patterns to be able to specify a mathematical operation (e.g. add, multiply, subtract). The specific instantiations of this placeholder are listed in Appendix 4. In the patterns, the placeholder for the '**Mathematical Operator**' can be repeated, which is designated with  $n^*$  see Table 5.3. This is done since a calculation can comprise many forms, from very simple to very complex ones.

### CHOICES – CONDITION PART

To indicate the beginning of the condition part, each condition part pattern starts with the fixed fundamental construct named '**Construct**'. From literature and practice, different instantiations emerged like: If, When, Only if. For consistency reasons, the choice has been made to choose ( If ) as fixed part to include in the patterns because this is widely applied (see Table 5.1, Table 5.4, Table 5.5).

The same as mentioned for the conclusion part, when the condition part comprises a classification, the pattern should always include a '**Propositional Operator**'. A fixed instantiation is included in the patterns instead of a placeholder to make the purpose of the patterns more apparent and to contribute to the establishment of a consistent business rule set. For classifications that check the consistency of a value with one other value, this fixed instantiation corresponds to: ( is [*not*] equal to ). When the consistency of one value is checked with another value selected from multiple values, the fixed part corresponds to ( is [*not*] equal to <....> of the following values ) see Table 5.4.



5.	<b>Consistency check - multiple VALUES</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] ( is [ not ] equal to <....> of the following values ) : - < Value > - < Value > - [ n * < Value > ]
<b>Example:</b> If the nationality of <i>the applicant</i> ( is equal to one of the following values ) : - 'CK' - 'GT' - 'ID' - 'MM' - 'NR' - 'NG' - 'PH'  <b>Example 2:</b> If the amount of <i>ordered items</i> ( is equal to one of the following values ) : - 10 - 50		

Table 5.4: Pattern example from Condition part – classification rule patterns

On the other hand, when the condition part comprises a ground, the pattern should always include a **‘Mathematical Operator’**. In the condition part, the ‘Mathematical Operator’ is only included as a placeholder since many different instantiations can be chosen to indicate the comparison with a value, subject, or basic ground. For example: less than, more than, more than or equal to etc. (see Table 5.5). These instantiations are listed along with the particular patterns in Appendix 4.

9.	<b>Comparison with SUBJECT</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] <Mathematical Operator> <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ]
<b>Example:</b> If the amount of rental days is more than the maximum amount of rental days		

Table 5.5: Pattern example from Condition part – ground rule patterns

As explained in section 5.1, besides the patterns to specify an individual condition of a condition part also a set of patterns is devised to specify the relation between multiple individual conditions in a condition part. The relation between these conditions is denoted by the fundamental construct **‘Connective’**. Chapter 3 indicated two ways to apply the ‘Connective’:

1. Using it once in a business rule which specifies the relation between several condition (e.g. meets one of the following conditions), or
2. Using the fundamental construct multiple times between each condition (e.g. AND).

As stated in Chapter 3, the way in which the *connective* is used will not be imposed by the **grammar rules** of the CNL. However, with regard to the patterns the first way will be imposed to contribute to the mutual exclusiveness criterion. To achieve this, it is decided to provide a fixed substantiation for each of the nine different ‘Connectives’ (see Table 5.1). For example, taking the Conjunction pattern into account, the ‘Connective’ is equal to ( meets all of the following conditions ). As described





previously, the nine patterns include the placeholder < *individual pattern* > (see Table 5.1). This placeholder can be replaced by one of the seven individuals classification or ground rule patterns (see 2.1 A - D and 2.2 A – C in Figure 5.1) . The reason to include this placeholder is because the different embodiments of an individual condition are already provided by means of the seven individual patterns. Otherwise, an abundance of patterns would emerge.

In this chapter, five patterns of the pattern set are provided. The entire pattern set is included in Appendix 4, where the pattern set is subdivided into five main categories as shown in Figure 5.1:

1. Conclusion part – classification rule patterns;
2. Conclusion part – ground rule patterns;
3. Condition part – classification rule patterns;
4. Condition part – ground rule patterns;
5. Condition part – merge condition rule patterns.

Each category contains a few patterns, for each pattern an associated example is provided. Furthermore, specific instantiations for several placeholders are provided in this appendix.

## 6 Pattern Validation

In previous chapter, the patterns to specify derivation business rules were identified. This chapter describes the validation process of these patterns. A validation round is performed from an instance level view during which a business rule set is specified by means of the patterns.

For this validation round, the data collection and data analysis process are explained below in two separate sub-sections: data collection and data analysis.

### 6.1 Data Collection

Below, the data collection process is described including the applied sampling strategy and selection criteria.

#### 6.1.1 Validation Round – Instance Level View

For the validation of the patterns, from the instance level view, a new business rule set is obtained. This business rule set was used during this validation round to specify the business rules by means of the established patterns of the CNL to investigate if specific instantiations could be captured by the patterns.

The business rule set was selected and provided by the case study company. Since the Dutch Tax and Customs Administration selected it, the business rule set directly complied to two theoretical selection criteria namely *relevancy* and *representativeness*. The set comprised business rules from the act called the “Zorgverzekeringswet” abbreviated as (ZVW). This act included 16 business rules that are established to determine and calculate the amount of days a taxpayer is income tax (IB) and ZVW-accountable within a tax year in order to determine the ZVW amount a taxpayer is due.

## 6.2 Data Analysis

In this section, a description of how the patterns are validated and the emerging results are provided.

### 6.2.1 Results Validation Round – Instance Level View

During this validation round, the business rule set that was obtained during the data collection process is specified by means of the patterns from the pattern catalogue. The aim of this specification was to validate if it was possible to specify each business rule from the set by applying the patterns from the pattern catalogue. Since the business rule set was written in Dutch and the patterns of the CNL were established in English, the patterns were translated to Dutch prior to the specification. This choice has been made to comply with the Dutch case company and for readability reasons. For example, when the pattern parts were not translated the following vague specification could occur:

< The > <begindatum premieplicht Zvw (H1) > [ is equated to ] < leeg > [ If ] < the > < indicatie geheel jaar geen Zvw plicht (a) > [ is equal to ] < Ja >

The translated version of this pattern looks like follows:

< De > < begindatum premieplicht Zvw (H1) > [ wordt gelijk gesteld aan ] < leeg > [ indien ] < de > < indicatie geheel jaar geen Zvw plicht (a) > [ gelijk is aan ] < Ja >

This translation step did not have an effect on the logic of the specification, only on the representation as already emerged during the preliminary validation of the fundamental constructs (see section 4.2.3). By means of this validation round, it is also demonstrated that the language aspect has no influence on the logic and applicability of the patterns since they can also specify Dutch business rules.

Besides the deviant language of the business rule set from the case company, the set also included “nested business rules”. Nested business rules are complex business rules that include at least more than one conclusion part (see example business rule 9 in the blue rectangle below) (Ligêza, 2006). Given the fact that the patterns are established to specify atomic business rules to comply with the single responsibility principle, the nested business rules also had to be decomposed in several atomic business rules that included exactly one conclusion part prior to the specification.

#### **Example business rule 9. Vaststellen bedrag aanslag (U1):**

ALS	[toepassing artikel 9.4] (U2) = 35	
DAN	[bedrag aanslag] (U1) := 0	(NIHIL-aanslag)
	[indicatie-nihil-aanslag] (U3) := 'ja'	
Anders:	[bedrag aanslag] (U1) :=	
	[bijdrage ZVW aanslag] (a)	(toepassing artikel 9.4 = 0 of 21: aanslag wordt in principe opgelegd)
Einde-Als		

The example business rule above is decomposed into three atomic business rules, see 9a, 9b, and 9c in the blue rectangle below:

#### **Business Rule Transformed to Atomic Business Rule for Specification**

<b>9a</b>	ALS	[toepassing artikel 9.4] (U2) = 35
	DAN	[bedrag aanslag] (U1) := 0
<b>9b</b>	ALS	[toepassing artikel 9.4] (U2) = 35
	DAN	[indicatie-nihil-aanslag] (U3) := 'ja'
<b>9c:</b>	ALS	[toepassing artikel 9.4] (U2) ≠ 35
	DAN	[bedrag aanslag] (U1) := [bijdrage ZVW aanslag] (a)

The entire business rule set comprised 11 out of 16 business rules which were nested. The decomposition process resulted into the establishment of 45 atomic business rules in total. Above provided example was the ninth decomposed business rule, the entire set of used and/or decomposed business rules is listed in Appendix 8.

## RESULTS

After the translation of the patterns and transformation of the business rule set into 45 atomic business rules, the 45 business rules are specified with the patterns. This entire specification can be found in Appendix 9. Two examples of this specification are provided in Table 6.1 and Table 6.2 below, where the left column of these tables provides the atomic business rule originated from the case and the right column shows the business rule specified by means of the applicable patterns. The patterns that are used to specify each business rule and the made additional choices are listed in the bottom row of the tables (see “choices”).

Original business rule NR	Business rule in pattern
<p><b>1b</b></p> <p><b>Als</b> [indicatie geheel jaar geen Zvw plicht] (a) = Ja</p> <p><b>Dan</b> [einddatum premieplicht Zvw] (H2) = [leeg]</p>	<p>&lt; De &gt; &lt; einddatum premieplicht Zvw (H2) &gt; ( wordt gelijk gesteld aan ) &lt; leeg &gt;</p> <p>[ indien ] &lt; de &gt; &lt; indicatie geheel jaar geen Zvw plicht (a) &gt; ( gelijk is aan ) &lt; Ja &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Table 6.1: Business Rule 1 Specified with Patterns

Original business rule NR	Business rule in pattern
<p><b>5b</b></p> <p><b>ALS</b> [vorig jaar alimentatie overgangstarief] (f) = 'nee'</p> <p><b>DAN</b> [bijdrage-inkomen ZVW aanslag] (U1) := [bijdrage-inkomen ZVW zonder alimentatie] (H1) plus [saldo alimentatie na aftrekbare kosten] (e)</p>	<p>&lt; De &gt; &lt; bijdrage-inkomen ZVW aanslag (U1) &gt; ( wordt berekend als )</p> <p>&lt; de &gt; &lt; bijdrage-inkomen ZVW zonder alimentatie (H1)&gt;</p> <p>&lt;plus&gt;</p> <p>&lt; het &gt; &lt;saldo alimentatie na aftrekbare kosten (e)&gt;</p> <p>[ indien ] &lt; de &gt; &lt; vorig jaar alimentatie overgangstarief (f) &gt; ( gelijk is aan ) &lt; nee &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground) and <b>pattern 4</b> (consistency check one value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Table 6.2: Business Rule 5 Specified with Patterns

From the validation, a specific finding occurred with regard to the “condition part - classification rule patterns”. These patterns check the consistency of a subject with something, which was originally indicated in the patterns by means of the following fixed pattern part for the Propositional Operator ( is equal to ). However, in the business rule set that was used for the specification some business rules were incorporated that specified that the subject should **not** be equal to something. Consider the example business rule 9c described above, this business rule checks if the subject “toepassing artikel 9.4 (U2)” is unequal to 35. When devising the pattern set, it was assumed that each business rule could be rewritten from a negative formulation to a positive formulation without affecting the business rule’s logic. Rewriting business rule 9c from negative to positive is possible by including two individual conditions in the business rule: if “toepassing artikel 9.4 (U2)” **is less than** 35 and “toepassing artikel 9.4 (U2)” **is more than** 35. However, in some cases this rewriting process is even not possible. For instance, if a business rule will specify that a subject should be checked to be equal to “all other options” where the amount of other options for the subject is infinite. In case the business rule would be formulated in a positive way, the business rule should list all these options to check the consistency with. Therefore, it has been decided to alter the “condition part – classification rule patterns” by including the optional pattern part [ *not* ] within the fixed pattern part for the Propositional Operator as follows: ( is [*not*] equal to ).

The specification of the entire business rule set showed that all the business rules could be specified with the established patterns. However, not every pattern from the 19 established patterns was required to specify the obtained business rule set. Table 6.3 below shows which patterns are validated with the obtained business rule set (see left column), and which patterns are not validated (see right column).

Validated patterns (9 patterns in total)	Not validated (10 patterns in total)
<b>pattern 1</b> (equate with value)	<b>pattern 5</b> (consistency check - multiple values)
<b>pattern 2</b> (equate with subject)	<b>pattern 6</b> (consistency check - one subject)
<b>pattern 3</b> (equate with basic ground)	<b>pattern 7</b> (consistency check - multiple subjects)
<b>pattern 4</b> (consistency check one value)	<b>pattern 10</b> (comparison with basic ground)
<b>pattern 8</b> (comparison with value)	<b>pattern 14</b> (conjunction within disjunction)
<b>pattern 9</b> (comparison with subject)	<b>pattern 15</b> (exact amount)
<b>pattern 11</b> (conjunction)	<b>pattern 16</b> (exact amount within conjunction)
<b>pattern 12</b> (disjunction)	<b>pattern 17</b> (exact amount within disjunction)
<b>pattern 13</b> (disjunction within conjunction)	<b>pattern 18</b> (conjunction within exact amount)
	<b>pattern 19</b> (disjunction within exact amount)

**Table 6.3:** Overview of Validated and Not Validated Patterns

In conclusion, validation round one showed that all the 45 business rules of the case company business rule set could be specified by means of the pattern catalogue. From the in total 19 patterns, 9 patterns could be validated and 10 patterns could not be validated during this validation round. However, for each of the 10 not validated patterns an example business rule is found during the establishment process of the patterns, which could be specified by means of these patterns.

## 7 Discussion

This chapter provides a critical perspective on the performance of this thesis project. First, the contributions of the research will be discussed. Subsequently, the limitations of the research and recommendations for further research are provided.

Considering the social contributions, this research provides further insight into the fundamental constructs of derivation business rules. With regard to these fundamental constructs, a lot of different terms to refer to similar concepts are found during this research. This finding is addressed in this thesis by listing and analyzing the available terminology options of each concept to provide insight for other researchers and practitioners. The preference for the application of a particular term will differ per author. Although a fixed name is chosen for each fundamental construct for this research, changing the terminology in the future will have no effect on the syntax and semantics of the deliverables. In the BRM domain, researchers already conducted studies to compare various business rule languages from a high abstraction level. The knowledge from this research can support the comparison and evaluation of business rule languages and systems from a more detailed view, since fundamental constructs and patterns as such provide independence from the implementation technology (Van der Aalst et al., 2003). Especially the similarities and differences with respect to their expressive power can be discovered. By taking the CNL as reference point and investigating if a certain language or system is able to express all the included fundamental constructs, an indication of the degree of expressiveness can be obtained. The comparison of the different levels of expressive power provides valuable information for an organization, as it gives in-depth insight into the suitability of a certain language or system for their organizational context. As a result, this can create awareness in the organization about the implementation consequences before a language or system is selected. So, this research can support organizations in the decision making process for selecting a business rule language and/ or system which can be a difficult task due to the multitude of commercially available options.

With regard to the scientific contribution, a few attempts have already been made to create CNLs and pattern catalogues for the specification of business rules (see section 1.1 and 2.3). These existing approaches are either not precise enough to be interpreted by an information system or they are especially intended for the type of business rules that guide business processes. Therefore, a CNL is created during this research which had to comply with specific language properties (i.e. PENS levels). These PENS levels were predefined as follows: Precision= 4, Expressiveness= 3, Naturalness= 3, and Simplicity = 4. To check if the created CNL complies with these predefined levels, the set of criteria defined by Kuhn is taken into account (see Appendix 2). Firstly, to reach a *precision-level* of 4, the language should be fully formal on the syntactic level and it should be possible to parse each text specified with the language to a formal logic representation. Both criteria are met by means of formulating a formal underlying grammar for the CNL. Secondly, an *expressiveness-level* of 3 is reached since the CNL conforms to the following criteria: the language is able to specify general rule structures (i.e. if/then), negation, and relations of arity greater than 1 (e.g. binary relations). Thirdly, the CNL meets a *naturalness-level* of 3 by adhering to the following criteria: natural elements predominate over unnatural ones, general structure equals to natural language grammar, and untrained readers are able to intuitively understand statements specified with the language. Lastly, the CNL complies with a *simplicity-level* of 4 because more than one page but not more than ten pages are required to provide an exact and comprehensive description of the language. By creating this CNL and a pattern catalogue specifically applicable for derivation business rules, this research has a scientific contribution by adding a new type of CNL and pattern catalogue to the scientific knowledge base.

Besides the contributions of this research, also a number of limitations can be appointed concerning the validation of the research. In terms of external validity, concerning the generalizability of the results, some remarks can be made. Firstly, the data set that is used for the validation of the pattern catalogue is derived from the Dutch Tax and Customs Administration. Given the fact that only one

company is taken into account, the pattern catalogue cannot be generalized for other organizations which limits the external validity. Secondly, not all the devised patterns of the pattern catalogue could be validated with this real life case study data which could also limit generalizability. However, it should be noted that the patterns that were not validated, have been found in other cases substantiating their inclusion in the catalogue. Another point of discussion attached to the validation of the results is that it is not certain whether the co-founding variables are eliminated. Since the majority of the data analysis process of each validation round was conducted by solely one researcher, there is a possibility that the internal validity is threatened by the so called ‘instrumentation threat’. In this case, the researcher is considered as the measurement device which could gain experience gradually traversing the four validation rounds. To address this possible threat, a reliability coder was involved for one of the data analysis processes with respect to the validation of the fundamental constructs.

Considering the limitations of the research, it is recommended to extend the research in the future since it will provide a broader supported and improved CNL and pattern catalogue. The first limitation with regard to the external validity could be addressed by altering the sampling strategy and selection criteria for further research. For instance, additional data (e.g. business rules) could be gathered originating from different industries for new validation rounds. Furthermore, additional business rules management systems could be included for further validation. In this way, insight into the applicability of the artifacts in other industries and for other systems could be obtained thereby increasing generalizability. The limited generalizability could also be enhanced by increasing the sample sizes. As described above, a part of the patterns of the pattern catalogue could not be validated with the case study company data. It is recommended to expand the learning data to validate the entire set of patterns for purpose of generalization. Although the four performed validation rounds and the amount of used input data for each round are considered as sufficient (i.e. 37 patterns, 150 business rules, 6 systems, 45 business rules from the case company), the size of each data set could be increased for further research to enhance the generalization of the results even further.

Another recommendation that can be given for further research is to perform the third validation phase from Wieringa (2013) called “field testing”. This last validation phase is left out during this research due to time constraints (see section 1.4), since it would require that the artifacts had actually been applied in practice on a large scale. During this research the focus lied on demonstrating that it was possible to specify a business rule set once which could then be transformed into various implementation dependent languages. This is one of the four positive effects of the research for the Dutch Tax and Customs Administration. As already described in section 1.4, the three other positive effects could not be verified during this research due to time constraints. Future research could investigate the effect of the usage of the CNL and pattern catalogue with respect to the amount of errors, the time spent and the corresponding amount of money. Moreover, it would be very valuable to investigate the automatic transformation by means of a parser, which is now demonstrated on paper. Since the overall aim of the field testing phase is to see if an artifact would produce the desired effects when it will be transferred to the market, it is advisable to perform this phase during further research to be able to verify the other three positive effects. In addition, when this field testing phase would be performed, also some relevant quality attributes could be verified. According to Hevner et al. (2004), different quality attributes can be appointed to evaluate the devised artifacts of a design-science research like: consistency, reliability, usability, etc. Considering the pattern catalogue as one of the artifacts of this design-science research, especially the mutual exclusiveness and completeness criteria are taken into account. Further research could for example reveal if the pattern catalogue is usable for business rule authors.

The gained knowledge and resulting artifacts of this research can serve as basis for future research to further improve or extend the created CNL and pattern catalogue. However, this knowledge could also be used and expanded by investigating if these artifacts are applicable for other types of business rules besides derivation business rules.

## 8 Conclusion

In the last ten to fifteen years, an abundance of different business rules management systems and related business rule languages are created to capture and manage derivation business rules. Derivation business rules are *“expressions that evaluate facts, by means of a calculation or classification, leading to a new fact”*. The abundance of available systems and languages, and the fact that they differ to a large extent regarding their expressive power, causes two problems. The first problem organizations may encounter are difficulties in selecting an appropriate business rules management system or business rule language, since no set of criteria exists which could be used as reference point for comparison. This can for instance lead to the selection of a language with a too extensive or too low level of expressive power. A second problem can arise when organizations have selected a language tailored to a particular system and then add or transfer to a new business rules management system. In this case, the entire business rule set has to be re-specified to be readable for this new system. This process can be very inefficient, expensive and error prone. These problems are formulated by means of the following problem statements:

- *“How can the problem be addressed that no tailored set of formal requirements exist, which can be used to verify if a business rule language is able to formulate derivation rules?”*
- *“How can the problem be addressed that business rules need to be re-modeled to comply with a new implementation dependent language?”*

To address the above mentioned problems, this research has been conducted for which the following main research question was formulated: *“How can derivation business rules be specified precisely and implementation independent?”* In order to answer this question, four sub-questions need to be answered first. The first sub-question is:

*SQ1: “Which notation forms can be used to specify derivation business rules?”*

Various business rule notation forms to specify derivation business rules are available. Two main formalism types exist: implementation dependent and implementation independent languages. The first type complies with a specific grammar which can only be processed by a particular system, resulting in the occurrence of the second problem statement. In contrast, an implementation independent language could be applied in multiple environments but is generally not precise enough to be directly executable. To address this gap, a controlled natural language (CNL) emerged from literature as applicable notation form. A CNL can comply with a high precision level and remain implementation independent at the same time. On the other hand, a CNL can resemble a natural language. In order to create a CNL for specifying derivation business rules, the fundamental constructs (i.e. building blocks of the language) and an underlying formal grammar needed to be established which is done by answering the next two sub-questions:

*SQ2: “Which fundamental constructs are necessary to construct a precise derivation business rule?”*

In literature, the following 15 fundamental constructs were identified which are necessary to be able to construct a precise derivation business rule: 1) Conclusion part, 2) Condition part, 3) Modal Claim Type, 4) Construct, 5) Connective, 6) Expression, 7) Subject, 8) Quantifier, 9) Relation, 10) Ground, 11) Classification, 12) Propositional Operator, 13) Value, 14) Mathematical Operator, and 15) Mathematical Function. This set of fundamental constructs responds to the first deliverable of the research. This deliverable could be used as reference point to assess and compare the precision level of business rules languages, addressing the first mentioned research problem.

Subsequently, it was significant to understand how these fundamental constructs were related by answering the third sub-question:



***SQ3: “Which grammar rules should be enforced on the fundamental constructs to specify precise derivation business rules?”***

Answering sub-question three, resulted in the establishment of a set of 40 grammar rules, which corresponds to the second deliverable. Given the precision requirement of the specification language, the grammar rules comply with a specific type of grammar namely a formal grammar. The aim of the grammar rules is to impose the syntax of the fundamental constructs for precision reasons. The fundamental constructs together with the grammar rules were incorporated in a meta-model.

To validate the fundamental constructs and grammar rules (i.e. meta-model), three validation rounds have been performed. During the first validation round, 37 business rule patterns from five different existing pattern catalogues were mapped onto the fundamental constructs. During the second validation round, 150 business rules from 11 different cases were mapped onto the fundamental constructs. In the third validation round, implemented business rules and components of six business rules management systems were mapped. The three validation rounds revealed that: 1) no fundamental constructs lacked, and 2) no fundamental construct was superfluous. In conclusion, this showed that all the selected fundamental constructs were significant to retain.

By answering the first three sub-questions, the necessary notation form, fundamental constructs and grammar rules were identified to specify precise derivation business rules. To make the envisioned CNL even more restrictive and precise, sub-question four was answered:

***SQ4: “Which patterns can be identified for specifying derivation business rules?”***

From the literature study and validation rounds, repeating elements and their relations came apparent. This knowledge together with the meta-model was used to devise the third deliverable: a set of 19 different patterns. One requirement to create these patterns was mutual exclusiveness, which implies that a business rule author can only select one pattern for each specific business rule.

To check the completeness of the created pattern catalogue, it was validated if all the business rules from the case study company data set could be re-specified by means of the patterns. The validation showed that each of the 45 business rules from the Dutch Tax and Customs Administration could be specified with the pattern catalogue.

Previously mentioned sub-questions were specified according to the following main research question:

***RQ: “How can derivation business rules be specified precisely and implementation independent?”***

The answer to the main research question is by using a CNL which consists of: 1) a set of fundamental constructs, 2) an underlying formal grammar, 3) a meta-model, and 4) a set of patterns.

In conclusion, the intended purpose of this research has been achieved through the creation of a CNL and pattern catalogue which can be used to specify a set of derivation business rules once, and which allows automatic transformation of the business rule set to be applicable for different business rule engines. The created artifacts are considered as the basis for further research.

## References

- Aikawa, T., Schwartz, L., King, R., Corston-Oliver, M., & Lozano, C. (2007). Impact of Controlled Language on Translation Quality and Post-editing in a Statistical Machine Translation Environment. *Proceedings of the MT Summit XI*, Copenhagen, 1-7.
- Alexander, C., Ishikiwa, S., & Silverstein, M. (1977). *A Pattern Language: Towns, Buildings, Construction*. Oxford: Oxford University Press.
- Bastun, Y. (2015). *Decision Model: "Vehicle Insurance – UServ Auto Insurance Product Derby" using OpenL Tablets (v1.0)*. <https://dmcommunity.wordpress.com/challenge/>: EIS Group.
- BBC. (2015). *Vehicle Insurance – UServ Product Derby*. from <http://www.businessrulesforum.com/>
- Blumberg, B., Cooper, D. R., & Schindler, P. S. (2011). *Business Research Methods*. London: McGraw-Hill Education.
- Boyer, J., & Mili, H. (2011). *Agile Business Rules Development: Process, Architecture and JRules Examples*. Heidelberg: Springer.
- British Council. (2015). *English Grammar*. from <https://learnenglish.britishcouncil.org/en/english-grammar/>
- Bryman, A., & Bell, E. (2003). *Business Research Methods*. Oxford: Oxford University Press.
- Caron, F., Vanthienen, J., & Baesens, B. (2013). Comprehensive Rule-Based Compliance Checking and Risk Management with Process Mining. *Decision Support Systems*, 54(3), 1357-1369.
- Ceri, S., & Fraternal, P. (1997). *Designing Database Applications with Objects and Rules: the IDEA Methodology*. New York: Addison-Wesley.
- Chamberlin, D. D., & Boyce, R. F. (1974). SEQUEL: A Structured English Query Language. *Proceedings of the 1974' ACM SIGFIDET Workshop on Data Description, Access and Control*, Ann Arbor, Michigan, 249-264.
- Chervak, S., Drury, C. G., & Ouellette, J. P. (1996). Field Evaluation of Simplified English for Aircraft Workcards. *Proceedings of the Tenth FAA/AAM Meeting on Human Factors in Aviation Maintenance and Inspection*, Alexandria, Virginia, 1-16.
- Clark, P., Harrison, P., Jenkins, T., Thompson, J. A., & Wojcik, R. H. (2005). Acquiring and Using World Knowledge Using a Restricted Subset of English. *Proceedings of the Eighteenth International FLAIR Conference*, Florida, 506-511.
- Dijkstra, E. W. (1982). On the role of scientific thought. In E. W. Dijkstra (Ed.), *Selected Writings on Computing: A Personal Perspective* (pp. 60-66). New York: Springer.
- DM Community. (2015). *Challenges*. from <https://dmcommunity.wordpress.com/>
- do Prado Leite, J. C. S., & Leonardi, M. C. (1998). Business Rules as Organizational Policies. *Proceedings of the Ninth International Workshop on Software Specification and Design*, Ise-Shima, 68-76.
- Earley, J. (1970). An Efficient Context-Free Parsing Algorithm. *Communications of the ACM*, 13(2), 94-102.
- FAA. (2014). *Air Traffic Control (Order JO 7110.65T)*. <https://www.faa.gov/documentLibrary/media/Order/ATC.pdf>: Federal Aviation Administration (FAA).
- Feldman, J. (2011). *Preparing a Tax Return (Release 6.1)*. <http://openrules.com/pdf/Tutorial.Dialog1040EZ.pdf>: OpenRules.
- Feldman, J. (2014). *Determine Patient Therapy. (Release 6.3.0)*. <http://openrules.com/pdf/Tutorial.DecisionPatientTherapy.pdf>: OpenRules.
- Feldman, J. (2015). *Decision Model "Vehicle Insurance – UServ Product Derby" (v1.0)*. <https://dmcommunity.wordpress.com/challenge/>: OpenRules.
- Gallier, J. (2011). *The Theory of Languages and Computation*. Pennsylvania: University of Pennsylvania.
- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-oriented Software*. Boston: Addison-Wesley.



- Ghose, A., & Koliadist, G. (2007). Auditing Business Process Compliance. *Proceedings of the Fifth International Conference on Service Oriented Computing*, Vienna, 169-180.
- Google. (2015). *Voice Actions*. from <https://developers.google.com/voice-actions/>
- Graham, I. (2006). *Business Rules Management and Service Oriented Architecture*. New York: Wiley.
- Hallett, C., Scott, D., & Power, R. (2007). Composing Questions through Conceptual Authoring. *Computational Linguistics*, 33(1), 105-133.
- Hay, D., & Healy, K. (2000). *Defining Business Rules: What are they really?* (Revision 1.3). [http://www.businessrulesgroup.org/first\\_paper/BRG-whatBR\\_3ed.pdf](http://www.businessrulesgroup.org/first_paper/BRG-whatBR_3ed.pdf): the Business Rules Group.
- Herbst, H. (1996). Business Rules in Systems Analysis: A Meta-model and Repository System. *Information Systems*, 21(2), 147-166.
- Herbst, H. (1997). *Business Rule-Oriented Conceptual Modeling*. Heidelberg: Physica-Verlag.
- Hevner, A., March, S., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75-105.
- Hoppenbrouwers, S. (2011). *RuleSpeak grammar*. Radboud University Nijmegen.
- HP. (2015). *IF Construct*. from [h21007.www2.hp.com/portal/download/files/unprot/fortran/docs/lrm/lrm0143.htm](http://h21007.www2.hp.com/portal/download/files/unprot/fortran/docs/lrm/lrm0143.htm)
- Humayoun, M., & Raffalli, C. (2010). Mathnat -Mathematical Text in a Controlled Natural Language. *Research in Computing Science — Special issue: Natural Language Processing and its Applications*, 46, 293-307.
- Iacob, M. E., Lankhorst, M. M., & Schrier, A. (2012). Patterns for Agility. In M. M. Lankhorst (Ed.), *Agile Service Development* (pp. 95-110). Heidelberg: Springer.
- Inglesant, P., Sasse, M. A., Chadwick, D., & Shi, L. L. (2008). Expressions of Expertness: the Virtuous Circle of Natural Language for Access Control Policy Specification. *Proceedings of the Fourth Symposium on Usable Privacy and Security*, Pittsburgh, 77-88.
- Johnson, E. (2000). Talking across Frontiers. *Proceedings of the International Conference on European Cross Border Cooperation: Lessons for and from Ireland*, Belfast, 1-23.
- Kramer, M. I. (1997). Business Rules: Automating Business Policies and Practices. *Distributed Computing Monitor*.
- Kuhn, T. (2010). *Controlled English for Knowledge Representation*. Doctoral Thesis. Zurich: University of Zurich.
- Kuhn, T. (2013). A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, 40(1), 1-50.
- Lee, A. S., & Baskerville, R. L. (2003). Generalizing Generalizability in Information Systems Research. *Information Systems Research*, 14(3), 221-243.
- Liao, S.-H. (2004). Expert System Methodologies and Applications - A Decade Review from 1995 to 2004. *Expert Systems with Applications*, 28(1), 93-103.
- Ligêza, A. (2006). *Logical Foundations for Rule-Based Systems*. Heidelberg: Springer.
- Mahoney, J. (1999). Nominal, Ordinal, and Narrative Appraisal in Macrocausal Analysis. *American Journal of Sociology*, 104(4), 1154-1196.
- March, S. T., & Smith, G. F. (1995). Design and Natural Science Research on Information Technology. *Decision Support Systems*, 15(4), 251-266.
- Massachusetts Senate. (2003). *Massachusetts Legislative Drafting Language* (Legislative Drafting and Legal Manual 3rd edition). [www.legislationline.org/documents/id/3642](http://www.legislationline.org/documents/id/3642): Massachusetts Senate.
- Mays, N., & Pope, C. (1995). Qualitative Research: Rigour and Qualitative Research. *BMJ*, 311(6997), 109-112.
- McGrath, J. E. (1981). Dilemmatics: The Study of Research Choices and Dilemmas. *American Behavioral Scientist*, 25(2), 179-210.
- Mill, J. (1906). *A system of Logic*. London: Longmans Green.



- Mitamura, T., & Nyberg, E. (1995). Controlled English for knowledge-based MT: Experience with the KANT system. *Proceedings of the Sixth International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven, 146-147.
- Morgan, T. (2002). *Business Rules and Information Systems: Aligning IT with Business Goals*. London: Addison-Wesley.
- Necas, I. (2011). *BDD as a Specification and QA Instrument*. Master Thesis, Masaryk University, Brno.
- Nelson, M. L., Peterson, J., Rariden, R. L., & Sen, R. (2010). Transitioning to a business rule management service model: Case studies from the property and casualty insurance industry. *Information & Management*, 47(1), 30-41.
- O'Brien, S., & Roturier, J. (2007). How Portable are Controlled Language Rules? A Comparison of Two Empirical MT Studies. *Proceedings of the MT Summit XI*, Copenhagen, 345-352.
- Object Management Group. (2008). *Semantics of Business Vocabulary and Business Rules (SBVR)* (v1.0). <http://www.omg.org/spec/SBVR/1.0/PDF>: Object Management Group.
- Object Management Group. (2013). *Semantics of Business Vocabulary and Business Rules (SBVR)* (v1.2). <http://www.omg.org/spec/SBVR/1.2/>: Object Management Group.
- Ortiguela, R. (2015). *Challenge: Vehicle Insurance – UServ Product Derby* (v1.0). Decide.
- Parish, M. (2014). *Rule Modeling Case Study Generic Diabetic Monitoring* (v1.0). <https://dmcommunity.wordpress.com/case-studies/#DiabeticPatientMonitoring>: Progress Software.
- Parish, M. (2015). *USERV Auto Insurance Rule Model in Corticon* (v1.0). <https://dmcommunity.wordpress.com/challenge>: Progress Software.
- Park, C., & Choi, I. (2004). Management of business process constraints using BPTrigger. *Computers in Industry*, 55(1), 29-51.
- Pease, A., & Li, J. (2010). Controlled English to Logic Translation. In R. Poli, M. Healy & A. Kameas (Eds.), *Theory and Applications of Ontology: Computer Applications* (pp. 245-258). Netherlands: Springer.
- Pesic, M., & van der Aalst, W. M. (2006). A Declarative Approach for Flexible Business Processes Management. *Proceedings of the Fourth International Business Process Management Conference*, Vienna, Austria, 169-180.
- Peters, S., & Westerståhl, D. (2006). *Quantifiers in Language and Logic*. Oxford: Oxford University Press.
- Pool, J. (2006). Can Controlled Languages Scale to the Web? *Proceedings of the Fifth International Workshop on Controlled Language Applications*, Cambridge, 1-12.
- Power, R. (2012). OWL Simplified English: a finite-state language for ontology editing. In T. Kuhn & N. E. Fuchs (Eds.), *Controlled Natural Language* (pp. 44-60). Zurich: Springer.
- Ranta, A. (2014). Embedded Controlled Languages. In B. Davis, K. Kaljurand & T. Kuhn (Eds.), *Controlled Natural Language* (pp. 1-7). Switzerland: Springer.
- Recker, J. C., Indulska, M., Rosemann, M., & Green, P. (2005). Do Process Modelling Techniques Get Better? A Comparative Ontological Analysis of BPMN. *Proceedings of the Sixteenth Australasian Conference on Information Systems*, Sydney, 1-10.
- Rosca, D., Greenspan, S., Feblowitz, M., & Wild, C. (1997). A Decision Making Methodology in Support of the Business Rules Lifecycle. *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, 236-246.
- Ross, R. G. (1987). *Entity Modeling: Techniques and Application*. Boston: Database Research Group.
- Ross, R. G. (2000). What Are Fact Models and Why Do You Need Them (Part 1). *Business Rules Journal*, 1(5).
- Ross, R. G. (2000). What Are Fact Models and Why Do You Need Them (Part 2). *Business Rules Journal*, 1(7).
- Ruffino, J. R. (1982). Coping with Machine Translation. In V. Lawson (Ed.), *Practical Experience of Machine Translation* (pp. 57-60). Amsterdam: North-Holland Publishing Company.
- Sangers-van Cappellen, G. (2014). *RegelSprak* (v3.2). Belastingdienst.



- Schadd, M. (2015). *Case Study: Vehicle Insurance UServ Product Derby (v1.0)*.  
[https://dmcommunity.wordpress.com/challenge: Blueriq](https://dmcommunity.wordpress.com/challenge:Blueriq).
- Segal, G. (2015). *UServ Product Derby Case Study (v1.0)*.  
[https://dmcommunity.wordpress.com/challenge: Sapiens Decision](https://dmcommunity.wordpress.com/challenge:SapiensDecision).
- Selfridge, P. G., Waters, R. C., & Chikofsky, E. J. (1993). Challenges to the field of reverse engineering. *Proceedings of the Working Conference on Reverse Engineering*, Baltimore, 144-150.
- Shubert, S., Spyridakis, J. H., Holmback, H. K., & Coney, M. B. (1995). The Comprehensibility of Simplified English in Procedures. *Journal of Technical Writing and Communication*, 25(4), 347-369.
- Sun, Y., Demirezen, Z., Mernik, M., Gray, J., & Bryant, B. (2008). Is My DSL a Modeling or Programming Language? *Proceedings of the Second International Workshop on Domain-Specific Program Development*, Nashville, 1-5.
- Temnikova, I. (2010). Cognitive Evaluation Approach for a Controlled Language Post-Editing Experiment. *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, Malta, 3485-3490.
- The Free Dictionary. (2015). *definition modality*. from <http://www.thefreedictionary.com/modality>
- Van der Aalst, W. (1996). Three Good Reasons for Using a Petri-net-based Workflow Management System. *Proceedings of the International Working Conference on Information and Process Integration in Enterprises*, Cambridge, 179-201.
- Van der Aalst, W., Ter Hofstede, A., & Weske, M. (2003). Business Process Management: A Survey. *Proceedings of the First International Conference on Business Process Management*, Eindhoven, Nederland, 1-12.
- Van der Aalst, W., & Ter Hofstede, A. H. (2005). YAWL: yet another workflow language. *Information Systems*, 30(4), 245-275.
- Van der Aalst, W., Ter Hofstede, A. H., Kiepuszewski, B., & Barros, A. P. (2003). Workflow Patterns. *Distributed and parallel databases*, 14(1), 5-51.
- Van Deursen, A., & Klint, P. (2002). Domain-Specific Language Design Requires Feature Descriptions. *Journal of Computing and Information Technology*, 10(1), 1-17.
- Van Deursen, A., Klint, P., & Visser, J. (2000). Domain-Specific Languages: An Annotated Bibliography. *Sigplan Notices*, 35(6), 26-36.
- Verschuren, P., & Doorewaard, H. (2007). *Het ontwerpen van een onderzoek*. Amsterdam: Lemma.
- Versendaal, J. (1991). *Separation of the User Interface and Application*. Doctoral Thesis. Rotterdam: Technische Universiteit Delft.
- Von Halle, B. (1994). Back to Business Rule Basics. *Database Programming & Design*, 15-18.
- Von Halle, B. (2001). *Business Rules Applied: Building Better Systems Using the Business Rules Approach*. New York: Wiley.
- Von Halle, B., & Goldberg, L. (2009). *The Decision Model: A Business Logic Framework Linking Business and Technology*. London: CRC Press.
- Wan-Kadir, W., & Loucopoulos, P. (2003). Relating Evolving Business Rules to Software Design. *Proceedings of the International Conference on Software Engineering Research and Practice*, Las Vegas, USA, 1-5.
- Wan-Kadir, W., & Loucopoulos, P. (2004). Relating Evolving Business Rules to Software Design. *Journal of Systems Architecture*, 50(7), 367-382.
- Wand, Y., & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. *Information Systems Journal*, 3(4), 217-237.
- Weske, M. (2007). *Business Process Management - Concepts, Languages, Architectures*. New York: Springer.
- Wieringa, R. (2013). Empirical research methods for technology validation: Scaling up to practice. *Journal of Systems and Software*, 95, 19-31.
- Wohed, P., Van der Aalst, W. M., Dumas, M., Ter Hofstede, A. H., & Russell, N. (2006). On the Suitability of BPMN for Business Process Modelling. *Proceedings of the Fourth International Conference on Business Process Management*, Vienna, 161-176.



- Wong, P. C., Whitney, P., & Thomas, J. (1999). Visualizing Association Rules for Text Mining. *Proceedings of the IEEE Symposium on Information Visualization*, San Francisco, 1-5.
- Zoet, M. (2014). *Methods and Concepts for Business Rules Management*. Doctoral Thesis. Utrecht: Universiteit Utrecht.
- Zoet, M., Ravesteyn, P., & Versendaal, J. (2011). A Structured Analysis of Business Rules Representation Languages: Defining a Normalization Form. *Proceedings of the Twentieth Australasian Conference on Information Systems*, Sydney, 1-10.
- Zoet, M., & Versendaal, J. (2013). Business Rules Management Solutions Problem Space: Situational Factors. *Proceedings of the Seventeenth Pacific Asia Conference on Information Systems*, Jeju, 1-13.
- Zur Muehlen, M., & Indulska, M. (2010). Modeling languages for business processes and business rules: A representational analysis. *Information Systems*, 35(4), 379-390.

## Appendix 1: Business Rule Classification Schemes

The business rule classification schemes that are found in literature are listed in the table below.

Source	Business rule class / type	Description	Sub-class / type	Description
Boyer & Mili (2011)	Structural rules	“Structural rules define the terms used by the business in expressing their business rules and the relationships (facts) among those terms. These include the vocabulary used in rule authoring.”		
	Operational rules	“Operational rules are the rules that implement business decision logic. They are the individual statements of business logic that are evaluated by the rule engine to determine the decision result.”	ProcessFlow	“Process flow routing rules direct the movement through a process flow or workflow.”
			Inference	“Inference rules create new objects or facts which may bring the engine to re-evaluate some other rule’s eligibility.”
			Guideline	“Rules that does not reject the transaction; they merely warn about an undesirable circumstance.”
			Mandatory constraints	“Rules that reject the attempted business transaction.”
			Computation	“Computation rules implement mathematical equations and assign values to variables according to a set of given criteria.”
			ActionEnabler	“Action enabler rules modify, create, or delete terms or association between terms, or execute methods. These rules test conditions and upon finding



Source	Business rule class / type	Description	Sub-class / type	Description
				them true, initiate another business event, message, business process or other activity."
			ECA	"Rules where the condition is evaluated once the occurrence of an event is found."
SBVR (OMG, 2013)	Structural (definitional) rule	"Rule that is a claim of necessity."		
	Operative (behavioral) business rule	"Business rule that is a claim of obligation."		
RuleSpeak (OMG, 2008)	Structural rules	"prescribe criteria for how the business chooses to organize ("structure") its business semantics.		
	Operative business rules	"focus directly on the propriety of conduct in circumstances (business activity) where willful or uninformed actions can fall outside the boundaries of behavior deemed acceptable. Unlike structural rules, operative rules can be violated <i>directly</i> ."		
Von Halle (2001)	Mandatory constraints	" a complete statement that expresses an unconditional circumstance that must be true or not true for the business event to complete with integrity."		
	Guidelines	"A complete statement that expresses a warning about a circumstance that should be true or not true. A guideline does not force the circumstance to be true or not true, but merely warns about it, allowing the human to make the decision."		
	Action-enablers	"a complete statement that tests conditions and upon finding them true, initiates		





Source	Business rule class / type	Description	Sub-class / type	Description
		another business event, message, or other activity. That is, an action enabler initiates a new action external to the scope of the system or increment under study."		
	Computations	"a complete statement that provides an algorithm for arriving at the value of a term where such algorithms may include sum, difference, product, quotient, count, maximum, minimum, average."		
	Inferences	"a complete statement that tests conditions and upon finding them true, establishes the truth of a new fact."		
Hay & Healy (2000)	Structural assertion	"a defined concept or a statement of a fact that expresses some aspect of the structure of an enterprise. This encompasses both terms and the facts assembled from these terms."		
	Action assertion	"A statement of a constraint or condition that limits or controls the actions of the enterprise."		
	Derivation	"A statement of knowledge that is derived from other knowledge in the business."		
Zoet (2014)	Structural Sequencing	"A Structural Sequencing Rule (SSR) is defined as a rule that influences the structural execution position of process elements. Each business process has an underlying blueprint indicating the sequence by which activities, events and decision elements (process elements) are executed."		
	Actor Inclusion	"An Actor Inclusion Rule (AIR) defines a rule that		



Source	Business rule class / type	Description	Sub-class / type	Description
		stating which process element an actor can or cannot execute."		
	Transactional Sequencing	"A Transactional Sequence Rule (TSR) defines a rule that influences the decision of an individual process instance based on the case at hand."		
	Data Condition	"Data Condition Rule (DCR) defines: 1) what data needs to be stored, 2) how the data is stored, 3) how long the data is stored, 4) and which authorizations are required concerning the access and modification of the data."		
	Outcome Control	"An Outcome Control Rule (OCR) is a rule that defines how results from process elements (undesirable or desirable) occurring in business processes are identified."		
Wan-Kadir & Loucopoulos (2004)	Constraint	"A Constraint rule is used to check for the result of the execution of business event on a Subject."	Mandatory Constraint	"A statement that specifies a mandatory feature (business behaviour or characteristics) that must be satisfied by a business entity."
			Guideline	"A statement that specifies an optional feature that should be satisfied by a business entity Upon the violation of this rule, system only raises a warning instead of rejecting the transaction."
	Action Assertion	"An action assertion rule is a statement that concerns a dynamic aspect of the business. It specifies the action that should be activated on the occurrence	Enabler	"Enabler rule enables or disables a rule, operation, process, or procedure according to certain conditions. It also



Source	Business rule class / type	Description	Sub-class / type	Description
		of a certain event or on satisfaction of a certain condition."		creates and deletes data under specified conditions."
			Copier	"Copier is concerned with the use of existing data or value, for example, using a certain value to set the initial value of an object's attributes or to determine the way on how to present existing data."
			Trigger	"Trigger is a rule that causes operation, process, procedure, or rule to be executed when the given condition is true or on the occurrence of a certain event."
	Derivation	"Derivation is a rule that derives a new fact based on the existing terms and facts."	Computation	"A statement that derives a value using an algorithm."
			Inference	"A statement that derives a fact using logical deduction or induction."
Leite & Leonardi (1998)	Functional rules	"Functional rules are general policies regarding organization functionality."		
	Non-functional business rules	"Non-functional business rules describe policies regarding constraints that the organization must follow."	Macrosystem rules	"This type of rule describes policies that are related to the specific characteristics of a Universe of Discourse. It relates Universe of Discourse concepts in order to impose a constraint on the organization."

Source	Business rule class / type	Description	Sub-class / type	Description
			Quality rules	“Quality rules are demands of an organization on the characteristics of its processes or products. They usually reflect general policies related to quality standard or quality expectations of an organization.”
Caron et al. (2013)	Cardinality-based rules	“Business rules that restrict the number of allowed instances of a specific process element type in a specific process instance.”		
	Coexistence rules	“Business rules that restrict the coexistence of process elements of different types over the execution of a specific process instance.”		
	Dynamic data-driven rules	“Business rules that specify the influence of specific data elements and their value on the occurrence of process elements in a specific process instance.”		
	Relative time rules	“Business rules that focus on specifying a time restriction on process elements relative to certain points in a process execution, for example the start of a process or the completion of a specific activity.”		
	Static property rules	“Business rules that deal with specifying a specific property for a particular type of process element at a predefined process state.”		

## Appendix 2: Detailed Explanation Pens

The table below includes a description of each of the five ranks per PENS dimension. This ranking is adapted from Kuhn (2013).

Dimension	Rank / Degree	Explanation / criteria
Precision	Imprecise languages (P1)	<ul style="list-style-type: none"> <li>• “Virtually every sentence of these languages is vague to a certain degree.</li> <li>• Without taking context into account, most sentences of a certain complexity are ambiguous.</li> <li>• The automatic interpretation of such languages is ‘AI-complete’.</li> <li>• Require a human reader to check syntax and meaning of Statements.”</li> </ul>
	Less imprecise languages (P2)	<ul style="list-style-type: none"> <li>• “Less ambiguity and vagueness than in natural languages.</li> <li>• Interpretation depends much less on context.</li> <li>• Restrict the use and/or the meaning of a wide range of the ambiguous, vague, or context-dependent constructs.</li> <li>• Restrictions are not sufficient to make automatic interpretation reliable.</li> <li>• No formal (i.e., mathematically precise) underpinning.”</li> </ul>
	Reliably interpretable languages (P3)	<ul style="list-style-type: none"> <li>• “Heavily restricted syntax (not necessarily formally defined).</li> <li>• Reliable automatic interpretation.</li> <li>• Logical underpinning or formal conceptual scheme to represent semantics.</li> <li>• No fully formalized mapping of sentences to their semantic representations.</li> <li>• External background knowledge, heuristics, or user feedback are required.”</li> </ul>
	Deterministically interpretable languages (P4)	<ul style="list-style-type: none"> <li>• “Fully formal on the syntactic level (can be defined by a formal grammar).</li> <li>• Parse deterministically to a formal logic representation (or a small closed set of all possible representations).</li> <li>• Representations may be underspecified: they may require certain parameters, background axioms, external resources, or heuristics to enable sensible deductions.”</li> </ul>
	Languages with fixed semantics (P5)	<ul style="list-style-type: none"> <li>• “Fully formal and fully specified on syntactic and semantic levels.</li> <li>• Each text has exactly one meaning, which can be automatically derived.</li> <li>• The circumstances in which inferences hold or do not hold are fully defined.</li> <li>• No heuristics or external resources are necessary.”</li> </ul>
Expressiveness	Inexpressive languages (E1)	<ul style="list-style-type: none"> <li>• “No universal quantification, or</li> </ul>



		<ul style="list-style-type: none"> <li>No relations of arity greater than 1 (e.g., binary relations)."</li> </ul>
	Languages with low expressiveness (E2)	<ul style="list-style-type: none"> <li>"Universal quantification over individuals (possibly limited).</li> <li>Relations of arity greater than 1 (e.g., binary relations).</li> <li>Are not E3-languages."</li> </ul>
	Languages with medium expressiveness (E3)	<ul style="list-style-type: none"> <li>"General rule structures: if{then statements with multiple universal quantification that can target all argument positions of relations.</li> <li>Negation (strong negation or negation as failure).</li> <li>Have all features of E2.</li> <li>Are not E4-languages."</li> </ul>
	Languages with high expressiveness (E4)	<ul style="list-style-type: none"> <li>"General second-order universal quantification over concepts and relations.</li> <li>Have all features of E3.</li> <li>Are not E5-languages."</li> </ul>
	Languages with maximal expressiveness (E5)	<ul style="list-style-type: none"> <li>"Can express anything that can be communicated between two human beings.</li> <li>Cover any statement in any type of logic"</li> </ul>
Naturalness	Unnatural languages (N1)	<ul style="list-style-type: none"> <li>"Languages that do not look natural.</li> <li>Heavy use of symbol characters, brackets, or unnatural keywords.</li> <li>Use of natural words or phrases as names for certain entities might be possible, but is neither required nor further defined.</li> </ul> <p>NOTE: These are not CNLs according to Kuhn's definition."</p>
	Languages with dominant unnatural elements (N2)	<ul style="list-style-type: none"> <li>"Natural language words or phrases are an integral part.</li> <li>Dominated by unnatural elements or unnatural statement structure.</li> <li>Natural elements do not connect in a natural way to each other.</li> <li>Untrained readers fail to intuitively understand the statements.</li> </ul> <p>NOTE: These are not CNLs according to Kuhn's definition."</p>
	Languages with dominant natural elements (N3)	<ul style="list-style-type: none"> <li>"Natural elements are dominant over unnatural ones.</li> <li>General structure corresponds to natural language grammar.</li> <li>Sentences cannot be considered valid natural sentences.</li> <li>Untrained readers do not recognize the statements as well-formed sentences of their language, but are nevertheless able to intuitively understand them to a substantial degree."</li> </ul>



	Languages with natural sentences (N4)	<ul style="list-style-type: none"> <li>• “Valid natural sentences.</li> <li>• If natural flow is maintained, minor deviations are permitted, including text color, indentation, hyphenation, and capitalization.</li> <li>• Untrained readers recognize the statements as sentences of their language and are able to correctly understand their essence.</li> <li>• Single sentences have a natural flow, but not complete texts.”</li> </ul>
	Languages with natural texts (N5)	<ul style="list-style-type: none"> <li>• “Complete texts and documents can be written in a natural style and with a natural text flow.</li> <li>• For spoken languages, complete dialogs can be produced with a natural flow and a natural combination of speech acts.”</li> </ul>
Simplicity	Very complex languages (S1)	<ul style="list-style-type: none"> <li>• “Have the complexity of natural languages.</li> <li>• Cannot be described in an exact and comprehensive manner.”</li> </ul>
	Languages without exhaustive descriptions (S2)	<ul style="list-style-type: none"> <li>• “Considerably simpler than natural languages.</li> <li>• A significant part of the complex structures are eliminated or heavily restricted.</li> <li>• Too complex to be described in an exact and comprehensive manner.</li> <li>• Usually described by restrictions on a given natural language.”</li> </ul>
	Languages with lengthy descriptions (S3)	<ul style="list-style-type: none"> <li>• “Can be defined in an exact and comprehensive manner.</li> <li>• Requires more than ten pages.”</li> </ul>
	Languages with short descriptions (S4)	<ul style="list-style-type: none"> <li>• “Exact and comprehensive description requires more than one page but not more than ten pages.”</li> </ul>
	Languages with very short descriptions (S5)	<ul style="list-style-type: none"> <li>• “Can be described in an exact and comprehensive manner on a single page.”</li> </ul>

## Appendix 3: Formal Grammar

In the table below, all grammar rules of the underlying formal grammar of the CNL are listed.

Derivation business rule	
Derivation business rule – 1..1 – Conclusion Part	A <b>Derivation business rule</b> consists of exactly one <b>Conclusion Part</b>
Derivation business rule – 0..* – Condition Part	A <b>Derivation business rule</b> consists of zero or more <b>Condition Part(s)</b>
Conclusion Part	
Conclusion Part – 1..1 – Derivation business rule	A <b>Conclusion Part</b> belongs to exactly one <b>Derivation business rule</b>
Conclusion Part – 0..1 – Modal Claim Type	A <b>Conclusion Part</b> consists of zero or one <b>Modal Claim Type</b>
Conclusion Part – 1..1 – Expression	A <b>Conclusion Part</b> consists of exactly one <b>Expression</b>
Conclusion Part – 1..* – Subject	A <b>Conclusion Part</b> consists of one or more <b>Subject(s)</b>
Condition Part	
Condition Part – 1..1 – Derivation business rule	A <b>Condition Part</b> belongs to exactly one <b>Derivation business rule</b>
Condition Part – 1..* – Construct	A <b>Condition Part</b> consists of one or more <b>Construct(s)</b>
Condition Part – 0..* – Connective	A <b>Condition Part</b> consists of zero or more <b>Connective(s)</b>
Condition Part – 1..* – Expression	A <b>Condition Part</b> consists of one or more <b>Expression(s)</b>
Condition Part – 1..* – Subject	A <b>Condition Part</b> consists of one or more <b>Subject(s)</b>
Subject	
Subject – 1..1 – Conclusion Part	A <b>Subject</b> belongs to exactly one <b>Conclusion Part</b>
Subject – 1..1 – Condition Part	A <b>Subject</b> belongs to exactly one <b>Condition Part</b>
Subject – 1..1 – Quantifier	A <b>Subject</b> is associated with exactly one <b>Quantifier</b>
Subject – 0..1 – Relation	A <b>Subject</b> is associated with zero or one <b>Relation</b>
Subject – 1..1 – Classification	A <b>Subject</b> belongs to exactly one <b>Classification</b>
Subject – 1..1 – Ground	A <b>Subject</b> belongs to exactly one <b>Ground</b>



Quantifier	
Quantifier – 1..1 – Subject	A <b>Quantifier</b> is associated with exactly one <b>Subject</b>
Relation	
Relation – 2..2 – Subject	A <b>Relation</b> is associated with exactly two <b>Subjects</b>
Expression	
Expression – 1..1 – Conclusion Part	A <b>Expression</b> belongs to exactly one <b>Conclusion Part</b>
Expression – 1..1 – Condition Part	A <b>Expression</b> belongs to exactly one <b>Condition Part</b>
An <b>Expression</b> is either a <b>Ground</b> or a <b>Classification</b>	
Classification	
Classification – 1..1 – Propositional Operator	A <b>Classification</b> consists of exactly one <b>Propositional Operator</b>
Classification – 0..* – Value	A <b>Classification</b> consists of zero or more <b>Value(s)</b>
Classification – 0..* – Subject	A <b>Classification</b> consists of zero or more <b>Subject(s)</b>
A <b>Classification</b> consists of at least one <b>Value</b> or of at least one <b>Subject</b>	
Ground	
Ground – 1..* – Mathematical Operator	A <b>Ground</b> consists of one or more <b>Mathematical Operator(s)</b>
Ground – 0..* – Mathematical Function	A <b>Ground</b> consists of zero or more <b>Mathematical Function(s)</b>
Ground – 0..* – Value	A <b>Ground</b> consists of zero or more <b>Value(s)</b>
Ground – 0..* – Subject	A <b>Ground</b> consists of zero or more <b>Subject(s)</b>
A <b>Ground</b> consists of at least one <b>Subject</b> or of at least one <b>Value</b>	
Value	
Value – 1..1 – Classification	A <b>Value</b> belongs to exactly one <b>Classification</b>
Value – 1..1 – Ground	A <b>Value</b> belongs to exactly one <b>Ground</b>
Propositional Operator	
Propositional Operator – 1..1 – Classification	A <b>Propositional Operator</b> belongs to exactly one <b>Classification</b>
Mathematical Operator	
Mathematical Operator – 1..1 – Ground	A <b>Mathematical Operator</b> belongs to exactly one <b>Ground</b>



Mathematical Function	
Mathematical Function – 1..1 – Ground	A <b><u>Mathematical Function</u></b> belongs to exactly one <b><u>Ground</u></b>
Modal Claim Type	
Modal Claim Type – 1..1 – Conclusion Part	A <b><u>Modal Claim Type</u></b> belongs to exactly one <b><u>Conclusion Part</u></b>
Construct	
Construct – 1..1 – Condition Part	A <b><u>Construct</u></b> belongs to exactly one <b><u>Condition Part</u></b>
Connective	
Connective – 1..1 – Condition Part	A <b><u>Connective</u></b> belongs to exactly one <b><u>Condition Part</u></b>
A <b><u>Connective</u></b> must be included to connect two or more <b><u>Conditions</u></b>	

## Appendix 4: Pattern Catalogue

In this appendix, first an explanation is provided in Table 1 for each of the used symbols and format of the patterns. Subsequently, Table 2 lists instantiations for three place holders which are applicable for every pattern. One of these place holders, the mathematical operator, has additional instantiations that are only applicable for specific *ground rule patterns*. These instantiations are listed in a separate table above the ground rule patterns which indicates for which of the patterns it can be used.

On the next page, all 19 patterns of the pattern catalogue are defined and for each pattern an example is given. The patterns are divided in five main categories based on the subdivision as shown in Figure 5.1: 1) Conclusion part – classification rule patterns, 2) Conclusion part – ground rule patterns, 3) Condition part – classification rule patterns, 4) Condition part – ground rule patterns, and 5) Condition part – merge condition rule patterns.

Symbol / Format	Explanation
< >	Angle brackets enclose placeholders. Most of these placeholders are fundamental constructs, which are defined in Chapter 3. The angle brackets refer to components that are fixed when defining a specific type and part of a business rule. They correspond to placeholders since the instantiation can vary. Sometimes, only a fixed set of instantiations is possible. In that case, the list of options is provided along with the pattern set. Otherwise, the business rule author is free to choose its own instantiation.
< .. >	Angle brackets including two dots indicate a placeholder that the business rule author can replace at its discretion.
< individual pattern >	Angle brackets including 'individual pattern' indicate where in the condition part a business rule author can incorporate one of the ten individual rule patterns.
[ ]	Square brackets enclose optional pattern parts. Besides the brackets, the optional pattern parts are made italic.
( )	Parentheses enclose an instantiation of a fundamental construct. In other words, an instantiation for a < placeholder >.
n *	n* indicates that the adjacent pattern part can repeatedly be included in the business rule.
	Vertical bars separate alternative pattern parts from which the business rule author can select one option.

Table 1: Explanation used symbols/format

Instantiations for place holders		
<Quantifier>	<Relation>	<Mathematical Operator>
A /an	of	plus + (addition)
The		minus - (substraction)
Each		divide / (division)
		times * (multiplication)

Table 2: instantiations for three place holders

## 1.1 CONCLUSION PART – *classification rule patterns*

NR	Patternname	Pattern
1.	<b>Equate with VALUE</b>	$\langle \text{Quantifier} \rangle \langle \text{Subject} \rangle [ n^* \langle \text{Relation} \rangle \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ]$ $[ \langle \text{Modal Claim Type} \rangle ] ( \text{is equated to} \mid \text{be equated to} ) \langle \text{Value} \rangle$
	<p><b>Example:</b> The status of <i>the client</i> must ( be equated to ) gold member</p> <p><b>Example 2:</b> The maximum amount of <i>leave days</i> must ( be equated to ) 26</p>	
2.	<b>Equate with SUBJECT</b>	$\langle \text{Quantifier} \rangle \langle \text{Subject} \rangle [ n^* \langle \text{Relation} \rangle \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ]$ $[ \langle \text{Modal Claim Type} \rangle ] ( \text{is equated to} \mid \text{be equated to} ) \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle$ $[ n^* \langle \text{Relation} \rangle \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ]$
	<p><b>Example:</b> The VAT rate must ( be equated to ) the VAT rate of <i>the current year</i></p>	

## 1.2 CONCLUSION PART – *ground rule patterns*

NR	Patternname	Pattern
3.	<b>Equate with Basic Ground</b>	$\langle \text{Quantifier} \rangle \langle \text{Subject} \rangle [ n^* \langle \text{Relation} \rangle \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ]$ $[ \langle \text{Modal Claim Type} \rangle ] ( \text{is computed as} \mid \text{be computed as} )$ $\langle \text{Quantifier} \rangle \langle \text{Subject} \rangle [ n^* \langle \text{Relation} \rangle \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ] \mid$ $\langle \text{Value} \rangle \mid \langle \text{Mathematical Function} \rangle n^* \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle [ n^* \langle \text{Relation} \rangle$ $\langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ]$ $[ n^* \langle \text{Mathematical Operator} \rangle$ $\langle \text{Quantifier} \rangle \langle \text{Subject} \rangle [ n^* \langle \text{Relation} \rangle \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ]$ $\mid \langle \text{Value} \rangle \mid \langle \text{Mathematical Function} \rangle n^* \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle$ $[ n^* \langle \text{Relation} \rangle \langle \text{Quantifier} \rangle \langle \text{Subject} \rangle ] ]$
	<p><b>Example:</b> The total amount of <i>profit of a declarant</i> must ( be computed as ) the total amount of <i>income of the declarant</i> minus the total amount of <i>costs of the declarant</i></p> <p><b>Example 2:</b> The amount of <i>income</i> ( is computer as ) the SUM of the total amount of <i>sold units of each order</i> multiplied by the unit price</p>	

## 2.1 CONDITION PART – classification rule patterns

NR	Patternname	Pattern
4.	<b>Consistency check - one VALUE</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] ( is [ not ] equal to ) <Value>
	<p><b>Example:</b> If the country of the applicant ( is equal to ) NL</p> <p><b>Example 2:</b> If the risk factor ( is not equal to ) 10</p>	
5.	<b>Consistency check - multiple VALUES</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] ( is [ not ] equal to <....> of the following values ) : - <Value > - <Value > - [ n * <Value > ]
	<p><b>Example:</b> If the nationality of the applicant ( is equal to one of the following values ) : - 'CK' - 'GT' - 'ID' - 'MM' - 'NR' - 'NG' - 'PH'</p> <p><b>Example 2:</b> If the amount of ordered items ( is equal to one of the following values ) : - 10 - 50</p>	
6.	<b>Consistency check - one SUBJECT</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] ( is [ not ] equal to ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ]
	<p><b>Example:</b> If the age of the applicant ( is equal to ) the minimum age of application</p>	
7.	<b>Consistency check - multiple SUBJECTs</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] ( is [ not ] equal to <....> of the following values ) : - <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] - <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] - [ n * <Quantifier> <Subject> [ [ n* <Relation> <Quantifier> <Subject> ] ]
	<p><b>Example:</b> If the date ( is not equal to one of the following values): - The end date of the registration - The current date</p>	



## 2.2 CONDITION PART – ground rule patterns

Instantiations for place holder <Mathematical Operator>	
is more than	(applicable for all ground rule patterns 8 till 10)
is less than	(applicable for all ground rule patterns 8 till 10)
is more than or equal to	(applicable for all ground rule patterns 8 till 10)
Is less than or equal to	(applicable for all ground rule patterns 8 till 10)
is earlier than	(only applicable for pattern 9)
is earlier than or equal to	(only applicable for pattern 9)
is later than or equal to	(only applicable for pattern 9)
is later than	(only applicable for pattern 9)
is [ not ]	(only applicable for pattern 10)
is [ not ] equal to	(only applicable for pattern 10)

NR	Patternname	Pattern
8.	<b>Comparison with VALUE</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] <Mathematical Operator> < Value >
	<b>Example:</b> If the annually income of an applicant is less than 34500	
9.	<b>Comparison with SUBJECT</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] <Mathematical Operator> <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ]
	<b>Example:</b> If the amount of rental days is more than the maximum amount of rental days	
10.	<b>Comparison with Basic Ground</b>	( If ) <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] <Mathematical Operator> <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ]   <Value>   <Mathematical Function> n* <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] [ n* <Mathematical Operator> <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ]   <Value>   <Mathematical Function> n* <Quantifier> <Subject> [ n* <Relation> <Quantifier> <Subject> ] ]
	<b>Example:</b> If the registration year of a customer is more than the current year minus 5  <b>Example 2:</b> If the total amount of subscriptions is less than or equal to the SUM of each subscription plus 1	

## 2.3 CONDITION PART – merge condition rule patterns

NR	Patternname	Pattern
11.	<b>Conjunction</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets all of the following conditions) :</p> <ul style="list-style-type: none"> <li>• &lt; Individual pattern &gt;</li> <li>• &lt; Individual pattern &gt;</li> <li>• [ &lt; Individual pattern &gt; ]</li> </ul>
	<p><b>Example:</b>  If the applicant (meets all of the following conditions) :</p> <ul style="list-style-type: none"> <li>• If the nationality of the applicant ( is equal to ) Dutch</li> <li>• If the employment_history_in_the_Netherlands of the applicant is less than 3</li> <li>• If the residence_in_the_Netherlands of the applicant is less than 3</li> </ul>	
12.	<b>Disjunction</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets at least &lt; .. &gt; of the following conditions) :</p> <ul style="list-style-type: none"> <li>• &lt; Individual pattern &gt;</li> <li>• &lt; Individual pattern &gt;</li> <li>• [ &lt; Individual pattern &gt; ]</li> </ul>
	<p><b>Example:</b>  If the person (meets at least one of the following conditions) :</p> <ul style="list-style-type: none"> <li>• If the driving license of the person ( is equal to ) TRUE</li> <li>• If the ID number of the person ( is equal to ) filled</li> </ul>	
13.	<b>Disjunction within Conjunction</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets all of the following conditions):</p> <ul style="list-style-type: none"> <li>• ( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets at least &lt; .. &gt; of the following conditions): <ol style="list-style-type: none"> <li>1. &lt; Individual pattern &gt;</li> <li>2. &lt; Individual pattern &gt;</li> <li>3. [ &lt; Individual pattern &gt; ]</li> </ol> </li> <li>• &lt; Individual pattern &gt;</li> <li>• [ &lt; Individual pattern &gt; ]</li> </ul>
	<p><b>Example:</b>  If the person (meets all of the following conditions) :</p> <ul style="list-style-type: none"> <li>• If the person (meets at least one of the following conditions) : <ol style="list-style-type: none"> <li>1. If the age at 31-01-fiscalyear of the person is more than the AOW age</li> <li>2. If the age at 31-12-fiscalyear of the person ( is equal to ) the AOW age</li> </ol> </li> <li>• If the marital status of the person ( is equal to ) married</li> </ul>	
14.	<b>Conjunction within Disjunction</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets at least &lt; .. &gt; of the following conditions):</p> <ul style="list-style-type: none"> <li>• (If) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets all of the following conditions): <ol style="list-style-type: none"> <li>1. &lt; Individual pattern &gt;</li> <li>2. &lt; Individual pattern &gt;</li> </ol> </li> </ul>



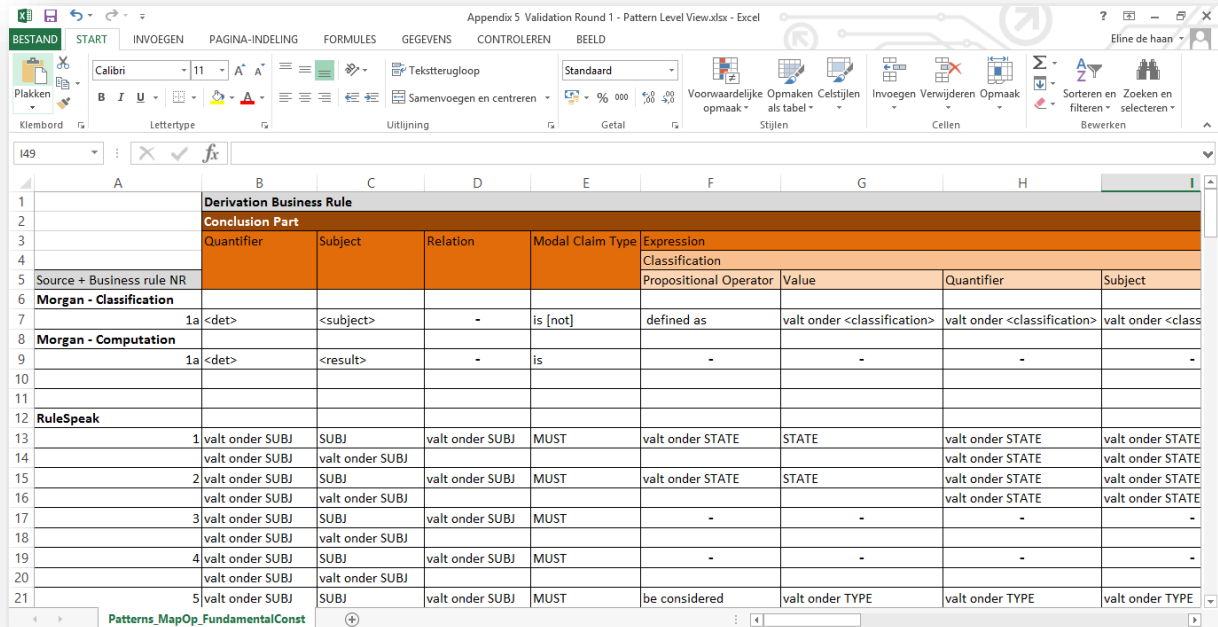
		<p>3. [<i>Individual pattern</i>]</p> <ul style="list-style-type: none"> <li>• <i>Individual pattern</i></li> <li>• [<i>Individual pattern</i>]</li> </ul>
	<p><b>Example:</b>                  If the prospect (meets at least one of the following conditions):</p> <ul style="list-style-type: none"> <li>• If the prospect (meets all of the following conditions):                         <ol style="list-style-type: none"> <li>1. If the age of the prospect is more than 11</li> <li>2. If the age of the prospect is less than 26</li> <li>3. If the area code of the residence of the prospect (is equal to) 020</li> </ol> </li> <li>• If the profession of the prospect is equal to business analyst</li> </ul>	
15.	<b>Exact Amount</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]                  (meets exactly &lt;..&gt; of the following conditions):</p> <ul style="list-style-type: none"> <li>• &lt;Individual pattern&gt;</li> <li>• &lt;Individual pattern&gt;</li> <li>• [<i>Individual pattern</i>]</li> </ul>
	<p><b>Example:</b>                  If the job applicant (meets exactly three of the following conditions):</p> <ul style="list-style-type: none"> <li>• If the work experience of the job applicant is more than 5</li> <li>• If the driving license of the job applicant (is equal to) TRUE</li> <li>• If the field of expertise of the job applicant (is equal to) ICT</li> <li>• If the field of expertise of the job applicant (is equal to) CRM</li> <li>• If the field of expertise of the job applicant (is equal to) BRM</li> </ul>	
16.	<b>Exact amount within Conjunction</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]                  (meets all of the following conditions):</p> <ul style="list-style-type: none"> <li>• ( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]                  (meets exactly &lt;..&gt; of the following conditions):                         <ol style="list-style-type: none"> <li>1. &lt;Individual pattern&gt;</li> <li>2. &lt;Individual pattern&gt;</li> <li>3. [<i>Individual pattern</i>]</li> </ol> </li> <li>• &lt;Individual pattern&gt;</li> <li>• [<i>Individual pattern</i>]</li> </ul>
	<p><b>Example:</b>                  If the product (meets all of the following conditions):</p> <ul style="list-style-type: none"> <li>• If the product (meets exactly one of the following conditions):                         <ol style="list-style-type: none"> <li>1. If the price of the product is more than 50</li> <li>2. If the category of the product (is equal to) hardware</li> </ol> </li> <li>• If the return date of the product is earlier than the expiration date of the warranty period</li> </ul>	



17.	<b>Exact amount within Disjunction</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets at least &lt;..&gt; of the following conditions) :</p> <ul style="list-style-type: none"> <li>• ( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets exactly &lt;..&gt; of the following conditions): <ol style="list-style-type: none"> <li>1. &lt; Individual pattern &gt;</li> <li>2. &lt; Individual pattern &gt;</li> <li>3. [ &lt; Individual pattern &gt; ]</li> </ol> </li> <li>• &lt; Individual pattern &gt;</li> <li>• [ &lt; Individual pattern &gt; ]</li> </ul>
<p><b>Example:</b>  If the client ( meets at least one of the following conditions ) :</p> <ul style="list-style-type: none"> <li>• If the client (meets exactly one of the following conditions): <ol style="list-style-type: none"> <li>1. If the latest invoice date of the client is later than or equal to 2014</li> <li>2. If the membership type of the client ( is equal to ) private</li> </ol> </li> <li>• If the solvency of the client ( is equal to ) high</li> </ul>		
18.	<b>Conjunction within Exact amount</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets exactly &lt;..&gt; of the following conditions):</p> <ul style="list-style-type: none"> <li>• ( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets all of the following conditions): <ol style="list-style-type: none"> <li>1. &lt; Individual pattern &gt;</li> <li>2. &lt; Individual pattern &gt;</li> <li>3. [ &lt; Individual pattern &gt; ]</li> </ol> </li> <li>• &lt; Individual pattern &gt;</li> <li>• [ &lt; Individual pattern &gt; ]</li> </ul>
<p><b>Example:</b>  If the policyholder ( meets exactly one of the following conditions ) :</p> <ul style="list-style-type: none"> <li>• If the policyholder ( meets all of the following condition ): <ol style="list-style-type: none"> <li>1. If the amount of due invoices of the policyholder ( is equal to ) 0</li> <li>2. If the duration of the insurance of the policyholder is more than or equal to 15</li> </ol> </li> <li>• If the insurance type of the policyholder ( is equal to ) platinum</li> </ul>		
19.	<b>Disjunction within exact amount</b>	<p>( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets exactly &lt;..&gt; of the following conditions)</p> <ul style="list-style-type: none"> <li>• ( If ) &lt;Quantifier&gt; &lt;Subject&gt; [ n* &lt;Relation&gt; &lt;Quantifier&gt; &lt;Subject&gt; ]  (meets at least &lt;..&gt; of the following conditions): <ol style="list-style-type: none"> <li>1. &lt; Individual pattern &gt;</li> <li>2. &lt; Individual pattern &gt;</li> <li>3. [ &lt; Individual pattern &gt; ]</li> </ol> </li> <li>• &lt; Individual pattern &gt;</li> <li>• [ &lt; Individual pattern &gt; ]</li> </ul>
<p><b>Example:</b>  If the customer ( meets exactly one of the following conditions ) :</p> <ul style="list-style-type: none"> <li>• If the customer (meets at least one of the following conditions): <ol style="list-style-type: none"> <li>1. If the marital status of the customer ( is equal to ) single</li> <li>2. If the gender of the customer ( is equal to ) female</li> </ol> </li> <li>• If the amount of children of the customer is more than 0</li> </ul>		

## Appendix 5: Validation Round 1 – Pattern Level View

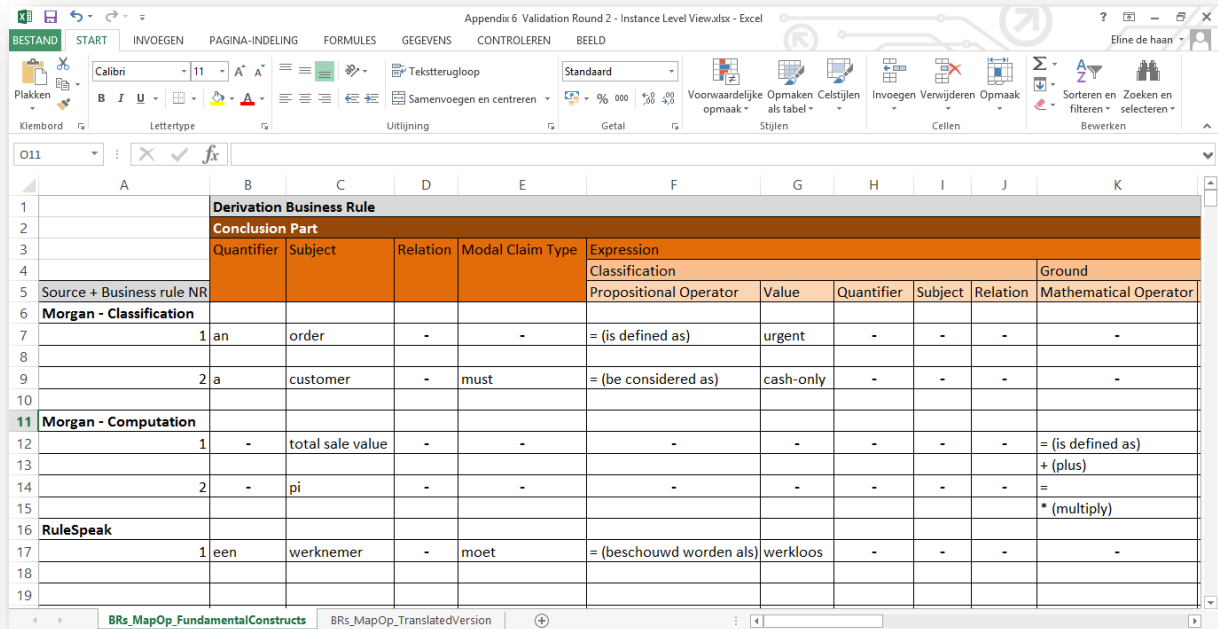
Due to the large size of the validation file, only a snapshot is included in this appendix. The entire appendix can be found on the enclosed USB-stick.



Derivation Business Rule										
Conclusion Part										
	Quantifier	Subject	Relation	Modal Claim Type	Expression	Classification	Propositional Operator	Value	Quantifier	Subject
Source + Business rule NR										
<b>Morgan - Classification</b>										
7	1a	<det>	<subject>	-	is [not]	defined as	valt onder <classification>	valt onder <classification>	valt onder <class	
<b>Morgan - Computation</b>										
9	1a	<det>	<result>	-	is	-	-	-	-	-
<b>RuleSpeak</b>										
13	1	valt onder SUBJ	SUBJ	valt onder SUBJ	MUST	valt onder STATE	STATE	valt onder STATE	valt onder STATE	valt onder STATE
14		valt onder SUBJ	valt onder SUBJ					valt onder STATE	valt onder STATE	valt onder STATE
15	2	valt onder SUBJ	SUBJ	valt onder SUBJ	MUST	valt onder STATE	STATE	valt onder STATE	valt onder STATE	valt onder STATE
16		valt onder SUBJ	valt onder SUBJ					valt onder STATE	valt onder STATE	valt onder STATE
17	3	valt onder SUBJ	SUBJ	valt onder SUBJ	MUST	-	-	-	-	-
18		valt onder SUBJ	valt onder SUBJ							
19	4	valt onder SUBJ	SUBJ	valt onder SUBJ	MUST	-	-	-	-	-
20		valt onder SUBJ	valt onder SUBJ							
21	5	valt onder SUBJ	SUBJ	valt onder SUBJ	MUST	be considered	valt onder TYPE	valt onder TYPE	valt onder TYPE	valt onder TYPE

## Appendix 6: Validation Round 2 – Instance Level View

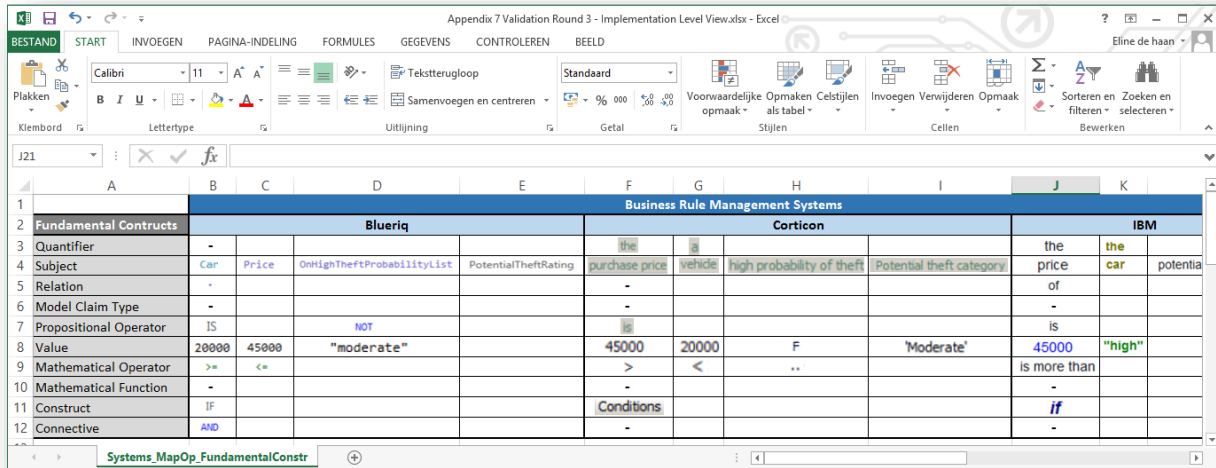
Due to the large size of the validation file, only a snapshot is included in this appendix. The entire appendix can be found on the enclosed USB-stick.



Derivation Business Rule										
Conclusion Part										
	Quantifier	Subject	Relation	Modal Claim Type	Expression	Classification				Ground
Source + Business rule NR	Propositional Operator	Value	Quantifier	Subject	Relation	Mathematical Operator				
<b>Morgan - Classification</b>										
1	an	order	-	-	= (is defined as)	urgent	-	-	-	-
2	a	customer	-	must	= (be considered as)	cash-only	-	-	-	-
<b>Morgan - Computation</b>										
1	-	total sale value	-	-	-	-	-	-	-	= (is defined as)
2	-	pi	-	-	-	-	-	-	-	+ (plus)
										=
										* (multiply)
<b>RuleSpeak</b>										
1	een	werknemer	-	moet	= (beschouwd worden als)	werkloos	-	-	-	-

## Appendix 7: Validation Round 3 – Implementation Dependent View

Due to the large size of the validation file, only a snapshot is included in this appendix. The entire appendix can be found on the enclosed USB-stick.



Business Rule Management Systems											
Fundamental Contracts	Blueriq				Corticon				IBM		
Quantifier	-				the	is			the	the	
Subject	Car	Price	OnHighTheftProbabilityList	PotentialTheftRating	purchase price	vehicle	high probability of theft	Potential theft category	price	car	potentia
Relation	-				-				-		
Model Claim Type	-				-				-		
Propositional Operator	IS		NOT		is				is		
Value	20000	45000	"moderate"		45000	20000	F	'Moderate'	45000	"high"	
Mathematical Operator	>=	<=			>	<	**		is more than		
Mathematical Function	-				-				-		
Construct	IF				Conditions				if		
Connective	AND				-				-		

## Appendix 8: Decomposed Case Study Business Rule Set

This appendix includes the entire set of decomposed business rules used for the validation of the pattern catalogue. These business rules are provided by the case study company and originate from the act called the “Zorgverzekeringswet” abbreviated as (ZVW). The tables below provide the original business rules of the case study business rule set, and the decomposed version of the nested business rules. The decomposition resulted into 45 atomic business rules.

### 6008 Bepalen ZVW-plicht en IB dagen 2013

<b>1. <u>Bepalen hulpdatums begin- en einddatum premieplicht (H1 en H2):</u></b>	
<b>Als</b>	[indicatie geheel jaar geen Zvw plicht] (a) = Ja <span style="float: right;">(Geen AWBZ)</span>
<b>Dan</b>	[begindatum premieplicht Zvw] (H1) = [leeg] en [einddatum premieplicht Zvw] (H2) = [leeg]
<b>Anders</b>	<span style="float: right;">(wel Zvw-plicht)</span>
<b>Als</b>	[begindatum afwijkende periodeplicht Zvw] (c) is niet gelijk aan [leeg] (bepalen begindatums)
<b>Dan</b>	[begindatum premieplicht Zvw] (H1) = [begindatum afwijkende periodeplicht Zvw] (c)
<b>Anders</b>	[begindatum premieplicht Zvw] (H1) = 1-1-belastingjaar (b)
<b>Einde als</b>	
<b>Als</b>	[einddatum afwijkende periodeplicht Zvw] (d) is niet gelijk aan [leeg] (bepalen einddatums)
<b>Dan</b>	[einddatum premieplicht Zvw] (H2) = [einddatum afwijkende periodeplicht Zvw] (d)
<b>Anders</b>	[einddatum premieplicht Zvw] (H2) = 31-12-belastingjaar (b)
<b>Einde Als</b>	
<b>Einde Als</b>	
<b>1. <u>Transformed for specification</u></b>	
<b>1a</b>	
<b>Als</b>	[indicatie geheel jaar geen Zvw plicht] (a) = Ja
<b>Dan</b>	[begindatum premieplicht Zvw] (H1) = [leeg]
<b>1b</b>	
<b>Als</b>	[indicatie geheel jaar geen Zvw plicht] (a) = Ja
<b>Dan</b>	[einddatum premieplicht Zvw] (H2) = [leeg]
<b>1c</b>	
<b>Als</b>	[begindatum afwijkende periodeplicht Zvw] (c) is niet gelijk aan [leeg]
<b>Dan</b>	[begindatum premieplicht Zvw] (H1) = [begindatum afwijkende periodeplicht Zvw] (c)
<b>1d</b>	
<b>Als</b>	[begindatum afwijkende periodeplicht Zvw] (c) is gelijk aan [leeg]
<b>Dan</b>	[begindatum premieplicht Zvw] (H1) = 1-1-belastingjaar (b)
<b>1e</b>	
<b>Als</b>	[einddatum afwijkende periodeplicht Zvw] (d) is niet gelijk aan [leeg]
<b>Dan</b>	[einddatum premieplicht Zvw] (H2) = [einddatum afwijkende periodeplicht Zvw] (d)
<b>1f</b>	
<b>Als</b>	[einddatum afwijkende periodeplicht Zvw] (d) is gelijk aan [leeg]
<b>Dan</b>	[einddatum premieplicht Zvw] (H2) = 31-12-belastingjaar (b)

## 2. Aantal premiedagen ZVW (U1):

**Als** [begindatum premieplicht Zvw] (H1) = [leeg]

**Dan** [aantal dagen ZVW-plicht] (U1) = 0

### Anders

**Als** [datum ingang actief militair] (e)

en

[datum beëindiging actief-militair] (f) aanwezig (gevuld)

**Dan** [aantal dagen ZVW-plicht] (U1) =

(maand uit [einddatum premieplicht Zvw] (H2) min

maand uit [begindatum premieplicht Zvw] (H1)) maal

30 plus

(dag uit [einddatum premieplicht Zvw] (H2) min

dag uit [begindatum premieplicht Zvw] (H1)) min

(maand uit [datum beëindiging actief-militair] (f) min

maand uit [datum ingang actief militair] (e)) maal

30 min

(dag uit [datum beëindiging actief-militair] (f) min

dag uit [datum ingang actief militair] (e))

**Anders** [aantal dagen ZVW-plicht] (U1) =

(maand uit [einddatum premieplicht Zvw] (H2) min

maand uit [begindatum premieplicht Zvw] (H1)) maal

30 plus

(dag uit [einddatum premieplicht Zvw] (H2) min

dag uit [begindatum premieplicht Zvw] (H1))

### Einde als

**Als** [aantal dagen ZVW-plicht] (U1) is kleiner dan 0

**Dan** [aantal dagen ZVW-plicht] (U1) = 0

### Einde als

**Einde als**

## 2. Transformed for specification

### 2a

**Als** [begindatum premieplicht Zvw] (H1) = [leeg]

**Dan** [aantal dagen ZVW-plicht] (U1) = 0

### 2b

**Als** datum ingang actief militair] (e) is aanwezig (gevuld)

en

[datum beëindiging actief-militair] (f) is aanwezig (gevuld)

**Dan** [aantal dagen ZVW-plicht] (U1) =

(maand uit [einddatum premieplicht Zvw] (H2) min

maand uit [begindatum premieplicht Zvw] (H1)) maal

30 plus

(dag uit [einddatum premieplicht Zvw] (H2) min

dag uit [begindatum premieplicht Zvw] (H1)) min

(maand uit [datum beëindiging actief-militair] (f) min

maand uit [datum ingang actief militair] (e)) maal

30 min

(dag uit [datum beëindiging actief-militair] (f) min

dag uit [datum ingang actief militair] (e))

**2c**

**Als** [datum ingang niet actief militair] (e) is afwezig (leeg)  
 en  
 [datum beëindiging actief-militair] (f) is afwezig (leeg)  
**Dan** [aantal dagen ZVW-plicht] (U1) =  
 (maand uit [einddatum premieplicht Zvw] (H2) min  
 maand uit [begindatum premieplicht Zvw] (H1)) maal  
 30 plus  
 (dag uit [einddatum premieplicht Zvw] (H2) min  
 dag uit [begindatum premieplicht Zvw] (H1))

**2d**

**Als** [aantal dagen ZVW-plicht] (U1) is kleiner dan 0  
**Dan** [aantal dagen ZVW-plicht] (U1) = 0

**3. Aantal IB-dagen (U2):**

[aantal IB-dagen] (U2) =  
 (maand uit [datum beëindigen (soort) belastingplicht] (h) min  
 maand uit [datum ingang (soort) belastingplicht] (g)) maal 30 plus  
 (dag uit [datum beëindigen (soort) belastingplicht] (h) min  
 dag uit [datum ingang (soort) belastingplicht] (g))

**3. Not transformed****7500 Bepalen bijdrage-inkomen ZVW aanslag 2013****4. Bijdrage-inkomen ZVW zonder alimentatie (H1):**

[bijdrage-inkomen ZVW zonder alimentatie] (H1) :=  
 [belastbare winst uit onderneming] (a) min  
 [in winst begrepen loon] (g) min  
 [winst deelvisser] (j) plus  
 [totaal buitenlandse inkomsten uit dienstbetrekking] (b) plus  
 [netto resultaat uit overige werkzaamheden] (c) plus  
 [saldo periodieke uitkeringen na aftrekbare kosten] (d)

**4. Not transformed for specification**

**5. Bijdrage-inkomen ZVW aanslag (U1):**

ALS [vorig jaar alimentatie overgangstarief] (f) = 'ja'  
 DAN [bijdrage-inkomen ZVW aanslag] (U1) :=  
     [bijdrage-inkomen ZVW zonder alimentatie] (H1)

ANDERS  
     [bijdrage-inkomen ZVW aanslag] (U1) :=  
         [bijdrage-inkomen ZVW zonder alimentatie] (H1) plus  
         [saldo alimentatie na aftrekbare kosten] (e)

EINDE-ALS

ALS [bijdrage-inkomen ZVW aanslag] (U1) is kleiner dan 0  
 DAN [bijdrage-inkomen ZVW aanslag] (U1) := 0  
 EINDE-ALS

**5. Transformed for specification****5a**

ALS [vorig jaar alimentatie overgangstarief] (f) = 'ja'  
 DAN [bijdrage-inkomen ZVW aanslag] (U1) := [bijdrage-inkomen ZVW zonder alimentatie] (H1)

**5b**

ALS [vorig jaar alimentatie overgangstarief] (f) = 'nee'  
 DAN [bijdrage-inkomen ZVW aanslag] (U1) := [bijdrage-inkomen ZVW zonder alimentatie] (H1) plus  
     [saldo alimentatie na aftrekbare kosten] (e)

**5c**

ALS [bijdrage-inkomen ZVW aanslag] (U1) is kleiner dan 0  
 DAN [bijdrage-inkomen ZVW aanslag] (U1) := 0

**6. Niet relevante VA ZVW**

ALS indicatie ZVW aanslag verwijderen is gevuld  
     ALS indicatie ZVW aanslag verwijderen = 'J'  
     DAN bijdrage-inkomen ZVW aanslag (U1) := 0  
     EINDE-ALS

EINDE-ALS

**6. Transformed for specification**

ALS indicatie ZVW aanslag verwijderen is gevuld  
     en

ALS indicatie ZVW aanslag verwijderen = 'J'  
 DAN bijdrage-inkomen ZVW aanslag (U1) := 0



## 7520 Bepalen bijdrage ZVW aanslag 2013

### 7. Bijdrage ZVW aanslag (U1):

[bijdrage ZVW aanslag] (U1) :=  
 [toegepast bijdrage-inkomen ZVW] (a) maal  
 [percentage verlaagd tarief] (b)

### 7. Not transformed for specification

## 7565 Toetsing art. 9.4 ZVW 2013

### 8. Bepalen toepassen artikel 9.4 (U2):

ALS **[artikel 9.4] (b) = [leeg]:** *(aanslag vaststellen)*  
 ALS [bijdrage ZVW aanslag] (a) is groter dan *(berekende ZVW boven aanslaggrens)*  
 [aanslag-grens artikel 9.4] (d)  
 DAN [toepassing artikel 9.4] (U2) := 21 *(aanslag opleggen)*  
 Anders: [toepassing artikel 9.4] (U2) := 35 *(NIHIL-aanslag)*  
 Einde-als

Anders:  
 ALS **[artikel 9.4] (b) = "0"** *(handmatig toegekend, alleen voor uitzonderingen)*  
 dan: [toepassing artikel 9.4] (U2) := 35 *(NIHIL-aanslag)*

Anders:  
 ALS **[artikel 9.4] (b) = "3"** *(termijn van 3 jaar is overschreden)*  
 dan: [toepassing artikel 9.4] (U2) := 0 *(te betalen bedrag: resulteert in signaalpost)*

Einde-Als

### 8. Transformed for specification

#### 8a:

ALS **[artikel 9.4] (b) = [leeg]:**  
 en  
 ALS [bijdrage ZVW aanslag] (a) is groter dan [aanslag-grens artikel 9.4] (d)  
 DAN [toepassing artikel 9.4] (U2) := 21

#### 8b:

ALS **[artikel 9.4] (b) = [leeg]:**  
 en  
 ALS [bijdrage ZVW aanslag] (a) is kleiner of gelijk aan [aanslag-grens artikel 9.4] (d)  
 DAN: [toepassing artikel 9.4] (U2) := 35

#### 8c:

ALS **[artikel 9.4] (b) = "0"**  
 DAN [toepassing artikel 9.4] (U2) := 35

#### 8d:

ALS **[artikel 9.4] (b) = "3"**  
 DAN [toepassing artikel 9.4] (U2) := 0

**9. Vaststellen bedrag aanslag (U1):**

ALS [toepassing artikel 9.4] (U2) = 35  
 DAN [bedrag aanslag] (U1) := 0 (NIHIL-aanslag)  
 [indicatie-nihil-aanslag] (U3) := 'ja'  
 Anders: [bedrag aanslag] (U1) :=  
 [bijdrage ZVW aanslag] (a)  
 (toepassing artikel 9.4 = 0 of 21: aanslag wordt in principe opgelegd)  
 Einde-Als

**9. Transformed for specification****9a**

ALS [toepassing artikel 9.4] (U2) = 35  
 DAN [bedrag aanslag] (U1) := 0

**9b**

ALS [toepassing artikel 9.4] (U2) = 35  
 DAN [indicatie-nihil-aanslag] (U3) := 'ja'

**9c:**

ALS [toepassing artikel 9.4] (U2) ≠ 35  
 DAN [bedrag aanslag] (U1) := [bijdrage ZVW aanslag] (a)

**10. Saldo aanslag voor heffingsrente en boete (U4):**

[saldo aanslag voor heffingsrente en boete] (U4) := [bedrag aanslag] (U1) min [eerdere aanslag(en)] (f)

**10. Not transformed for specification**

## Bepalen toegepast bijdrage-inkomen ZVW aanslag 2013b

### 11. Gecorrigeerd bijdrage-inkomen ZVW (U1):

ALS [correctie-inkomen ZVW aanslag] (e) is gevuld

DAN

[gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'

[gecorrigeerd bijdrage-inkomen ZVW] (U1) := [bijdrage-inkomen ZVW aanslag] (a) min

ABS [correctie-inkomen ZVW aanslag] (e)

ANDERS

[gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'N'

EIND-ALS

ALS [gecorrigeerd bijdrage-inkomen ZVW] (U1) is kleiner dan 0

DAN [gecorrigeerd bijdrage-inkomen ZVW] (U1) is gelijk aan 0

### 11. Transformed for specification

#### 11a.

ALS [correctie-inkomen ZVW aanslag] (e) is gevuld

DAN [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'

#### 11b.

ALS [correctie-inkomen ZVW aanslag] (e) is gevuld

DAN [gecorrigeerd bijdrage-inkomen ZVW] (U1) := [bijdrage-inkomen ZVW aanslag] (a) min

ABS [correctie-inkomen ZVW aanslag] (e)

#### 11c.

ALS [correctie-inkomen ZVW aanslag] (e) is leeg

DAN [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'N'

#### 11d.

ALS [gecorrigeerd bijdrage-inkomen ZVW] (U1) is kleiner dan 0

DAN [gecorrigeerd bijdrage-inkomen ZVW] (U1) is gelijk aan 0

**12. Herleid bijdrage-inkomen ZVW aftrekmethode (U3):**

[Herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'N'

ALS [bijdrage-inkomen ZVW aanslag buitenland] (d) is gevuld  
 DAN [herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'J'

ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'  
 DAN [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) :=  
     [gecorrigeerd bijdrage-inkomen ZVW] (U1) min  
     [bijdrage-inkomen ZVW aanslag buitenland] (d)

ANDERS  
     [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) :=  
     [bijdrage-inkomen ZVW aanslag] (a) min  
     [bijdrage-inkomen ZVW aanslagbuitenland](d)

EINDE-ALS  
 EINDE-ALS

**12. Transformed for specification****12a.**

[Herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'N'

**12b.**

ALS [bijdrage-inkomen ZVW aanslag buitenland] (d) is gevuld  
 DAN [herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'J'

**12c.**

ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'  
 DAN [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) :=  
     [gecorrigeerd bijdrage-inkomen ZVW] (U1) min  
     [bijdrage-inkomen ZVW aanslag buitenland] (d)

**12d.**

ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'N'  
 DAN [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) :=  
     [bijdrage-inkomen ZVW aanslag] (a) min  
     [bijdrage-inkomen ZVW aanslagbuitenland](d)

**13. Toegepast maximum bijdrage-inkomen ZVW (U4):**

[toegepast maximum bijdrage-inkomen ZVW] (U4) :=  
 [maximum bijdrage-inkomen ZVW] (g)

ALS [aantal dagen ZVW-plicht] (c) is kleiner dan  
 [aantal dagen in belastingjaar]  
 EN  
 ( [einde ZVW door overlijden] (f) = 'N'  
 OF

[einde ZVW door overlijden] (f) = <Leeg>  
 DAN [toegepast maximum bijdrage-inkomen ZVW] (U4) :=  
 ( [aantal dagen ZVW-plicht] (c) delen door  
 [aantal dagen in belastingjaar] ) maal  
 [maximum bijdrage-inkomen ZVW] (g)

EINDE-ALS

**13. Transformed for specification****13a.**

[toegepast maximum bijdrage-inkomen ZVW] (U4) := [maximum bijdrage-inkomen ZVW] (g)

**13b.**

ALS [aantal dagen ZVW-plicht] (c) is kleiner dan  
 [aantal dagen in belastingjaar]  
 EN  
 ( [einde ZVW door overlijden] (f) = 'N'  
 OF  
 [einde ZVW door overlijden] (f) = <Leeg>

DAN [toegepast maximum bijdrage-inkomen ZVW] (U4) :=  
 ( [aantal dagen ZVW-plicht] (c) delen door  
 [aantal dagen in belastingjaar] ) maal  
 [maximum bijdrage-inkomen ZVW] (g)

**14. Heffingsruimte bijdrage-inkomen ZVW (U5):**

[heffingsruimte bijdrage-inkomen ZVW] (U5) :=  
 [toegepast maximum bijdrage-inkomen ZVW] (U4) min  
 [bijdrage-inkomen ZVW inhouding] (h)

ALS [heffingsruimte bijdrage-inkomen ZVW] (U5) is kleiner dan 0

DAN [heffingsruimte bijdrage-inkomen ZVW] (U5) := 0

EINDE-ALS

**14. Transformed for specification****14 a:**

[heffingsruimte bijdrage-inkomen ZVW] (U5) :=  
 [toegepast maximum bijdrage-inkomen ZVW] (U4) min  
 [bijdrage-inkomen ZVW inhouding] (h)

**14b:**

ALS [heffingsruimte bijdrage-inkomen ZVW] (U5) is kleiner dan 0

DAN [heffingsruimte bijdrage-inkomen ZVW] (U5) := 0

**15. Toegepast bijdrage-inkomen ZVW (U6):**

[toegepast bijdrage-inkomen ZVW] (U6) :=

[heffingsruimte bijdrage-inkomen ZVW] (U5)

ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'

ALS [gecorrigeerd bijdrage-inkomen ZVW] (U1) is kleiner dan  
[toegepast bijdrage-inkomen ZVW] (U6)

DAN [toegepast bijdrage-inkomen ZVW] (U6) :=  
[gecorrigeerd bijdrage-inkomen ZVW] (U1)

EINDE-ALS

EINDE-ALS

ALS [herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'J'

ALS [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) is kleiner dan  
[toegepast bijdrage-inkomen ZVW] (U6)

DAN [toegepast bijdrage-inkomen ZVW] (U6) :=  
[herleid bijdrage-inkomen ZVW aftrekmethode] (U3)

EINDE-ALS

EINDE-ALS

ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) = 'N' en

[herleid bijdrage-inkomen aftrek aanwezig] (H3) = 'N'

ALS [bijdrage-inkomen ZVW aanslag] (a) is kleiner dan  
[heffingsruimte bijdrage-inkomen ZVW] (U5)

DAN [toegepast bijdrage-inkomen ZVW] (U6) :=  
[bijdrage-inkomen ZVW aanslag] (a)

EINDE-ALS

EINDE-ALS

ALS [toegepast bijdrage-inkomen ZVW] (U6) is groter dan

[maximum bijdrage-inkomen ZVW] (g)

DAN [toegepast bijdrage-inkomen ZVW] (U6) := [maximum bijdrage-inkomen ZVW] (g)

EINDE-ALS

**15. Transformed for specification****15a.**

[toegepast bijdrage-inkomen ZVW] (U6) := [heffingsruimte bijdrage-inkomen ZVW] (U5)

**15b.**

ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'

en

ALS [gecorrigeerd bijdrage-inkomen ZVW] (U1) is kleiner dan

[toegepast bijdrage-inkomen ZVW] (U6)

DAN [toegepast bijdrage-inkomen ZVW] (U6) := [gecorrigeerd bijdrage-inkomen ZVW] (U1)

**15c.**

ALS [herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'J'

en

ALS [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) is kleiner dan

[toegepast bijdrage-inkomen ZVW] (U6)

DAN [toegepast bijdrage-inkomen ZVW] (U6) := [herleid bijdrage-inkomen ZVW aftrekmethode] (U3)

**15d.**

ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) = 'N'

en

[herleid bijdrage-inkomen aftrek aanwezig] (H3) = 'N'



	en	
ALS	[bijdrage-inkomen ZVW aanslag] (a) is kleiner dan [heffingsruimte bijdrage-inkomen ZVW] (U5)	
DAN	[toegepast bijdrage-inkomen ZVW] (U6) := [bijdrage-inkomen ZVW aanslag] (a)	
<b>15e.</b>		
ALS	[toegepast bijdrage-inkomen ZVW] (U6) is groter dan [maximum bijdrage-inkomen ZVW] (g)	
DAN	[toegepast bijdrage-inkomen ZVW] (U6) := [maximum bijdrage-inkomen ZVW] (g)	

<b>16. Niet relevante VA ZVW</b>		
ALS	indicatie ZVW aanslag verwijderen is gevuld	
ALS	indicatie ZVW aanslag verwijderen = J	
DAN	gecorrigeerd bijdrage-inkomen ZVW (U1)	= 0
	herleid bijdrage-inkomen ZVW aftrekmethode (U3)	= 0
	heffingsruimte bijdrage-inkomen ZVW (U4)	= 0
EINDE-ALS		
EINDE-ALS		
<b>16. Transformed for specification</b>		
<b>16a.</b>		
ALS	indicatie ZVW aanslag verwijderen is gevuld	
	en	
ALS	indicatie ZVW aanslag verwijderen = J	
DAN	gecorrigeerd bijdrage-inkomen ZVW (U1)	= 0
<b>16b.</b>		
ALS	indicatie ZVW aanslag verwijderen is gevuld	
	en	
ALS	indicatie ZVW aanslag verwijderen = J	
DAN	herleid bijdrage-inkomen ZVW aftrekmethode (U3)	= 0
<b>16c.</b>		
ALS	indicatie ZVW aanslag verwijderen is gevuld	
	en	
ALS	indicatie ZVW aanslag verwijderen = J	
DAN	heffingsruimte bijdrage-inkomen ZVW (U4)	= 0

## Appendix 9: Validation Patterns – Instance Level View

This appendix includes the validation of the patterns by means of specifying the 45 atomic business rules of the case study data set with the pattern catalogue.

### BR 1: Bepalen hulpdatums begin- en einddatum premieplicht (H1 en H2)

Original business rule NR	Business rule in pattern
<p><b>1a</b></p> <p><b>Als</b> [indicatie geheel jaar geen Zvw plicht] (a) = Ja</p> <p><b>Dan</b> [begindatum premieplicht Zvw] (H1) = [leeg]</p>	<p>&lt; De &gt; &lt; begindatum premieplicht Zvw (H1) &gt; ( wordt gelijk gesteld aan ) &lt; leeg &gt;</p> <p>( indien ) &lt; de &gt; &lt; indicatie geheel jaar geen Zvw plicht (a) &gt; ( gelijk is aan ) &lt; Ja &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>1b</b></p> <p><b>Als</b> [indicatie geheel jaar geen Zvw plicht] (a) = Ja</p> <p><b>Dan</b> [einddatum premieplicht Zvw] (H2) = [leeg]</p>	<p>&lt; De &gt; &lt; einddatum premieplicht Zvw (H2) &gt; ( wordt gelijk gesteld aan ) &lt; leeg &gt;</p> <p>( indien ) &lt; de &gt; &lt; indicatie geheel jaar geen Zvw plicht (a) &gt; ( gelijk is aan ) &lt; Ja &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	





Original business rule NR	Business rule in pattern
<p><b>1f</b></p> <p><b>Als</b> [einddatum afwijkende periodeplicht Zvw] (d) is gelijk aan [leeg]</p> <p><b>Dan</b> [einddatum premieplicht Zvw] (H2) = 31-12-belastingjaar (b)</p>	<p>&lt; De &gt; &lt; einddatum premieplicht Zvw (H2) &gt; ( wordt gelijk gesteld aan ) &lt; 31 - 12 - belastingjaar (b) &gt;</p> <p>( indien ) &lt; de &gt; &lt; einddatum afwijkende periodeplicht Zvw (d) &gt; ( gelijk is aan ) &lt; leeg &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

**BR2: Aantal premiedagen ZVW (U1):**

Original business rule NR	Business rule in pattern
<p><b>2a</b></p> <p><b>Als</b> [begindatum premieplicht Zvw] (H1) = [leeg]</p> <p><b>Dan</b> [aantal dagen ZVW-plicht] (U1) = 0</p>	<p>&lt; Het &gt; &lt; aantal dagen ZVW-plicht (U1) &gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; de &gt; &lt; begindatum premieplicht Zvw (H1) &gt; ( gelijk is aan ) &lt; leeg &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>2b</b></p> <p><b>Als</b> [datum ingang actief militair] (e) is aanwezig (gevuld) en [datum beëindiging actief-militair] (f) is aanwezig (gevuld)</p> <p><b>Dan</b> [aantal dagen ZVW-plicht] (U1) = (maand uit [einddatum premieplicht Zvw] (H2) min maand uit [begindatum premieplicht Zvw] (H1)) maal 30 plus (dag uit [einddatum premieplicht Zvw] (H2) min dag uit [begindatum premieplicht Zvw] (H1)) min (maand uit [datum beëindiging actief-militair] (f) min maand uit [datum ingang actief militair] (e)) maal 30 min (dag uit [datum beëindiging actief-militair] (f) min dag uit [datum ingang actief militair] (e))</p>	<p>&lt; Het &gt; &lt; aantal dagen ZVW-plicht (U1) &gt; ( wordt berekend als ) &lt; maand uit &gt; &lt; de &gt; &lt; einddatum premieplicht Zvw (H2) &gt; &lt; min &gt; &lt; maand uit &gt; &lt; de &gt; &lt; begindatum premieplicht Zvw (H1) &gt; &lt; maal &gt; &lt; 30 &gt; &lt; plus &gt; &lt; dag uit &gt; &lt; de &gt; &lt; einddatum premieplicht Zvw (H2) &gt; &lt; min &gt; &lt; dag uit &gt; &lt; de &gt; &lt; begindatum premieplicht Zvw (H1) &gt; &lt; min &gt; &lt; maand uit &gt; &lt; de &gt; &lt; datum beëindiging actief-militair (f) &gt; &lt; min &gt; &lt; maand uit &gt; &lt; de &gt; &lt; datum ingang actief militair (e) &gt; &lt; maal &gt; &lt; 30 &gt; &lt; min &gt; &lt; dag uit &gt; &lt; de &gt; &lt; datum beëindiging actief-militair (f) &gt; &lt; min &gt; &lt; dag uit &gt; &lt; de &gt; &lt; datum ingang actief militair (e) &gt;</p> <p>( indien ) &lt; het &gt; &lt; aantal dagen ZVW-plicht (U1) &gt; ( aan alle volgende voorwaarden voldoet ) :</p> <ul style="list-style-type: none"> <li>• ( indien ) &lt; de &gt; &lt; datum ingang actief militair (e) &gt; ( gelijk is aan ) &lt; aanwezig (gevuld) &gt;</li> <li>• ( indien ) &lt; de &gt; &lt; datum beëindiging actief-militair (f) &gt; ( gelijk is aan ) &lt; aanwezig (gevuld) &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following three separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground), <b>pattern 11</b> (conjunction) and <b>pattern 4</b> (consistency check one value).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>2c</b></p> <p><b>Als</b> [datum ingang actief militair] (e) is afwezig (leeg) en [datum beëindiging actief-militair] (f) is afwezig (leeg)</p> <p><b>Dan</b> [aantal dagen ZVW-plicht] (U1) = (maand uit [einddatum premieplicht Zvw] (H2) min maand uit [begindatum premieplicht Zvw] (H1)) maal 30 plus (dag uit [einddatum premieplicht Zvw] (H2) min dag uit [begindatum premieplicht Zvw] (H1))</p>	<p>&lt; De &gt; &lt; aantal dagen ZVW-plicht (U1) &gt; ( wordt berekend als )</p> <p>&lt; maand uit &gt; &lt; de &gt; &lt; einddatum premieplicht Zvw (H2) &gt; &lt; min &gt; maand uit &lt; de &gt; &lt; begindatum premieplicht Zvw (H1) &gt; &lt; maal &gt; &lt; 30 &gt; &lt; plus &gt; &lt; dag uit &gt; &lt; de &gt; &lt; einddatum premieplicht Zvw (H2) &gt; &lt; min &gt; &lt; dag uit &gt; &lt; de &gt; &lt; begindatum premieplicht Zvw (H1) &gt;</p> <p>( indien ) &lt; de &gt; &lt; aantal dagen ZVW-plicht (U1) &gt; ( aan alle volgende voorwaarden voldoet ) :</p> <ul style="list-style-type: none"> <li>( indien ) &lt; de &gt; &lt; datum ingang actief militair (e) &gt; ( gelijk is aan ) &lt; afwezig (leeg) &gt;</li> <li>( indien ) &lt; de &gt; &lt; datum beëindiging actief-militair (f) &gt; ( gelijk is aan ) &lt; afwezig (leeg) &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>&lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>The following three separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground), <b>pattern 11</b> (conjunction), <b>pattern 4</b> (consistency check one value)</li> <li>“Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>2d</b></p> <p><b>Als</b> [aantal dagen ZVW-plicht] (U1) is kleiner dan 0</p> <p><b>Dan</b> [aantal dagen ZVW-plicht] (U1) = 0</p>	<p>&lt; Het &gt; &lt; aantal dagen ZVW-plicht (U1) &gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; het &gt; &lt; aantal dagen ZVW-plicht (U1) &gt; &lt; is kleiner dan &gt; &lt; 0 &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>&lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 8</b> (comparison with value)</li> <li>“Dan” is omitted, only for readability.</li> </ul>	

**BR 3: Aantal IB-dagen (U2)**

Original business rule NR	Business rule in pattern
<b>3</b> [aantal IB-dagen] (U2) = (maand uit [datum beëindigen (soort) belastingplicht] (h) min maand uit [datum ingang (soort) belastingplicht] (g)) maal 30 plus (dag uit [datum beëindigen (soort) belastingplicht] (h) min dag uit [datum ingang (soort) belastingplicht] (g))	< Het > < aantal IB-dagen (U2) > ( wordt berekend als ) < maand uit > < de > < datum beëindigen (soort) belastingplicht (h) > <min> < maand uit > < de > < datum ingang (soort) belastingplicht (g) > < maal> <30> <plus > < dag uit > < de > < datum beëindigen (soort) belastingplicht (h) > < min > < dag uit > < de > < datum ingang (soort) belastingplicht (g) >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following pattern is used: <b>pattern 3</b> (equate with basic ground)               <ul style="list-style-type: none"> <li>○ No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

**BR 4: Bijdrage-inkomen ZVW zonder alimentatie (H1)**

Original business rule NR	Business rule in pattern
<b>4</b> [bijdrage-inkomen ZVW zonder alimentatie] (H1) := [belastbare winst uit onderneming] (a) min [in winst begrepen loon] (g) min [winst deelvisser] (j) plus [totaal buitenlandse inkomsten uit dienstbetrekking] (b) plus [netto resultaat uit overige werkzaamheden] (c) plus [saldo periodieke uitkeringen na aftrekbare kosten] (d)	< De > < bijdrage-inkomen ZVW zonder alimentatie (H1) > ( wordt berekend als ) < de > <belastbare winst uit onderneming (a)> <min> < de > <in winst begrepen loon (g)> <min> < de > <winst deelvisser (j)> <plus> < de > <totaal buitenlandse inkomsten uit dienstbetrekking (b)> <plus> < de > <netto resultaat uit overige werkzaamheden (c)> < plus > < de > <saldo periodieke uitkeringen na aftrekbare kosten (d)>
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following pattern is used: <b>pattern 3</b> (equate with basic ground)               <ul style="list-style-type: none"> <li>○ No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

**BR 5 Bijdrage-inkomen ZVW aanslag (U1):**

Original business rule NR	Business rule in pattern
<p><b>5a</b></p> <p><b>ALS</b> [vorig jaar alimentatie overgangstarief] (f) = 'ja'</p> <p><b>DAN</b> [bijdrage-inkomen ZVW aanslag] (U1) := [bijdrage-inkomen ZVW zonder alimentatie] (H1)</p>	<p>&lt; De &gt; &lt; bijdrage-inkomen ZVW aanslag (U1) &gt; ( wordt gelijk gesteld aan ) &lt; de &gt; &lt; bijdrage-inkomen ZVW zonder alimentatie (H1)&gt;</p> <p>( indien ) &lt; de &gt; &lt; vorig jaar alimentatie overgangstarief (f) &gt; ( gelijk is aan ) &lt; ja &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 2</b> (equate with subject) and <b>pattern 4</b> (consistency check one value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>5b</b></p> <p><b>ALS</b> [vorig jaar alimentatie overgangstarief] (f) = 'nee'</p> <p><b>DAN</b> [bijdrage-inkomen ZVW aanslag] (U1) := [bijdrage-inkomen ZVW zonder alimentatie] (H1) plus [saldo alimentatie na aftrekbare kosten] (e)</p>	<p>&lt; De &gt; &lt; bijdrage-inkomen ZVW aanslag (U1) &gt; ( wordt berekend als )</p> <p>&lt; de &gt; &lt; bijdrage-inkomen ZVW zonder alimentatie (H1)&gt;</p> <p>&lt;plus&gt;</p> <p>&lt; het &gt; &lt;saldo alimentatie na aftrekbare kosten (e)&gt;</p> <p>( indien ) &lt; de &gt; &lt; vorig jaar alimentatie overgangstarief (f) &gt; ( gelijk is aan ) &lt; nee &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground) and <b>pattern 4</b> (consistency check one value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>5c</b></p> <p><b>ALS</b> [bijdrage-inkomen ZVW aanslag] (U1) is kleiner dan 0</p> <p><b>DAN</b> [bijdrage-inkomen ZVW aanslag] (U1) := 0</p>	<p>&lt; De &gt; &lt; bijdrage-inkomen ZVW aanslag (U1) &gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag (U1) &gt; &lt; is kleiner dan &gt; &lt; 0 &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 8</b> (comparison with value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

**BR 6. Niet relevante VA ZVW**

Original business rule NR	Business rule in pattern
<p><b>6</b></p> <p><b>ALS</b> indicatie ZVW aanslag verwijderen is gevuld en</p> <p><b>ALS</b> indicatie ZVW aanslag verwijderen = 'J'</p> <p><b>DAN</b> bijdrage-inkomen ZVW aanslag (U1) := 0</p>	<p>&lt; De &gt; &lt; bijdrage-inkomen ZVW aanslag (U1) &gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag (U1) &gt; ( aan alle volgende voorwaarden voldoet ) :</p> <ul style="list-style-type: none"> <li>( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; gevuld &gt;</li> <li>( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; J &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>&lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>The following three separate patterns are used and combined: <b>pattern 1</b> (equate with value), <b>pattern 11</b> (conjunction), and <b>pattern 4</b> (consistency check one value)</li> <li>"Dan" is omitted, only for readability.</li> </ul>	

**BR 7 Bijdrage ZVW aanslag (U1):**

Original business rule NR	Business rule in pattern
<p><b>7</b></p> <p>[bijdrage ZVW aanslag] (U1) := [toegepast bijdrage-inkomen ZVW] (a) maal [percentage verlaagd tarief] (b)</p>	<p>&lt; De &gt; &lt; bijdrage ZVW aanslag (U1) &gt; ( wordt berekend als )</p> <p>&lt; de &gt; &lt;toegepast bijdrage-inkomen ZVW (a)&gt; &lt;maal&gt; &lt; het &gt; &lt;percentage verlaagd tarief (b)&gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>&lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>The following pattern is used: <b>pattern 3</b> (equate with basic ground) <ul style="list-style-type: none"> <li>No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

**BR8 Bepalen toepassen artikel 9.4 (U2):**

Original business rule NR	Business rule in pattern
<b>8a:</b> ALS [artikel 9.4] (b) = [leeg]: en ALS [bijdrage ZVW aanslag] (a) is groter dan [aanslag-grens artikel 9.4] (d) DAN [toepassing artikel 9.4] (U2) := 21	< De > < toepassing artikel 9.4 (U2) > ( wordt gelijk gesteld aan ) < 21 >  ( indien ) < de > < toepassing artikel 9.4 (U2) > ( aan alle volgende voorwaarden voldoet ) : <ul style="list-style-type: none"> <li>• ( indien ) &lt; het &gt; &lt; artikel 9.4 (b) &gt; ( gelijk is aan ) &lt; leeg &gt;</li> <li>• ( indien ) &lt; de &gt; &lt; bijdrage ZVW aanslag (a) &gt;  &lt; is groter dan &gt; &lt; aanslag-grens artikel 9.4 (d) &gt;</li> </ul>
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following four separate patterns are used and combined: <b>pattern 1</b> (equate with value), <b>pattern 11</b> (conjunction), <b>pattern 4</b> (consistency check one value), and <b>pattern 9</b> (comparison with subject).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<b>8b:</b> ALS [artikel 9.4] (b) = <b>gevuld</b> en ALS [bijdrage ZVW aanslag] (a) is kleiner of gelijk aan [aanslag-grens artikel 9.4] (d) DAN: [toepassing artikel 9.4] (U2) := 35	< De > < toepassing artikel 9.4 (U2) > ( wordt gelijk gesteld aan ) < 35 >  ( indien ) < de > < toepassing artikel 9.4 (U2) > ( aan alle volgende voorwaarden voldoet ) : <ul style="list-style-type: none"> <li>• ( indien ) &lt; het &gt; &lt; artikel 9.4 (b) &gt; ( gelijk is aan ) &lt; gevuld &gt;</li> <li>• ( indien ) &lt; de &gt; &lt; bijdrage ZVW aanslag (a) &gt;  &lt; is kleiner of gelijk aan &gt; &lt; aanslag-grens artikel 9.4 (d) &gt;</li> </ul>
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following four separate patterns are used and combined: <b>pattern 1</b> (equate with value), <b>pattern 11</b> (conjunction), <b>pattern 4</b> (consistency check one value), and <b>pattern 9</b> (comparison with subject).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	



Original business rule NR	Business rule in pattern
<b>8c:</b>	< De > < toepassing artikel 9.4 (U2) >
ALS [artikel 9.4] (b) = "0"	( wordt gelijk gesteld aan ) < 35 >
DAN [toepassing artikel 9.4] (U2) := 35	( indien ) < het > < artikel 9.4 (b) > ( gelijk is aan )
	< 0 >
<b>Choices:</b>	
<ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value).</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<b>8d:</b>	< De > < toepassing artikel 9.4 (U2) >
ALS [artikel 9.4] (b) = "3"	( wordt gelijk gesteld aan ) < 0 >
DAN [toepassing artikel 9.4] (U2) := 0	( indien ) < het > < artikel 9.4 (b) > ( gelijk is aan )
	< 3 >
<b>Choices:</b>	
<ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value).</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

**BR 9 Vaststellen bedrag aanslag (U1):**

Original business rule NR	Business rule in pattern
<b>9a</b>	< Het > < bedrag aanslag (U1) >
ALS [toepassing artikel 9.4] (U2) = 35	( wordt gelijk gesteld aan ) < 0 >
DAN [bedrag aanslag] (U1) := 0	( indien ) < de > < toepassing artikel 9.4 (U2) >
	( gelijk is aan ) < 35 >
<b>Choices:</b>	
<ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value).</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<b>9b</b> ALS [toepassing artikel 9.4] (U2) = 35 DAN [indicatie-nihil-aanslag] (U3) := 'ja'	< De > < indicatie-nihil-aanslag (U3) > ( wordt gelijk gesteld aan ) < ja >  ( indien ) < de > < toepassing artikel 9.4 (U2) > ( gelijk is aan ) < 35 >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value).</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<b>9c:</b> ALS [toepassing artikel 9.4] (U2) ≠ 35 DAN [bedrag aanslag] (U1) := [bijdrage ZVW aanslag] (a)	< Het > < bedrag aanslag (U1) > ( wordt gelijk gesteld aan ) < de > < bijdrage ZVW aanslag (a) >  ( indien ) < de > < toepassing artikel 9.4 (U2) > ( niet gelijk is aan ) < 35 >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 2</b> (equate with subject) and <b>pattern 4</b> (consistency check one value).</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

#### BR 10 Saldo aanslag voor heffingsrente en boete (U4):

Original business rule NR	Business rule in pattern
[saldo aanslag voor heffingsrente en boete] (U4) := [bedrag aanslag] (U1) min [eerdere aanslag(en)] (f)	< Het > < saldo aanslag voor heffingsrente en boete (U4) > ( wordt berekend als ) < het > < bedrag aanslag (U1)> <min> < elke > < eerdere aanslag (f) >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following pattern is used: <b>pattern 3</b> (equate with basic ground) <ul style="list-style-type: none"> <li>○ No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

**BR11 Gecorrigeerd bijdrage-inkomen ZVW (U1):**

Original business rule NR	Business rule in pattern
<b>11a.</b> ALS [correctie-inkomen ZVW aanslag] (e) is gevuld  DAN [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'	< De > <gecorrigeerd bijdrage-inkomen aanwezig (H1)> ( wordt gelijk gesteld aan ) < J >  ( indien ) < de > < correctie-inkomen ZVW aanslag (e) > ( gelijk is aan ) < gevuld >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<b>11b.</b> ALS [correctie-inkomen ZVW aanslag] (e) is gevuld  DAN [gecorrigeerd bijdrage-inkomen ZVW] (U1) := [bijdrage-inkomen ZVW aanslag] (a) min ABS [correctie-inkomen ZVW aanslag] (e)	< De > <gecorrigeerd bijdrage-inkomen ZVW (U1)> ( wordt berekend als ) < de > < bijdrage-inkomen ZVW aanslag (a) > <min> < de > < ABS > < correctie-inkomen ZVW aanslag (e) >  ( indien ) < de > < correctie-inkomen ZVW aanslag (e) > ( gelijk is aan ) < gevuld >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground) and <b>pattern 4</b> (consistency check one value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<b>11c.</b> ALS [correctie-inkomen ZVW aanslag] (e) is leeg  DAN [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'N'	< De > <gecorrigeerd bijdrage-inkomen aanwezig (H1)> ( wordt gelijk gesteld aan ) < N >  ( indien ) < de > < correctie-inkomen ZVW aanslag (e) > ( gelijk is aan ) < leeg >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>11d.</b></p> <p>ALS [gecorrigeerd bijdrage-inkomen ZVW] (U1) is kleiner dan 0</p> <p>DAN [gecorrigeerd bijdrage-inkomen ZVW] (U1) is gelijk aan 0</p>	<p>&lt; De &gt; &lt;gecorrigeerd bijdrage-inkomen ZVW (U1) &gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; de &gt; &lt;gecorrigeerd bijdrage-inkomen ZVW (U1)&gt; &lt; is kleiner dan &gt; &lt; 0 &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 8</b> (comparison with value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

**BR12 Herleid bijdrage-inkomen ZVW aftrekmethode (U3):**

Original business rule NR	Business rule in pattern
<p><b>12a.</b></p> <p>[Herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'N'</p>	<p>&lt; De &gt; &lt;Herleid bijdrage-inkomen aftrek aanwezig(H3)&gt; ( wordt gelijk gesteld aan ) &lt; N &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following pattern is used: <b>pattern 1</b> (equate with value) <ul style="list-style-type: none"> <li>◦ No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>12b.</b></p> <p>ALS [bijdrage-inkomen ZVW aanslag buitenland] (d) is gevuld</p> <p>DAN [herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'J'</p>	<p>&lt; De &gt; &lt; Herleid bijdrage-inkomen aftrek aanwezig(H3)&gt; ( wordt gelijk gesteld aan ) &lt; J &gt;</p> <p>( indien ) &lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag buitenland (d) &gt; ( gelijk is aan ) &lt; gevuld &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 4</b> (consistency check one value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>12c.</b></p> <p>ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J'</p> <p>DAN [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) := [gecorrigeerd bijdrage-inkomen ZVW] (U1) min [bijdrage-inkomen ZVW aanslag buitenland] (d)</p>	<p>&lt; De &gt; &lt;Herleid bijdrage-inkomen ZVW aftrekmethode (U3)&gt; ( wordt berekend als )</p> <p>&lt; de &gt; &lt; gecorigeerde bijdrage-inkomen ZVW (U1)&gt; &lt;min&gt; &lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag buitenland (d) &gt;</p> <p>( indien ) &lt; de &gt; &lt;gecorrigeerd bijdrage-inkomen aanwezig (H1) &gt; ( gelijk is aan ) &lt; J &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground) and <b>pattern 4</b> (consistency check one value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>12d.</b></p> <p>ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'N'</p> <p>DAN [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) := [bijdrage-inkomen ZVW aanslag] (a) min [bijdrage-inkomen ZVW aanslagbuitenland](d)</p>	<p>&lt; De &gt; &lt;Herleid bijdrage-inkomen ZVW aftrekmethode (U3)&gt; ( wordt berekend als )</p> <p>&lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag (a)&gt; &lt;min&gt; &lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag buitenland (d) &gt;</p> <p>( indien ) &lt; de &gt; &lt;gecorrigeerd bijdrage-inkomen aanwezig (H1) &gt; ( gelijk is aan ) &lt; N &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground) and <b>pattern 4</b> (consistency check one value)</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

**BR13 Toegepast maximum bijdrage-inkomen ZVW (U4):**

Original business rule NR	Business rule in pattern
<b>13a.</b> <b>[toegepast maximum bijdrage-inkomen ZVW]</b> (U4) := [maximum bijdrage-inkomen ZVW] (g)	< De > < toegepast maximum bijdrage-inkomen ZVW (U4) > ( wordt gelijk gesteld aan ) < de > < maximum bijdrage-inkomen ZVW (g) >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following pattern is used: <b>pattern 2</b> (equate with subject)               <ul style="list-style-type: none"> <li>○ No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

Original business rule NR	Business rule in pattern
<b>13b.</b> ALS [aantal dagen ZVW-plicht] (c) is kleiner dan [aantal dagen in belastingjaar] EN ( [einde ZVW door overlijden] (f) = 'N' OF [einde ZVW door overlijden] (f) = <Leeg> ) DAN <b>[toegepast maximum bijdrage-inkomen ZVW]</b> (U4) := ( [aantal dagen ZVW-plicht] (c) delen door [aantal dagen in belastingjaar] ) maal [maximum bijdrage-inkomen ZVW] (g)	< De > <toegepast maximum bijdrage-inkomen ZVW (U4)> ( wordt berekend als ) < het > < aantal dagen ZVW-plicht (c) > <delen door> < het > < aantal dagen in belastingjaar > <maal> < het > < maximum bijdrage-inkomen ZVW (g)> ( indien ) < de > < toegepast maximum bijdrage-inkomen ZVW (U4) > ( aan alle volgende voorwaarden voldoet ): <ul style="list-style-type: none"> <li>• ( indien ) &lt; het &gt; &lt; einde ZVW door overlijden (f) &gt; ( aan tenminste één van de volgende voorwaarden voldoet ):               <ol style="list-style-type: none"> <li>1. ( indien ) &lt; het &gt; &lt;einde ZVW door overlijden (f) &gt; ( gelijk is aan ) &lt; N&gt;</li> <li>2. ( indien ) &lt; het &gt; &lt;einde ZVW door overlijden (f) &gt; ( gelijk is aan ) &lt; Leeg &gt;</li> </ol> </li> <li>• ( indien ) &lt; het &gt; &lt;aantal dagen ZVW- plicht (c) &gt; &lt; is kleiner dan &gt; &lt; het &gt; &lt;aantal dagen in belastingjaar&gt;</li> </ul>
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following four separate patterns are used and combined: <b>pattern 3</b> (equate with basic ground), <b>pattern 13</b> (Disjunction within conjunction), <b>pattern 4</b> (consistency check one value), and <b>pattern 9</b> (comparison with subject).</li> <li>• "Dan" is omitted, only for readability.</li> </ul>	

**BR 14 Heffingsruimte bijdrage-inkomen ZVW (U5):**

Original business rule NR	Business rule in pattern
<b>14 a:</b> [heffingsruimte bijdrage-inkomen ZVW] (U5) := [toegepast maximum bijdrage-inkomen ZVW] (U4) min [bijdrage-inkomen ZVW inhouding] (h)	< De > <heffingsruimte bijdrage-inkomen ZVW (U5)> ( wordt berekend als ) < de > <toegepast maximum bijdrage- inkomen ZVW (U4)> <min> < de > <bijdrage-inkomen ZVW inhouding (h)>
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following pattern is used: <b>pattern 3</b> (equate with basic ground)               <ul style="list-style-type: none"> <li>○ No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

Original business rule NR	Business rule in pattern
<b>14b:</b> ALS [heffingsruimte bijdrage-inkomen ZVW] (U5) is kleiner dan 0  DAN [heffingsruimte bijdrage-inkomen ZVW] (U5) := 0	< De > <heffingsruimte bijdrage-inkomen ZVW (U5)> ( wordt gelijk gesteld aan ) < 0 >  ( indien ) < de > < heffingsruimte bijdrage-inkomen ZVW (U5) > < is kleiner dan > < 0 >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 1</b> (equate with value) and <b>pattern 8</b> (comparison with value)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

**BR 15 Toegepast bijdrage-inkomen ZVW (U6):**

Original business rule NR	Business rule in pattern
<b>15a.</b> [toegepast bijdrage-inkomen ZVW] (U6) := [heffingsruimte bijdrage-inkomen ZVW] (U5)	< De > < toegepast bijdrage-inkomen ZVW (U6) > ( wordt gelijk gesteld aan ) < de > < heffingsruimte bijdrage-inkomen ZVW (U5) >
<b>Choices:</b> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following pattern is used: <b>pattern 2</b> (equate with subject)               <ul style="list-style-type: none"> <li>○ No condition included in the business rule, so only one pattern was required.</li> </ul> </li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>15b.</b></p> <p>ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) := 'J' en</p> <p>ALS [gecorrigeerd bijdrage-inkomen ZVW] (U1) is kleiner dan [toegepast bijdrage-inkomen ZVW] (U6)</p> <p>DAN [toegepast bijdrage-inkomen ZVW] (U6) := [gecorrigeerd bijdrage-inkomen ZVW] (U1)</p>	<p>&lt; De &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; ( wordt gelijk gesteld aan ) &lt; de &gt; &lt; gecorrigeerd bijdrage-inkomen ZVW (U1) &gt;</p> <p>( indien ) &lt; de &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; ( aan alle volgende voorwaarden voldoet ) :</p> <ul style="list-style-type: none"> <li>• ( indien ) &lt; de &gt; &lt; gecorrigeerde bijdrage-inkomen aanwezig (H1) &gt; (gelijk is aan ) &lt;J&gt;</li> <li>• ( indien ) &lt; de &gt; &lt; gecorrigeerde bijdrage-inkomen ZVW (U1) &gt; &lt; is kleiner dan &gt; &lt; de &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following four separate patterns are used and combined: <b>pattern 2</b> (equate with subject), <b>pattern 11</b> (conjunction), <b>pattern 4</b> (consistency check one value), and <b>pattern 9</b> (comparison with subject).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>15c.</b></p> <p>ALS [herleid bijdrage-inkomen aftrek aanwezig] (H3) := 'J' en</p> <p>ALS [herleid bijdrage-inkomen ZVW aftrekmethode] (U3) is kleiner dan [toegepast bijdrage-inkomen ZVW] (U6)</p> <p>DAN [toegepast bijdrage-inkomen ZVW] (U6) := [herleid bijdrage-inkomen ZVW aftrekmethode] (U3)</p>	<p>&lt; De &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; ( wordt gelijk gesteld aan ) &lt; de &gt; &lt; herleid bijdrage-inkomen ZVW aftrekmethode (U3) &gt;</p> <p>( indien ) &lt; de &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; ( aan alle volgende voorwaarden voldoet ) :</p> <ul style="list-style-type: none"> <li>• ( indien ) &lt; de &gt; &lt; herleid bijdrage-inkomen aftrek aanwezig (H3) &gt; ( gelijk is aan ) &lt;J&gt;</li> <li>• ( indien ) &lt; de &gt; &lt; herleid bijdrage-inkomen ZVW aftrekmethode (U3) &gt; &lt; is kleiner dan &gt; &lt; de &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following four separate patterns are used and combined: <b>pattern 2</b> (equate with subject), <b>pattern 11</b> (conjunction), <b>pattern 4</b> (consistency check one value), and <b>pattern 9</b> (comparison with subject).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	



Original business rule NR	Business rule in pattern
<p><b>15d.</b></p> <p>ALS [gecorrigeerd bijdrage-inkomen aanwezig] (H1) = 'N' en [herleid bijdrage-inkomen aftrek aanwezig] (H3) = 'N' en ALS [bijdrage-inkomen ZVW aanslag] (a) is kleiner dan [heffingsruimte bijdrage-inkomen ZVW] (U5)</p> <p>DAN [toegepast bijdrage-inkomen ZVW] (U6) := [bijdrage-inkomen ZVW aanslag] (a)</p>	<p>&lt; De &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; ( wordt gelijk gesteld aan ) &lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag (a) &gt;</p> <p>( indien ) &lt; de &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; ( aan alle volgende voorwaarden voldoet ) :</p> <ul style="list-style-type: none"> <li>• ( indien ) &lt; de &gt; &lt; gecorrigeerd bijdrage-inkomen aanwezig (H1) &gt; ( gelijk is aan ) &lt;N&gt;</li> <li>• ( indien ) &lt; de &gt; &lt; herleid bijdrage-inkomen aftrek aanwezig (H3) &gt; ( gelijk is aan ) &lt; N &gt;</li> <li>• ( indien ) &lt; de &gt; &lt; bijdrage-inkomen ZVW aanslag (a) &gt; &lt; is kleiner dan &gt; &lt; de &gt; &lt; heffingsruimte bijdrage-inkomen ZVW (U5) &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following four separate patterns are used and combined: <b>pattern 2</b> (equate with subject), <b>pattern 11</b> (conjunction), <b>pattern 4</b> (consistency check one value), and <b>pattern 9</b> (comparison with subject).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>15e.</b></p> <p>ALS [toegepast bijdrage-inkomen ZVW] (U6) is groter dan [maximum bijdrage-inkomen ZVW] (g)</p> <p>DAN [toegepast bijdrage-inkomen ZVW] (U6) := [maximum bijdrage-inkomen ZVW] (g)</p>	<p>&lt; De &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; ( wordt gelijk gesteld aan ) &lt; de &gt; &lt; maximum bijdrage-inkomen ZVW (g) &gt;</p> <p>( indien ) &lt; de &gt; &lt; toegepast bijdrage-inkomen ZVW (U6) &gt; &lt; is groter dan &gt; &lt; maximum bijdrage-inkomen ZVW (g) &gt;</p>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following two separate patterns are used and combined: <b>pattern 2</b> (equate with subject) and <b>pattern 9</b> (comparison with subject)</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

**BR 16 Niet relevante VA ZVW:**

Original business rule NR	Business rule in pattern
<p><b>16a.</b></p> <p>ALS indicatie ZVW aanslag verwijderen is gevuld en</p> <p>ALS indicatie ZVW aanslag verwijderen = J</p> <p>DAN gecorrigeerd bijdrage-inkomen ZVW (U1) = 0</p>	<p>&lt; De &gt; &lt; gecorrigeerd bijdrage-inkomen ZVW (U1) &gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; de &gt; &lt; gecorrigeerd bijdrage-inkomen ZVW (U1) &gt; (aan alle volgende voorwaarden voldoet) :</p> <ul style="list-style-type: none"> <li>• ( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; gevuld &gt;</li> <li>• ( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; J &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following three separate patterns are used and combined: <b>pattern 1</b> (equate with value), <b>pattern 11</b> (conjunction), and <b>pattern 4</b> (consistency check one value).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	

Original business rule NR	Business rule in pattern
<p><b>16b.</b></p> <p>ALS indicatie ZVW aanslag verwijderen is gevuld en</p> <p>ALS indicatie ZVW aanslag verwijderen = J</p> <p>DAN herleid bijdrage-inkomen ZVW aftrekmethode (U3) = 0</p>	<p>&lt; De &gt; &lt; herleid bijdrage-inkomen ZVW aftrekmethode (U3) &gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; de &gt; &lt; herleid bijdrage-inkomen ZVW aftrekmethode (U3) &gt; (aan alle volgende voorwaarden voldoet) :</p> <ul style="list-style-type: none"> <li>• ( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; gevuld &gt;</li> <li>• ( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; J &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following three separate patterns are used and combined: <b>pattern 1</b> (equate with value), <b>pattern 11</b> (conjunction), and <b>pattern 4</b> (consistency check one value).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	



Original business rule NR	Business rule in pattern
<p><b>16c.</b></p> <p>ALS indicatie ZVW aanslag verwijderen is gevuld en</p> <p>ALS indicatie ZVW aanslag verwijderen = J</p> <p>DAN heffingsruimte bijdrage-inkomen ZVW (U4) = 0</p>	<p>&lt; De &gt; &lt;heffingsruimte bijdrage-inkomen ZVW (U4)&gt; ( wordt gelijk gesteld aan ) &lt; 0 &gt;</p> <p>( indien ) &lt; de &gt; &lt; heffingsruimte bijdrage-inkomen ZVW (U4) &gt; (aan alle volgende voorwaarden voldoet) :</p> <ul style="list-style-type: none"> <li>• ( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; gevuld &gt;</li> <li>• ( indien ) &lt; de &gt; &lt; indicatie ZVW aanslag verwijderen &gt; ( gelijk is aan ) &lt; J &gt;</li> </ul>
<p><b>Choices:</b></p> <ul style="list-style-type: none"> <li>• &lt;Quantifier&gt; not included in original business rule, but is a fixed pattern part. Therefore, an appropriate article (i.e. de / het ) is chosen.</li> <li>• The following three separate patterns are used and combined <b>pattern 1</b> (equate with value), <b>pattern 11</b> (conjunction), and <b>pattern 4</b> (consistency check one value).</li> <li>• “Dan” is omitted, only for readability.</li> </ul>	