Creating flexible data quality validation processes using Business Rules

Joris Scheppers

April 6, 2009

Name	J.J.R. Scheppers
Number	0075337
Place	Deventer
Period	August 2008 - February 2009
Institute	University of Twente, Enschede, The Netherlands
Faculties	Electrical Engineering, Mathematics and Computer Science Management and Governance
Programmes	Telematics Industrial Engineering and Management
Company	Topicus, Deventer
Committee	dr. M.E. Iacob *1st university supervisor* Management and Governance dr. ir. M. van Sinderen *2nd university supervisor* Electrical Engineering, Mathematics and Computer Science J. Willems *1st company supervisor* Topicus W. de Jong *2nd company supervisor* Topicus

Abstract

Over the past few years, the quality of data has become increasingly important to organisations. This is caused by the fact that these organisations rely more and more on the data they collect to make decisions and to react to an ever faster changing environment. Changes in this environment mean changes in the internal processes of an organisation, which in turn puts more emphasis on the need for high quality data. Organisations such as those in the Dutch Health Care system have learned that using high quality data is of major importance. Yet the change in demands on this data is not being met by flexibility of the processes within these organisations.

Topicus is currently developing an application to automate the handling of electronic claim messages. The demand on the quality of these claims changes often, and must be met by a flexible validation process. However, this is currently not the case: the validation methods that are available have been implemented hard-code, which makes the process very inflexible to changes. This problem was the justification for this thesis: the application of the Business Rules approach to a data validation process to increase its flexibility and thereby its ability to react to changing demands from the environment.

The concept of data quality is explored in detail to ensure a good understanding of what causes data quality to be poor and with which methods and tools the quality of data can be assessed and influenced. The Business Rules approach is also extensively explored to be able to effectively apply this approach to the data validation process. The result is a combined approach to the validation of the quality of data using checks stated in Business Rules and evaluated using a Business Rule Engine from ILOG.

The results of the prototype are positive: the rule engine performed well and even uncovered some quality defects that were not expected initially. Subsequently, this thesis concludes with the recommendation to start using Business Rules in future projects where a more flexible data quality validation process is needed. However, more research has to be conducted to fully grasp the change in performance and to assess what exactly the impact of the use of Business Rules will be in Topicus' applications.

List of Figures

1.1 Dutch health care chain (UML Communication Diagram notation) 3
1.2 Graphical representation of HA304 message 5
1.3 Black-box view of Claim Factory (BizzDesigner / ISDL notation) 8
1.4 Inner processes of Claim Factory (BizzDesigner / ISDL notation) 10
1.5 Research model (based on [Verschuren], chapter numbers between brackets) 15
1.6 Document structure (chapter numbers between brackets) 16

5.1 Prototype class diagram (UML class diagram notation) 40
5.2 Message evaluation sequence diagram (UML sequence diagram notation) . 43
5.3 Inner processes of Claim Factory with validation process (BizzDesigner / ISDL notation) 45
5.4 'Payment to' Range check rule (IntelliRule format) 47
5.5 'Information System code' If-then rule (IntelliRule format) 47
5.6 'Begin-End-date claim period' Cross-check rule (IntelliRule format) 47
5.7 'Total amount' Zero-control rule (IntelliRule format) 48
5.8 HA-message Rule flow (ILOGs RuleFlow editor notation) 49

List of acronyms

ACErules	Attempto Controlled English Rules
AGB	Algemeen Gegevensbeheer (General Data Management)
BPMS	Business Process Management System
BRE	Business Rule Engine
BRG	Business Rule Group
CHA	ClearingHouse Apothekers
DFD	Data Flow Diagram
ECA	Event Condition Action
EI	Externe Integratie (External Integration)
ERP	Enterprise Resource Planning
HA	Huisartsen (General Practitioners)
HTML	HyperText Markup Language
IS	Information System
NDB	Nedasco Declaratie Bericht (Nedasco Claim Message)
OMG	Object Modeling Group
R2ML	REWERSE Rule Markup Language
RDF	Resource Description Framework
RS	Real-world System
RuleML	Rule Markup Language
SBVR	Semantics of Business Vocabulary and Business Rules
SOA	Service Oriented Architecture
UI	User Interface
UML	Unified Markup Language

XMI	XML Metadata Interchange
XML	Extensible Modeling Language

Contents

1 Introduction **1**
- 1.1 Care Chain . . . 1
  - 1.1.1 Overview . . . 1
  - 1.1.2 Vektis and VECOZO details . . . 3
  - 1.1.3 Problems . . . 6
- 1.2 Case . . . 7
  - 1.2.1 Introduction . . . 7
  - 1.2.2 As-is situation . . . 7
  - 1.2.3 Problems . . . 11
  - 1.2.4 Flexibility . . . 12
- 1.3 Research . . . 12
  - 1.3.1 Objectives . . . 12
  - 1.3.2 Questions . . . 13
    - 1.3.2.1 Central research question . . . 13
    - 1.3.2.2 Sub questions . . . 14
  - 1.3.3 Approach . . . 14
  - 1.3.4 Thesis structure . . . 15

2 Data quality **17**
- 2.1 Need for data quality . . . 17
- 2.2 Defining data quality . . . 17
- 2.3 Determining data quality . . . 18
  - 2.3.1 Validation methods . . . 19
- 2.4 Summary . . . 19
- 2.5 Relation to case . . . 20
  - 2.5.1 Data Quality Assurance . . . 20

3 Business Rules **23**
- 3.1 History . . . 23
- 3.2 Definition . . . 23
- 3.3 Classification . . . 24
  - 3.3.1 Fundamental classification . . . 24
    - 3.3.1.1 Structural assertions . . . 24
    - 3.3.1.2 Action assertions . . . 25
    - 3.3.1.3 Derivations . . . 25
  - 3.3.2 Other classifications . . . 26

3.3.2.1 Integrity Maintenance rules . . . 26
3.3.2.2 Service composition rules . . . 26
3.3.2.3 Business Process integration rules . . . 26
3.3.2.4 Event-Condition-Action (ECA) rules . . . 27
3.4 Specification . . . 27
3.4.1 Classic specification methods . . . 28
3.4.2 Contemporary specification methods . . . 28
3.4.2.1 Near-natural languages . . . 28
3.4.2.2 Extensible Markup Language (XML)-based languages . . . 29
3.4.2.3 Rule Engine specific languages . . . 29
3.4.3 Applicability . . . 30
3.4.4 System components . . . 30
3.4.4.1 Business Rule Engine . . . 30
3.4.4.2 Business Rule Repository . . . 30
3.4.4.3 Business Rule authoring tools . . . 31
3.5 Summary . . . 31
3.6 Relation to case . . . 31

4 Combination method **34**
4.1 Assuring data quality using Business Rules . . . 34
4.2 Creating flexibility in Data Quality validation processes . . . 34
4.3 Other researches . . . 35
4.4 Summary . . . 36

5 Prototype **38**
5.1 Design . . . 38
5.1.1 Environment . . . 38
5.1.2 User Interface . . . 38
5.1.3 Object Model . . . 39
5.1.4 Rules . . . 39
5.1.5 Rule sets and -flow . . . 40
5.1.6 System components . . . 41
5.1.7 Validator class . . . 42
5.1.8 Execution sequence . . . 42
5.2 Implementation . . . 42
5.2.1 Environment . . . 42
5.2.2 System components . . . 44
5.2.3 Object model . . . 44
5.2.4 Rule project . . . 46
5.2.4.1 Rule set parameters . . . 46
5.2.4.2 Rules . . . 46
5.2.4.3 Rule Flow . . . 48
5.2.5 Validator class . . . 48
5.3 Feasibility . . . 50

5.4 Summary 51

6 Validation proposition **52**
6.1 Criteria 52
6.2 Performance. 53
6.2.1 Indicators 53
6.3 Summary 54

7 Conclusions and recommendations **55**
7.1 Conclusions 55
7.2 Recommendations 57
7.3 Open issues 57

I Appendices **63**

A **64**

B **66**

C **68**

D **70**

1 Introduction

This chapter provides background information on this thesis, outlines the research and introduces the case study and its problems.

1.1 Care Chain

In the Supply Chain Management concept, the management of a network of interconnected businesses involved in the ultimate provision of product and service packages, the coordination of supply and demand is what gives one supply chain the competitive advantage over another. The exchange of information between supplying and consuming participants in the chain is a very important activity. This communication can only be done effectively if some pre-defined way of information exchange is used. In a typical supply chain, most participants will have ERP-like software applications to manage aspects like inventory and production quantities and will communicate with supplier(s) and consumer(s) using some protocol definition.

1.1.1 Overview

Though the Dutch Health Care chain (called 'care chain' in the remainder of this document) is not a typical supply chain, it does have similar features: the care chain contains multiple participants who collectively provide a product or service, there is a 'producing' and 'consuming' side of the care chain and communication between participants in the care chain is vital for correct delivery of health care.

The producing and consuming participants in the care chain are the persons and organisations providing health care and the patient who receive the health care, respectively. Because The Netherlands has a Social Security system, every person that lives in The Netherlands has an obligation to pay insurance fees and every Dutch insurance company has the obligation to accept any person who wants to be insured. This adds a third party to the Health Care chain: the insurance company. The so-called 'Basic Insurance Law' (Basisverzekeringswet) defines for which types of care the costs are being paid by the insurance companies.

In principle, communication between these three participants is simple: the patient receives care from the care provider. The care provider then issues a claim with the insurance company where the patient is insured. When the claim is handled, either the provided care is of the type that is (partly) covered by the insurance company, or the patient is fully responsible for paying for the received care. In both cases, the health care provider receives the payment from the insurance company, in the latter case the patient also receives an invoice to pay for (the rest of the amount for) the received care.

There are exceptions to this pattern, for example in dental care. When a patient receives dental care, that patient must first pay the invoice he/she gets from the care provider before issuing a claim with his/her insurance company.

With each issued claim a return message is constructed by the participant that evaluates the claim, which contains information about the actions that were taken to come to the amount that is being payed. Using the information in that return message the care provider that initially issued the claim can update its administration. Except for the invoicing, all the communication is done electronically.

Because the Basic Insurance Law was designed to transform the care chain to a free market system, a lot of new insurance companies have entered the Dutch Health Care market. The law also allows these insurance companies to establish contracts with care providers of their choice to agree on more favorable care rates. This causes one insurance company to provide the same care for a smaller premium than another insurance company who does not have a contract with this care provider and thus create a better competitive position for itself. Also, every person in The Netherlands who falls under the Basic Insurance Law has a right to change insurance companies once per year, at the end of the year.

All these issues put enormous pressure on the exchange of information regarding patient's insurance, provided care to patients and contracts between insurance companies and health care providers. Errors in this exchange of information results in incorrect payments to care providers or patients, incorrect contractual terms between care providers and insurance companies or even (often irreversible) errors in provided health care.

The only way all the participants in the care chain can effectively exchange information is by using some predefined and mutually agreed-upon protocol. For this reason, the Dutch government has instated two additional participants in the chain: Vektis and VECOZO. Vektis is an organisation set-up by the Dutch government to create and maintain the protocol message standards. The VECOZO platform was created by a joint-effort of insurance companies. This platform acts as a central hub for communication between health care providers and insurance companies. The details of these two organisations are described later.

There are two additional types of participants that are present in the care chain: agencies and intermediaries. These participants mainly provide services to health care providers and insurance companies to outsource respectively the issuing and the handling of Health Care claims. The agencies operate on the 'producing' side of the care chain, the intermediaries operate on the 'consuming' side.

Specific health care specialisation groups[1] have agencies that handle the issuing of claims for the different providers within a health care specialisation group. For example, CHA (Clearing House Apothekers) handles claim issuing for pharmacies. This has an advantage for individual health care providers, who can now concentrate more on their patients and less on the administrative processes.

On the consuming side of the care chain, certain intermediaries take over the handling of claims for one or more insurance companies. In most cases, one such intermediary

[1] i.e. hospitals, pharmaceutical care, general practitioners, obstetric care etc.

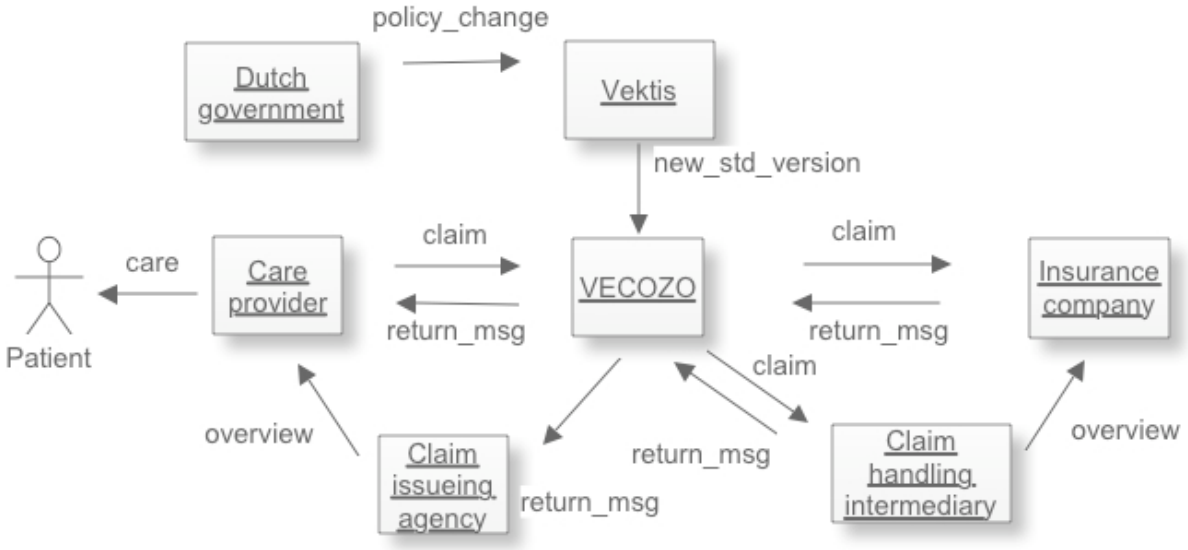

Figure 1.1: Dutch health care chain (UML Communication Diagram notation)

handles the claim evaluation for multiple insurance companies. The intermediary provides overviews (reports) of handled claims to the insurance company and often issues payment assignments.

An overview of the care chain is shown in figure 1.1. It shows the flow of claims and return messages between health care providers and insurance companies as well as the position of Vektis and VECOZO. The flow of payment traffic (invoices and payments) is not shown in this figure.

1.1.2 Vektis and VECOZO details

Vektis As described in section 1.1.1, Vektis is the organisation that creates and maintains the protocol message standards with which electronic health care claims are exchanged. These message standards describe the information that is needed to specify a claim for a certain health care specialisation group. Almost every care group has its own set of message standards. A set of standards consists of a claim message standard and a return message standard. The claim message is used to specify the actual claim, the return message is used to specify the status of that claim. The return message contains the same message as the original claim it was based on, with the addition of a number of status fields at the end of each record. These fields are used to specify feedback information, such as errors in the message that were observed during the evaluation process of the receiving participant.

In total there are 13 sets of claim message standards. These sets are all described and defined in the so-called 'Externe Integratie' (EI)-program on the Vektis website[Vektis].

Every message standard consists of a number of 'records.' The composition of records in a message standard as well as the number of each type of record depends on that

message standard.

Each record consists of a number of elements that can (and in most cases *have to)* be used to specify the claim. These elements are called 'fields'. The standard dictates the position and the length (in number of characters) in the record that the different fields should occupy and whether they are mandatory, conditional or optional. Mandatory and conditional fields have to be filled with specific information, for example a date or a code from a list of possible treatments[2]. These lists are also maintained by Vektis. Optional fields may contain any type of information..

Figure 1.2 shows a graphical representation of one of the message standards: the General Practitioner's EI-standard message 'HA'. Only a small selection of fields is shown, the actual standard consists of 5 different record types which in total consist of 104 different fields. Some records can have multiple instances within one message, as shown in the figure. The HA-message consists of one Opening and one Closing record. The message must contain 1 or more (denoted by 'n') Patient-records, which are denoted in the figure by 'p'. Each Patient-record is associated with at most 1 Debit-record (denoted by '1 per p'). Each Patient-record is associated with at least one (denoted by 'm') Treatment-record. The value for 'm' is not necessarily the same for each Patient-record.

The message itself is encoded using an ASCII String representation before it is electronically transported between participants in the care chain. Details about the field definitions and their possible values are located at the Vektis website[Vektis]

VECOZO As described in section 1.1.1, the VECOZO-platform is a central participant in the care chain, as most electronic claims pass through it. At the moment, VECOZO provides the following services[VECOZO]

- Claim handling: health care providers can submit claims directly through an Electronic Claim Portal. These claims will be encrypted and sent to the concerning insurance company. This service is mostly used by the provider's local software suite to automatically send the claims to VECOZO. VECOZO will then ensure that the claim is sent to the right insurance company.

- Insurance Rights look-up: VECOZO provides a possibility to check a patient's insurance details on-line. These details may include where the patient is insured and the kind of insurance policy. This service can also be integrated in the provider's local software suite.

- Secure message exchange: messages can be sent between health care providers and insurance companies using certificates to ensure security.

- AGB (Algemeen Gegevensbeheer, General Data Management) consulting: through this service, certain Health Care provider's information (i.e. addresses) can be acquired by insurance companies. This service also provides a way to check which providers have contracts with which insurance companies.

[2]a consult at the General Practitioner's office is an example of a treatment

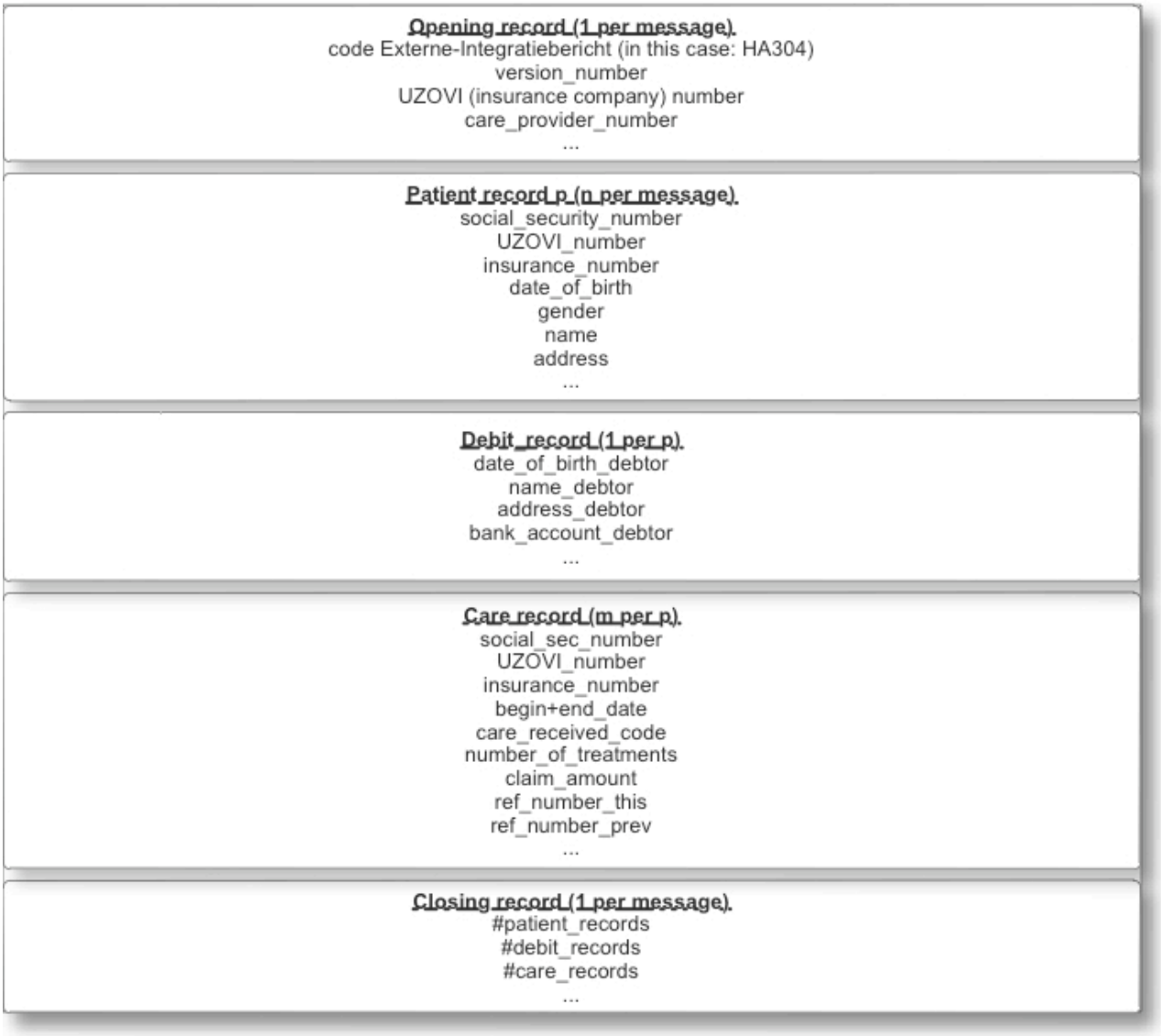

Figure 1.2: Graphical representation of HA304 message

- Digital contracting: certain Health Care providers don't receive the contracts from insurance companies on paper, but instead do this digitally through the VECOZO portal.

1.1.3 Problems

In theory, the communication between care providers and insurance companies should be flawless because of the protocol standards and the central VECOZO platform. However, in practice, there are some problems that occur within the care chain. Some of these problems are described below:

1. Currently, some parties (such as big insurance companies) have formed their own interpretation of the standards, because some elements in the standards can be interpreted in different ways. These parties have such a strong financial and organisational position that they can enforce the use of 'their' standard on every participant that communicates with them. This defeats the purpose of having one universal communication standard, as some parties have to be able to interpret multiple 'dialects' instead of just one 'language'. The cause of this problem is the fact that there is no central authority that enforces the use of one version of the standard.

2. Health Care insurance companies still have to be able to handle paper claims issued by individual patients. These paper claims exist because some health care specialisation groups do not have an EI-standard. One such health care specialisation group is Alternative Health Care. Most insurance companies provide (partial) compensation for the alternative treatments, but because no EI-standard exists for these types of treatments, a paper claim is issued manually. Other health care providers simply do not support the electronic claiming and still send their invoice directly to the patient. The patient has to pay the invoice in advance and can claim their restitution by sending the invoice to their insurance company.
   The total number of these paper claims take up around 20 percent of the total number of issued claims. The insurance companies have to be able to handle these claims, or else they face the risk of losing a lot of clients. A lot of insurance companies have whole departments whose only responsibility it is to insert paper claims into the digital system. The conversion from paper to electronic claim is a time-consuming and error-prone activity.

3. Because the government changes the legislation concerning Health Care from time to time, a new version of the message standards is issued by Vektis each time important changes are made in the legislation. When this happens, the software that handles claims at each participant in the health care chain has to be updated as well. This can be a time-consuming and thus costly activity.

4. During the transition period between two versions of a message standard, it can occur that one participant in the care chain already complies to the new standard,

where another participant does not yet comply. This could result in communication problems and thus in a lot of incorrect claim acceptances/denials.

1.2 Case

1.2.1 Introduction

Topicus is an innovative ICT Service Provider which focuses mainly on chain integration in the Finance, Education and Health Care sectors. This chain integration is achieved by providing multiple participants in the chain with 'Software as a Service' solutions to improve administrative processes. One of Topicus' clients is Nedasco, a financial intermediary, which has the authority to handle Health Care claims for a number of Dutch Health Insurance companies. Nedasco's position in the care chain is shown in figure 1.1 as 'intermediary'. Topicus is currently developing the so-called 'Claim Factory' ('Declaratiefabriek'), a software suite that enables Nedasco to automatically evaluate and process the claims it receives.

During the development of the Claim Factory, Topicus encountered a number of issues. These issues are mainly related to the validation of received claim messages. The validation process is a process which checks if a received claim message complies to the message standard it claims to be constructed with.

When a new version of the message standard is released by Vektis, the validation process has to be updated to be able to successfully validate message constructed in this new message standard version[3]. In this case a simplified view on the Claim Factory is used. In reality, besides the digital VECOZO-messages the Claim Factory also receives claims directly from CHA(described in section 1.1.1) and paper claims. Details about the Claim Factory and the validation process in particular are described below.

1.2.2 As-is situation

In this section the current architecture of the Claim Factory is explored.

Black-box Claim Factory The Claim Factory evaluates the amount of money that will be reimbursed by looking at the client's insurance policy and the treatment that patient has received. It uses the *VECOZO claim message* as input and produces a *return message* and 0 or more *bookings* (payment order) as output. This black-box view is shown in figure 1.3.

Inner processes Claim Factory When this black-box is opened, the inner processes of the Claim Factory are visible. When VECOZO sends a *claim message* to the Claim Factory, it is received by a *web service component.* This component then transfers the message to the first *transformation module.* This module transforms the ASCII String

[3]The validation process is not the only part of the Claim Factory that has to be updated when a new version of the message standard is released, but it is the focal point for this research.

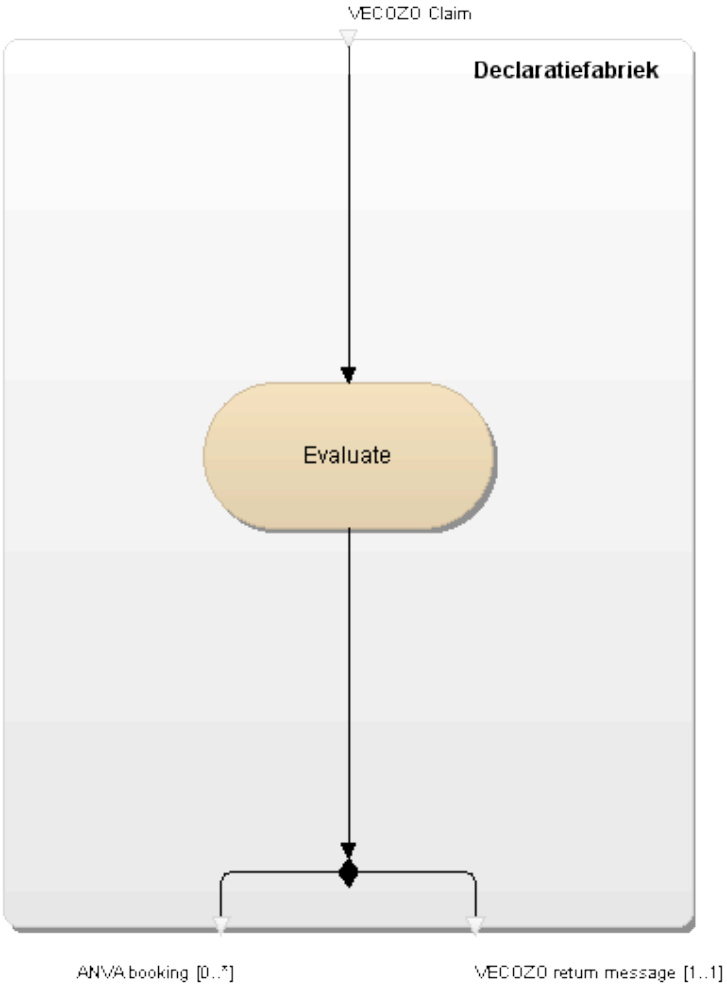

Figure 1.3: Black-box view of Claim Factory (BizzDesigner / ISDL notation)

model into a predefined *Object model* using a mapping from ASCII to Object notation. This mapping only knows three data-types to make the mapping: DATETIME (to model dates and time, although the TIME-component is not used), STRING (to represent any combination of characters) and INTEGER (which represents a natural number). The Object model is then used as input for the second *transformation module.* During this transformation information in the message is 'enriched' with information known by Nedasco about the participants which are present in the claim message, for example the claim history of a patient. This module transforms the first Object model into another Object model called the Nedasco Claim Message ('Nedasco Declaratie Bericht') or *NDB.* This model is only used within Nedasco.

The next step is the Evaluation process. The *evaluation process* performs the actual evaluation and determines which amount has to be paid. The actual evaluation is performed on so-called claim '*lines*'. One such line contains one Patient-record combined with one treatment-record. The reason for this division into lines is that a claim message can contain several claims for several different patients and every patient-treament-combination is evaluated independently[4]. When every treatment record is evaluated, all the claim lines are assembled to form the original message.

The exporter process waits for all the lines in a message to be evaluated. If no error has occurred that would be a reason to reject the claim, a booking is created for each line in the claim. This set of bookings is then sent to ANVA, the Back Office application responsible for the payouts by Nedasco. When every step described in this paragraph is executed *correctly*, a return message (which is based on the original claim message) is constructed by the exporter module. This message describes how much and because of what reason the amount is paid out. This amount can of course be zero, for instance when a patient's insurance policy did not cover the treatment at all, or if some information was not correctly specified in the claim. If something went *wrong* in the evaluation process, a return message is constructed describing the part of the claim where the error was detected. Of course, in this situation no payment orders are issued.

In both situations (correct or erroneous claim) the return message which contains the original claim message with the comments from the evaluation process attached to the fields,(which provide feedback for the participant that issued the claim) is sent back to the participant who issued the claim. This participant's information system will register this return message and, in the case of a rejected claim message, will most likely correct the error(s) and re-send the claim.

The inner processes of the Claim Factory are shown in figure 1.4.

As described in section 1.1, a successful exchange of messages can only occur when both parties in the exchange use the same definition of the message standard. The message standard dictates the conditions the information inside the message must comply to in order to eliminate any dispute about what is meant with the information stated in the message.

[4]as described in section 1.1.2

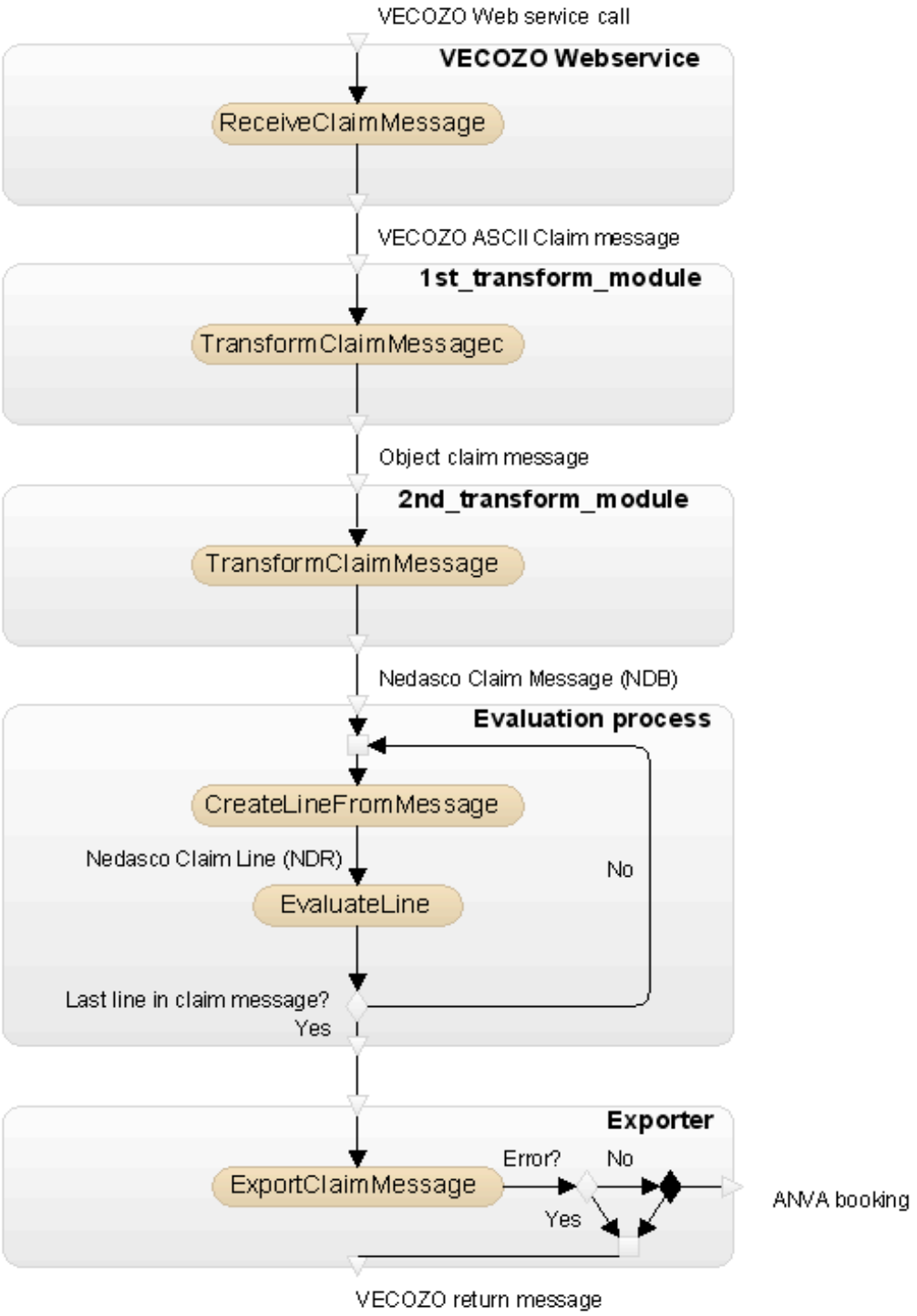

Figure 1.4: Inner processes of Claim Factory (BizzDesigner / ISDL notation)

1.2.3 Problems

The validation process mentioned in section 1.2.1 is not an explicitly defined process, it is therefor not present in figure 1.4. However, it is a process in the sense that it is a collection of related activities that produce a specific service, namely that of the validation of the quality of a claim message. From here on, this thesis defines the 'validation process' as the collection of related activities that take place to verify the conformance of a claim message to the corresponding message standard.

The validation process brings about two related problems, which are related to problem 3 stated in section 1.1.3:

1. The validation process is very complicated. It has to validate different types of message standards as well as different versions per type. Also, all message standards contain conditional fields which are hard to validate. Because of time constraints and design decisions, Topicus has implemented a limited number of validation checks in different parts of the Claim Factory, primarily in the evaluation processes. These checks are also hard-coded in the implementation, which makes the validation process hard to update when a new message standard is released, and costs a lot of time.
   Also, Topicus only concentrated on the implementation of validation of *mandatory* fields of a claim message. This decision could result in messages that pass the validation process which are not constructed in the way the message standard describes. It could be the case that this error is noticed later on in the evaluation process, where such errors are much more costly.

2. Currently no procedure exists to automatically update the software in the health care chain when a new version of the message standard is released. Vektis does have a channel to which an organisation can subscribe which broadcasts information about upcoming releases, but the update process itself is still done manually. This costs a lot of time and money.

The urgency for a solution to the second problem is less than that of the first problem, because at this point in time an ad-hoc solution for problem 2 already exists, using a script to translate the standard definition stated in HTML from the Vektis website to an Object model which can be used to update the first transformation module described in section 1.2.2. This results in a mapping from ASCII to the Object model, which can be used to automate the updating of the first transformation module described in section 1.2.2.

The need for a more *flexible* validation process is a lot higher because the validation process is the most thorough check that is performed on every claim that is handled by the Claim Factory. If the validation process lets a message pass which should *not* get passed, it means that it will cause a problem later on in the evaluation process. Thus increasing the flexibility of the validation process has a big impact on the system as a whole.

Also, this module is the same for every participant in the chain that handles claim messages. Since Topicus focuses on chain integration, this module can be used for prac-

tically every participant in the care chain and can therefore have a very large impact on the performance in every participant in the care chain. Used in cooperation with the update script described above, this provides a partial solution to the second stated problem.

In conclusion, this thesis will be based on solving problem 1. The formal specification of the research and its objective is given in section 1.3.

1.2.4 Flexibility

The previous sections mentioned the concept of flexibility. Because this concept plays an important part in this research, the concept must be explored and defined. Van Eijndhoven researched flexibility in business processes[Eijndhoven2008]. He noticed that the term was often mentioned, but was rarely properly defined and measured. Van Eijndhoven adopted the dimensions of flexibility expressed by Kasi and Tang in [Kasi2005]. These dimensions are:

1. Time - the time it takes to adapt the process to a change in the environment
2. Cost - the cost that is related to changing the process
3. Ease - the easiness with which the process is changed

Combining these three dimensions results in a method to measure the flexibility of a process, where a process is more flexible if it can be changed in less time, with less cost and with more ease relative to another process. The first two dimensions, Time and Cost, are easily measured, while the Ease-dimension is a bit harder. Van Eijndhoven defined indicators for the Ease-dimension as the number of items that have to be changed and the necessary steps that have to be taken to translate the requirements of a process to the actual implementation.

To make business processes more flexible, the concept of Business Rules has been developed. As stated by the BRG, the Business Rules Group, in [BRG2000], the appliance of the Business Rules concept generally increases the flexibility of business processes. Some other recent research projects, such as [vonHalle2002, Vasilecas2007, Chanana2007, Eijndhoven2008], have come to this conclusion as well. The application of Business Rules to the problem stated before could increase the flexibility of the claim evaluation process. That is why this thesis concentrates on the appliance of Business Rules to the problem stated in section 1.1.3.

1.3 Research

1.3.1 Objectives

As is described in section 1.2.3, the current implementation of the message validation process is inflexible regarding message standard updates. As stated in section 1.2.4, the appliance of the Business Rules concept could improve the flexibility of that process. Combining these aspects results in the goal of this thesis:

To find a method to increase the flexibility of message validation in the Dutch health care chain by using Business Rules and to demonstrate this in a prototype.

To be able to reach this objective, a number of research steps are taken. These steps are:

- creating an overview of various Business Rules concepts and technologies
- creating an overview of various data quality validation concepts and techniques
- researching a way to combine the previous two steps, i.e. to assure data quality using Business Rules
- creating a prototype to demonstrate the applicability of the previously stated concept to the case

1.3.2 Questions

With the problem definition of section 1.2.3 and research objectives of section 1.3.1 in mind, the following central research question is defined:

1.3.2.1 Central research question

How can Business Rules be used in a process that validates messages constructed using some standard to increase the flexibility in handling changes in this standard?

To be able to answer this central research question there are several aspects of the problem area that need to be researched. This leads to a decomposition of the central research question into several subquestions, which are answered independently throughout this report.

The first set of subquestions is centered on the concept of data quality. As mentioned in section 1.2.3 the quality of a claim message has to be determined before it can be processed. The notion of data quality is rather intuitive, but greater knowledge of data quality is necessary to be able to better understand what causes data quality to be poor and how the quality of data can be determined and influenced. The data quality subquestions explore the concept of data quality and methods to determine the quality of data.

The second set of subquestions is centered on the concept of Business Rules. As mentioned in section 1.2.4 business processes can be made more flexible by applying the Business Rules concept. The Business Rules subquestions explore the concept of Business Rules, the way they are classified and specified and the way they can be created and stored.

The third set of subquestions is centered on the appliance of Business Rules concepts to the data quality determination methods and how that application can increase the flexibility of data quality validation processes. The fourth set of subquestions is centered on the combination of data quality and Business Rules concepts in practice.

1.3.2.2 Sub questions

Data quality

- What is data quality?
- How is data quality determined?
- How can the quality of messages constructed in some message standard be validated?

Business Rules

- What are Business Rules?
- What technologies are available to classify and specify Business Rules?
- What technologies are available to create and store Business Rules and sets of Business Rules?

Using Business Rules techniques in data quality assurance

- How can Business Rules techniques be used to assure data quality?
- How can Business Rules techniques increase the flexibility of a data quality validation processes?

Data quality assurance using Business Rules in practice

- Can Business Rules be applied effectively to enhance the flexibility of the Claim Factory?
- What burdens and/or benefits does this combined approach have in the context of the Claim Factory?
- What future research has to be undertaken to enhance the solution to the flexibility problem?

1.3.3 Approach

The methodology of Verschuren and Doorewaard[Verschuren] is adopted to structure the research activities. The resulting structure is shown in figure 1.5. First, research will be done on the topics of data quality and Business Rules. This leads to an overview of the technologies and tools as well as the most important concepts in these domains. The general concepts of data quality will be related to the general concepts of Business Rules to find relations between them. With these findings, research will be done on a method to integrate the two domains, which forms the basis for a prototype to demonstrate and validate the feasibility and performance of this method. This validation will be done using

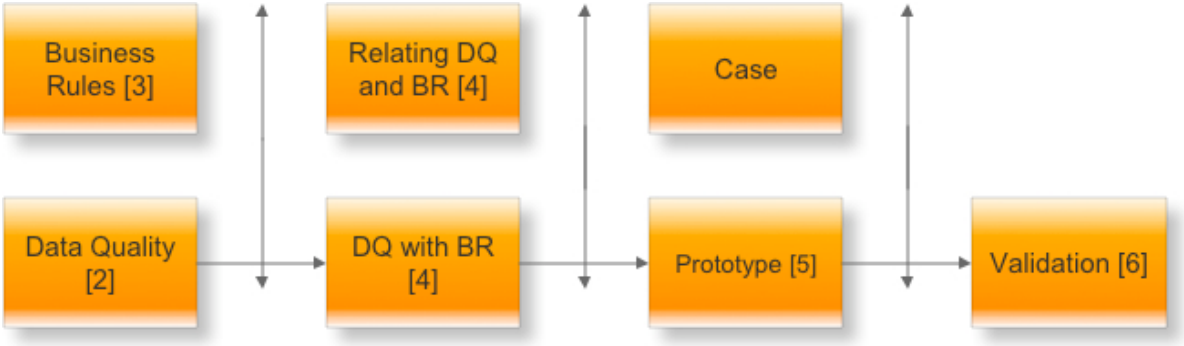

Figure 1.5: Research model (based on [Verschuren], chapter numbers between brackets)

the improvement in flexibility as a performance criterion. The feasibility is determined by using the stability or decrease of average time a claim message spends in the Claim Factory and the stability or decrease of number of false positives and false negatives as feasibility constraints.

1.3.4 Thesis structure

The structure of this BSc thesis report is represented graphically in figure 1.6.

- In the first chapter of this document the research is introduced and the design of the research is outlined.
- In chapter two and three, the two major elements of this research (namely data quality and Business Rules) are explored.
- In chapter four the method that combines data quality validation and Business Rules is discussed.
- Chapter five describes the design and implementation of the prototype which demonstrates the method discussed in chapter four.
- Chapter six discusses the validation of the method.
- Chapter seven provides conclusions and recommendations as well as some future research proposals.

At the end of each chapter, the relevant concepts, techniques and methods in each chapter will be linked to the case of Topicus. This will provide insight in the application of the researched material in a practical case.

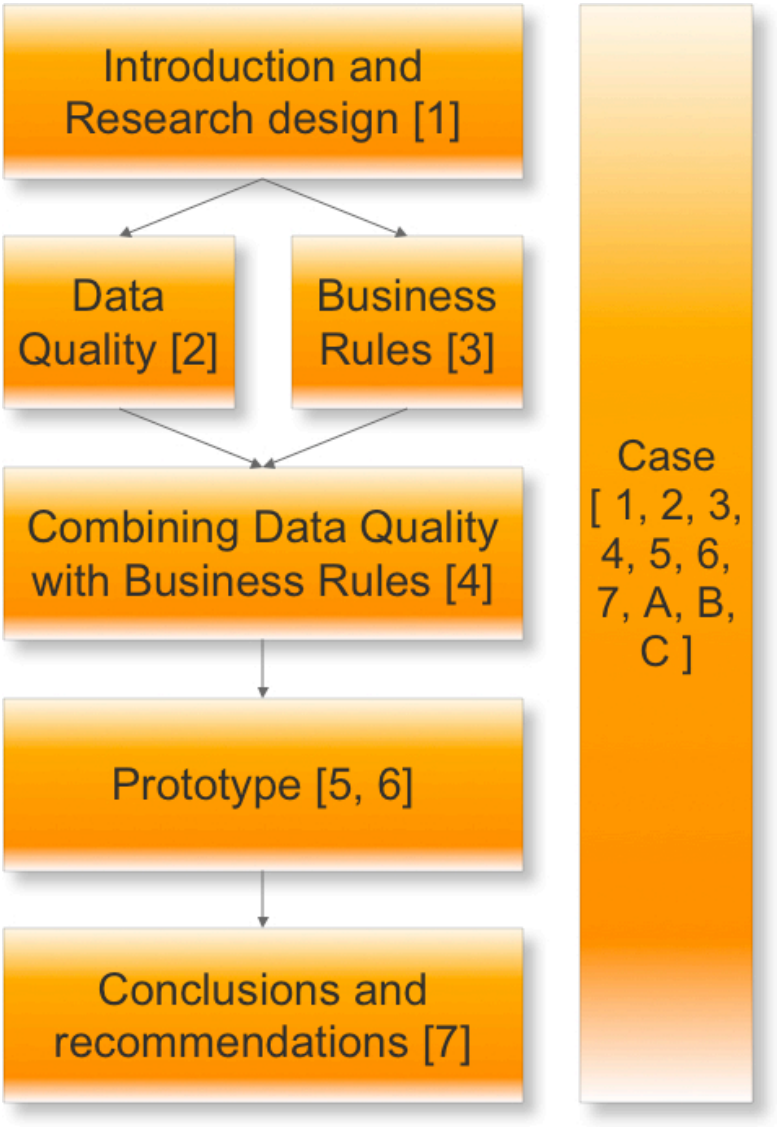

Figure 1.6: Document structure (chapter numbers between brackets)

2 Data quality

This chapter explores the concept of data quality. It provides the definition used in this thesis, lists means to measure the quality of data and proposes a way to measure the quality of the messages used in the case study.

2.1 Need for data quality

The need for 'clean', high quality data has been universally identified. High quality data "can be a major business asset, a unique source of competitive advantage"[Herzog2007] while on the other side, poor quality data "can reduce customer satisfaction, [...] lower employee job satisfaction, [...] and also breed organisational mistrust and make it hard to mount efforts that lead to needed improvements" [Herzog2007]. Poor data quality also "inhibits good data-driven decision making"[vonHalle2002, Chanana2007]. In conclusion, data quality has impact on every process in an organisation. It is therefore of great importance to keep the quality of data high.

In order to gain more insight in the quality of data, the concept must first be explored. data quality is the term that is used to encapsulate multiple requirements of data used in any system. There is much literature available on data quality, and most of them have their own definition of the term. Most literature on data quality focuses on the use of data in a survey context. As surveys are mostly statistical in nature, the probabilistic checks that are proposed and described in the literature are concentrated on probability and (automated) error correction ('imputation'). These types of checks and error corrections are not part of the scope of this research and will therefore not be investigated.

2.2 Defining data quality

One definition of data quality, used by the International Association for Information and data quality[IQDS] is stated as "the degree to which information consistently meets the requirements and expectations of all knowledge workers who require it to perform their processes". This definition uses the fact that there are one or more 'knowledge workers' that have to handle the correct data in order to do their job right.

Juran et al. [Juran1980, Juran1989, Dobyns1991, Juran1999, Herzog2007] define the concept of data quality as being high when "they are "Fit for Use" in their intended operational, decision-making and other roles". This definition takes up the view of the data user or 'consumer'. Herzog et al. [Herzog2007] also incorporate another view on quality, namely that of the data 'producer', as "Conformance to Standards". This way,

agreement can be reached on the desired attributes of data quality used in the exchange between the producer(s) and the consumer(s) of data.

This last definition is adopted in this research, because it uses a set of standards to measure data quality, which is what is used in the case of the Claim Factory.

2.3 Determining data quality

The determination of the quality of data is also known as 'data quality assurance'. According to Wang et al.[Wang1995], data quality assurance "includes all those planned and systematic actions necessary to provide adequate confidence that a data product will satisfy a given set of quality requirements". This means that to be able to determine the quality of some set of data, that set must be reviewed using a given set of requirements.

As Herzog et al. [Herzog2007] state: "Consistency checks can be highly effective in detecting data quality problems within data systems". To be able to make any claim about the quality of data, the data has to be 'edited'. This term is widely used in the literature and usually references the error-detection and error-correction of survey results. As stated in section 2.1, this thesis does not focus on the (automated) correction of errors in data, just on the detection of errors.

The error-detection phase is also called 'validation'. This validation process is referred to as "a process that consists of an examination of all the data collected, in order to identify and single out all the elements that could be the result of errors or malfunctioning"[Bandini2002, Vasilecas2007]. This means that some number of checks have to be performed on the data to be able to determine its quality. These checks can be performed on single entities of data as well as collections of data.

Herzog et al.[Herzog2007] identify a number of deterministic tests that can be performed on individual fields or on sets of data. These tests include:

- *Range test:* The check is made if the value of a single data element is an element of some pre-determined set of values. This can be a deterministic set (i.e. {A, B, C, D}) or a continuous set (i.e. decimal numbers from 0 to 100). Note that the deterministic set can also consist of just one element, so the range check will effectively be just an equation.

- *If-Then(-Else) test:* This test checks if, in the case that a certain condition involving element A is true, some (other) condition involving element B must also be true.

- *Ratio Control test:* the ratio of two numerical data elements can be used as an input value for a range test described above

- *Zero Control test:* this test is usually performed for control or 'checksum' purposes. It can be performed on two or more data elements and is usually checked as 'the sum of data elements 1 through 10 has to be in range A'.

Some additional common checks are identified by McKnight et al. [McKnight2004, Herbst1994] as:

- *Null constraints:* a check to see if mandatory values are present
- *Cross checks:* this type of check is the 'parent' of the ratio- and zero control tests states above. It checks the conformance to some constraint relation between different items.
- *Type check*: this check determines if the value of a data element is of the correct type (i.e. a Date)
- *Format check:* a check that determines if a data element is present in the right format (i.e. a Date element has to be stated in 'year-month-day'-format)

When all defined checks run against a set of data and the outcome of all checks proves to be positive, that set of data is validated and the quality is assured. The degree by which the quality of data drops depends on the weight of the checks. In turn, this weight depends on a number of different variables, such as the environment the checks are placed in as well as the importance of the data being validated.

2.3.1 Validation methods

There are several moments in time when data can be validated. Herzog et al. [Herzog2007] define three scenarios:

1. *Prevent:* The recommended scenario: check incoming data for errors before it reaches a process where its quality is important.
2. *Detect:* Run periodic checks on data already in the production process and detect and possibly repair the errors that are found
3. *Repair*: Don't do anything proactively, just detect and try to repair errors as they occur in the production process. This is by far the most costly scenario.

As Herzog et al. [Herzog2007] state: "It is usually less expensive to find and correct errors early in the process than it is in the later stages". "Editing is well-suited for identifying fatal errors because the editing process can be so easily automated. Perform this activity as quickly and as expeditiously as possible." It is clear that the validation of data is best performed according to scenario 1.

2.4 Summary

There is an ever-growing need for high-quality data in today's information-driven organisations. The assurance of data quality should be a high priority for organisations who handle and make decisions based on large amounts of data. There are different ways to define what exactly makes the quality of data high. The most appropriate definition in this case is the conformation of data to some set of standards. To measure the quality of data, a method of assessing the degree of conformance to the appropriate standard(s) is used.

It has been identified that the earlier in the data handling process the quality of the data is assessed, the less costly it is. This is caused by the fact that data errors that are detected later on in the process have a larger impact on the process itself and therefore are more costly to correct.

2.5 Relation to case

2.5.1 Data Quality Assurance

Using the data quality validation checks described in section 2.3, the quality of data can be determined. As described in section 2.2, the more a piece of data (in this case, a claim message) differs from the applicable standard, the lower the quality is. The degree of quality is measured by the number of checks and their weights the data fails on, as described in section 2.3. The next step in measuring the quality of a message is defining these checks for a given message standard.

The Vektis platform offers, besides public access to the message standard definitions and documents, a testing platform called PORTES. This platform contains a set of documents designed for developers. This set contains information about the levels of control an entity that handles claim messages can perform on the messages it creates or receives.

However, this set is not complete in the sense that it covers every possible indicator of poor data quality. That's why these documents can only be used as a reference on which types of checks are identified by Vektis and not as a step-by-step guide to the requirements on the quality of data.

Vektis identifies five levels of control in PORTES[Portes]:

1. Physical file
    a) File not found
    b) File not readable
    c) Message standard specification does not exist
    d) Incorrect file format
2. File fields
    a) Record type is incorrect
    b) Record type not part of message standard
    c) Record type sequence error
    d) Record identification number not ascending
    e) Record length incorrect
    f) Opening record not in correct place
    g) Opening record not present
    h) More than one opening record
    i) Closing record not present

j) More than one closing record
k) Comment record has no corresponding detail record
l) Record which can exist 0 or 1 times in a record exist twice or more

3. Field format
   a) Numeric field has alphanumeric value
   b) Incorrect field format (Date)
   c) Mandatory field not present
   d) Value summary fields in closing record not correct
4. Field content
   a) Field value does not correspond to code list
   b) Field value does not correspond to regular expression
5. Field inter-relation. This level contains many possible checks. Because of the focus on the HA standard in this case, the following list is a selection of checks from the PORTES documents on the HA message standard made on control level 5. The complete list of the HA-checks can be found in Appendix A, the complete list for all message standards can be found on the Vektis website [Vektis]
   a) If field 0106 "Code information system software vendor" is filled in, then field 0107 "Version information system software vendor" is mandatory.
   b) If field 0113 "identification code payment to" is filled in with value 03 (= practice) then field 0111 "Practice code" is mandatory.
   c) If field 0111 "Practice code" is filled in, then field 0110 "Care provider code" is mandatory.
   d) The value of field 0115 "End date claim period" must be greater or equal to the value of field 0114 "Begin date claim period".
   e) The value of field 0205 "Patient insurance number" must be unique in the message.
   f) The value of field 0222 "Debit number" must be equal to the value of field 0303 "Debit number".
   g) If field 0223 "Indication client deceased" is filled in with value 1, then field 0326 "Relation type debit account" is mandatory.
   h) If field 9907 "Total claim amount" equals zero then field 9908 "Indication debit/credit" must be filled with value 'D'.

After analysing the case information obtained from Topicus and the documents from the PORTES and Vektis websites, the conclusion was drawn that the level 5 checks from the PORTES documents were not complete. Some additional checks had to be defined, using the check types from section 2.3. The complete list is shown in Appendix B

The checks described above can be related to the different types of checks identified in section 2.3. The table shown in table 2.1 demonstrates this by providing an example from the above list of checks for each type of check identified in section 2.3.

Check type	Vektis checks
Range test	4a: check if value exists in a (code)list
If-then/else test	5a: check that if field 1 holds a certain value, field 2 also holds a certain value
Ratio test	not used in this case
Zero Control test	Appendix B, check 17: check if the total amount of claims in a message is the same as the total claim amount field in the closing record
Null test	3c : check if mandatory fields are present
Cross test	5d: the End date treatment must be on or after Begin date treatment
Type test	3a: check if field type matches definition
Format test	3b: check if the format is properly used

Table 2.1: Vektis checks mapping

When all checks for a given message standard implemented, the quality of a message constructed using that standard can be determined. The checks of type 1, 2, 3 and 4b are currently performed during the mapping of the first transformation module described in section 1.2.2. Check type 4a is currently not performed at all (corresponding to scenario 3 in section 2.3.1) and checks of type 5 are done selectively.

3 Business Rules

This chapter provides an overview of what Business Rules are, how they are formalised and how they can be used to create flexibility in processes. The Business Rules concept is then applied to the case.

3.1 History

Since 1989, the Business Rules Group (formally known as the GUIDE Business Rules Project) has been developing the Business Rules concept. The need for Business Rules arose because system analysts had tended to neglect the explicit identification of constraints under which an organisation operates[BRG2000]. These constrains were not formally specified until the moment had come that they had to be translated into programming code. Also, the modeling techniques that were used to model processes put implicit constraints on the designed model.

If these constraints were made explicit in the design phase, the otherwise unnoticed and possibly inappropriate constraints would become known to the analysts. Another advantage of this approach was that when the process model had to be changed, the constraints would not implicitly change as well. This creates flexibility in the development process.

Knolmayer and Herbst[Knolmayer1993] also identified the importance of Business Rules in the development of Information Systems by stating that "there are strong arguments for a non-redundant, (at least logically) centralized implementation of the IS-relevant Business Rules in the database."

3.2 Definition

The Business Rules Group (BRG) defines Business Rules as "a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. The Business Rules which concern the project are atomic; that is, they cannot be broken down further"[BRG2000]. In other words, Business Rules are atomic statements that steer or limit the behaviour of some aspect of an organisation.

According to the Business Rule Group, there are two distinct perspectives on the concept of Business Rules: the business- and the information system perspective. Although these two perspectives can appear equal in most situations, there are subtle but important differences. The business perspective mostly deals rules on human activities, where

the information system perspective deals with rules on the behaviour of data. The information system perspective is preferred because it was already possible to understand and model Business Rules as constraints on data.

Although there are many different applications of Business Rules, the concept remains the same: capture knowledge of constraints in an organisation. By extracting the rules from the processes in an organisation, the rules are made explicit and visible. Moreover, when they are stored in a central place, they can be easily maintained and reused across multiple organisational units.

3.3 Classification

As the Business Rule approach is applicable in many different fields, many different classification of Business Rules exists. This section provides insight in some of the most commonly used classification methods.

3.3.1 Fundamental classification

The goal of the BRG was "to formalize an approach for identifying and articulating the rules which define the structure and control the operation of an enterprise"[BRG2000]. The research done by the BRG resulted in the definitions and descriptions of Business Rules and associated concepts. These results were used to create a conceptual model of Business Rules and how to express them. The research done by the BRG provides fundamental insight in the business rule concept.

In [BRG2000], the BRG identified three types of Business Rules:

1. Structural assertions
2. Action assertions
3. Derivations

3.3.1.1 Structural assertions

Structural assertions are defined as "a statement that something of importance to the business either exists as a concept of interest or exists in relationship to another thing of interest. It details a specific, static aspect of the business, expressing things known or how known things fit together"[BRG2000]. The definition portrays two aspects: the 'known thing of interest' (from now on called 'term') and the relationship between these terms (from now on called 'facts').

Terms can be divided into two types: *common terms* and *business terms:*

1. Common terms are well-known, unambiguous and part of the basic vocabulary. An example of a common term is 'car'. There should not be any dispute about what is meant with the term 'car'.

2. Business terms are terms which are specific for that business and its meaning is not clear by the term alone. They have to be explicitly defined in facts.
   An example of a business term is 'reservation'. It is not immediately clear what is meant by this term.

The fact 'a *reservation* is the exchange of a *car* for *cash* between a *rental office* and a *customer*, designated to take place on a certain future *date'* makes clear what is meant by the term 'reservation' and at the same time relates the terms *'reservation', 'car', 'rental office', 'customer' and 'date'* together. Facts can usually be stated in multiple ways while still having the same meaning. A fact can also relate other facts together, this is called a *compound fact*. Facts can also be classified as 'base facts' and 'derived facts'. Base facts are facts that are elementary true, derived facts are derived from other facts,

The following example clarifies the terms base fact, compound fact and derived fact:

1. Base facts are: 'a *car* has a *class*' and 'a *class* has a *base rental rate*'
2. A compound fact would be: 'a *car* has a *class* and a *base rental rate'*
3. Using the base facts stated above, the fact 'the *base rental price* of a *car* is equal to the *base rental rate* of the *class* of the *car'* can be derived

3.3.1.2 Action assertions

As described in 3.2, Business Rules steer and limit behaviour. Where structural assertions steers the behaviour, action assertions limits it. They put constraints on the results of actions. Action assertions are classified in three types by the BRG[BRG2000]:

1. Condition: a condition is applied when another specified business rule is evaluated as true.
   Example: 'if a *car* does not have a *registration number*, no *customer* can rent it'
2. Integrity constraints: an integrity constraint specifies a condition that always has to be true. This prohibits actions from happening if the result would cause the condition to be false.
   Example: 'a *car* can have one and only one *registration number*'
3. Authorization: authorizations limit certain actions from being triggered by certain 'users'.
   Example: 'only the *manager* of a *rental office* may change the *base rental rate* of a *car class'*

3.3.1.3 Derivations

Derivations are facts which are derived from other Business Rules. They can either be calculated or inferred from terms, factors, action assertions or other derivations([BRG2000]).

1. Calculated derivation: the derivation is based on some mathematical calculation
   Example: '*total rental amount* equals *base rental rate* of the *class* of the *car* multiplied by *rental days* minus *discount* for *preferred customer'*

2. Inferred derivation: the derivation is based logical induction or deduction on other Business Rules
   Example: see section 3.3.1.1

3.3.2 Other classifications

When put in an Information System or Database System context, additional distinctions between Business Rule classifications can be made.

3.3.2.1 Integrity Maintenance rules

Urban et al. [Urban1992] based their classifications on the difference between consistency (integrity constraints on valid database states) and activity (the actions or operation sequences) rules. Within the consistency rule classification, the distinction can be made between active rules (which maintain consistency by manipulating data) and passive rules (which prevent actions that may result in inconsistent database states from happening).

3.3.2.2 Service composition rules

Van Eijndhoven[Eijndhoven2008] describes additional Business Rule classifications. These classifications are usable in specific situations, such as the integration with business processes and the composition of web services. For the latter purpose, Orriens et al. identified the following classification[Orriens2003]:

- Structure rules: this type of rule is used to restrict the transition possibilities of activities in a process flow.
- Data rules: this type of rule is used to model relations between the in- and outputs of different processes and put constraints on the message flow between them.
- Constraint rules: this type of rule is used to put constraints on message integrity.
- Resource rules: this type of rule is used in the dynamic selection of service connectors in a service composition
- Exception rules: this type of rule is used to model exceptional behaviour of a web service

3.3.2.3 Business Process integration rules

For the integration of Business Rules in business processes, Van Eijndhoven compiled two similar approaches by Charfi and Mezini[Charfi2004] and Taveter and Wagner[Taveter2001] to come to the following classification:

- Integrity rules (structural assertions): this type of rule is used to guard the integrity of processes and process elements. In the case of a processes as a whole, the structural assertion acts as a process invariant. In the case of a process element, it acts as a guard condition to restrict the flow from one state to another.
- Computation rules (derivations): this type of rule is used to calculate a value of a term based on other terms or values/constants.
- Inference rules (derivations): this type of rule is used to create rules based on the knowledge of other rules and enables the capturing of the value of a rule into a variable.
- Reaction rules (action assertions): this type of rule is used to model the so-called Event-Condition-Action-rule. The basic operation is that when an event happens a condition is evaluated after which, if the evaluation return true, the action component is executed.
- Deontic assignments (action assertions): this type of rule is used to restrict access to certain process components for certain users.

3.3.2.4 Event-Condition-Action (ECA) rules

Because of their frequent use in recent studies ([Taveter2001, Charfi2004]), ECA rules will be further explained. As described in section 3.3.2, the ECA rules are of the 'reaction rules' type. This means that there must be some event that triggers this rule. This happening is defined as the Event. This can be an explicitly called event (such as: 'rule number 1 must now be executed') or some condition (such as 'the moment a *customer* arrives').

The Condition component of an ECA rule defines what property must evaluate to 'true' in order to activate the Action component of an ECA rule. ECA rules can be 'linked' by letting the Action component trigger another rule's Event-component. Herbst identified several sources that portray the need for an extended version of the ECA rules[Herbst1995]. This extension adds an 'alternative action' or Else-Then component. This specifies which Action will be performed when the Condition does not evaluate to 'true'.

ECA rules are especially applicable in modeling process flows.

3.4 Specification

Now that it is clear with which 'building blocks' Business Rules can be formulated, an overview of methods to specify the Business Rules is provided. Early works on fundamental modeling languages done by Herbst et al.[Herbst1994] provide insight in the classic ways of the specification of Business Rules. Van Eijndhoven[Eijndhoven2008] has conducted extensive research on contemporary ways to specify Business Rules. The following sections contain summaries of their work.

3.4.1 Classic specification methods

Herbst et al. provide an overview of classical modeling techniques to represent Business Rules constructs[Herbst1994]. Among the reviewed modeling techniques are Data Flow Diagrams, the Merise framework, State Transition Diagrams, Petri Nets, Entity Relationship Models, Entity Life History and Object Oriented methodologies. His work concludes that none of the techniques that were reviewed provide enough support for accurate Business Rule modeling. The most common problem is that the technique does not provide enough support for elemental rule constructs (Events, Conditions, Actions).

3.4.2 Contemporary specification methods

3.4.2.1 Near-natural languages

The near-natural language for specifying Business Rules is by far the easiest method to use and to comprehend. This allows a domain expert (who is not necessarily an IT expert) to create the rules, instead of just specifying them in natural language and letting the IT expert convert them to the used rule specification language. In the latter case, information may be lost in the translation and thus introduce errors in the information system.

Semantics of Business Vocabulary and Business Rules (SBVR) The Object Management Group (OMG) defines SBVR as [OMG2006]: "The vocabulary and rules for documenting the semantics of business vocabulary, business facts and Business Rules; as well as an XMI schema for the interchange of business vocabularies and Business Rules among organisations and between software tools". The SBVR are used to define the meaning of things, specifically the concepts of a vocabulary that accurately describes Business Rules. To allow the vocabularies and Business Rules to be specified in near-natural language, the SBVR uses a "structured English" language.

Example of a business rule stated in SBVR (from [Eijndhoven2008]):

> "It is obligatory that each rental car is owned by exactly one branch"

Attempto Controlled English Rules (AceRules) Another method to specify Business Rules in near-natural language is the AceRules system, which is based on the Attempto Controlled English language specification[Fuchs1990, Kuhn2007]. The goal of this system is "to show that controlled natural languages can be used to represent and execute formal rule systems". In order to be able to execute the rules specified in AceRules, the constructed statements need to be converted into a form of predicate logic, after which it can be converted to an executable programming language. Fundamental rule types, as described in section 3.3.1, can be directly represented using this system.

Example of a business rule stated in AceRules format (from [Eijndhoven2008]):

> "If a customer does not have a creditcard and is not provably a criminal then the customer gets a discount"

3.4.2.2 Extensible Markup Language (XML)-based languages

Rule Markup Language (RuleML) Since XML was developed, the language is widely used in systems and applications where an interchangeable data format is a key issue. Research [Wagner2002, Wagner2006]has indicated that there is a need for an interchangeable business rule specification format. One of the efforts to create an interchangeable rule format is made by the RuleML initiative. Their website states that their goal is to "develop RuleML as the canonical Web language for rules using XML markup, formal semantics, and efficient implementations"[RuleML]. Their approach is to create a rule interoperability between the various industry standards to specify Business Rules that exist today.

Example of a business rule stated in RuleML (from [Eijndhoven2008]):

<!– "Peter Miller's spending has been min 5000 euro in the previous year."
–>
<Atom>
<Rel> spending </Rel>
<Ind> Peter Miller </Ind>
<Ind> min 5000 euro </Ind>
<Ind> previous year </Ind>
</Atom>

The REWERSE Rule Markup Language (R2ML) was developed with the same goal as RuleML, but has a richer syntax. This is necessary to be able to support lossless interchange of rule specification languages. As opposed to RuleML, R2ML does support all constructs of different rule languages[Wagner2006]. As a downside, the resulting XML document becomes a lot more complicated than when stated in RuleML.

A different approach is taken by various researches which developed prototypes to record rules into XML documents, like XML/RDF[Boley2001, Wagner2004]. These XML-documents can then be translated or imported for use in other rule engines.

3.4.2.3 Rule Engine specific languages

As described in section 3.4.4.1, there are several different Business Rule engines available to execute Business Rules. Most of these engines use their own proprietary rule specification language. Others allow for rules to be stated in several languages, including open source languages such as RuleML.

The main difference between these engine specific languages lies in the types of rules they support, but also in the way rules are recorded. Some engines allow the user to import or export the rules to another format to allow for interchanging of rules. There is even an initiative that allows rule editors to specify their rules using Microsoft Excel[OpenRules] or Microsoft Word[ILOG].

Example of a business rule stated in iLOG's IntelliRule format (from [Eijndhoven2008]):

If

'*the question*' is "how_do_you_feel_today"

then

set '*the answer*' to "good";

3.4.3 Applicability

Today, the Business Rules approach is widely adopted by software vendors who provide business process modeling tools with the ability to explicitly define the constraints that are put on these business processes. Among them are Oracle's SOA Suite[SOASuite], which provides a very rich business process modeling environment as well as business rule integration. Microsoft has a similar product called Microsoft BizzTalk Server[BizTalk], which in its most recent version incorporates a Business Rules engine.

Apart from these large software vendors, some smaller vendors have also released their Business Rules capable products. Some examples are Mendix[Mendix](based on Microsoft's .NET language) and ILOG JRules (based on Java). There are also open source initiatives such as the Java Rule Engine[JavaRuleEngine] developed by Sun Microsystems.

3.4.4 System components

3.4.4.1 Business Rule Engine

The core of a Business Rule information system is the Business Rule Engine (BRE). The main task of the BRE is the execution of rules. This execution can be triggered by multiple means. The BRE is usually integrated in a Business Process Management System (BPMS), which describes the process flow of some aspect of a business. The flow usually consists of states and there exist a number of transitions between states. Business Rules in a BPMS restrict the number of transitions of one state to other states.

The BRE determines what transitions are available based on the current state and possibly external data sources. Besides the executing of rules, most BREs support the registering, classification and management of rules as well as consistency verification, as there should be no rules that contradict each other, though this functionality is often seen in Business Rule authoring tools, described in section 3.4.4.3.

3.4.4.2 Business Rule Repository

A repository is a valuable asset in a Business Rules information system. It allows the rules to be stored and accessed in a central place and therefore every information (sub)system that uses these rules use the most recent version. A repository also provides consistency across the organisation, as there can be no dispute which set of rules is to be used: there is only one. Some implementations of Business Rule repositories also enable the organisation to limit access to the stored rules by some security feature.

3.4.4.3 Business Rule authoring tools

Some vendors of Business Rule applications supply tools to author the rules used in the BRE. Depending on the complexity level of the languages supported by the BRE, these tools allow different types of users (i.e. IT personnel as well as domain experts) to create, edit and audit Business Rules. Some of these tools also incorporate rule verification methods to ensure the rules are stated correctly. An example of such an authoring tool is ILOG's IntelliRule editor[ILOG].

3.5 Summary

There are several different classifications of Business Rules. Each classification represents some form of application of the Business Rules concept, such as the composition of (web)services, the orchestration of processes and the maintenance of data integrity.

The languages to specify Business Rules are abundant. Some are very formalised, others use a more readable format. There is even a language that supports the constructs of most specification languages available. Every language imposes some constraints on the specification of Business Rules

The most important benefit of the Business Rules approach is the flexibility it creates in the authoring, execution and maintenance of process logic that would normally be 'embedded' in the model or programming code.

3.6 Relation to case

As described in the previous sections, there exist a lot of different methods to classify and specify Business Rules. This means that a choice has to be made which classification and specification method is used in this case. First, a choice has to be made on which set of classification concepts are used to model the checks mentioned above. The following list shows the classification concepts with some pros and cons:

- Fundamental classification:
  - Pros: These classification concepts define the very basis of Business Rules and are applicable in most applications of Business Rules
  - Cons: The downside of the general applicability of the fundamental concepts is that in a data quality context the different concepts lack specific constructs to model different types of checks. They will all fall under one category, namely 'Integrity constraints'. Although the fundamental classification is certainly not useless in this case, there are better alternatives.
- Integrity Maintenance
  - Pros: The Integrity Maintenance classification is centered on the concept of keeping the database on which the rules are applicable free from illegal states (a set of data that exists in the database and that does not represent a valid

state in the real world), i.e. a Patient with a name that does not equal the real name of that patient.
  - Cons: In the case of a claim message, the passive consistency rules concepts have to be used, as the claim message itself should only be *validated*, not *manipulated*. This means that the active consistency rules concepts are not used. This does not mean that the classification method is useless, it means that it is not the best fit

- Service composition
  - Pros: As the name implies, this classification is used in the composition of services. The resource rules can be used to route messages of different types (HA, ZH etc) to different rule sets. The constraint-rules can be used to model data quality rules.
  - Cons: As is the case with the classic classification method, the service composition classification is applicable to the data quality validation concept, but there are alternatives that better fit the profile.

- Business Process integration
  - Pros: The only construct in this classification that is remotely applicable to the data quality validation concept is the integrity rule, which would implicate the modeling of a process for every data quality check.
  - Cons: This classification method does not contain any construct that is directly applicable in the data quality validation concept. This means a lot of modifications have to be made to apply this classification method to the case.

- Event-Condition-Action
  - Pros: The Action-component of an ECA rule is very useful in a data quality context because different quality checks can have different consequences for the assessment of quality. Depending on the message standard the message is constructed in, different rules apply to different messages. The arrival of a message constructed using a certain standard could therefore be the trigger event for the certain set of rules to be executed. There is always a condition that has to be satisfied in order to determine the quality of the data. The Action-component can be used to model the resulting decrease in quality, as it is not always a binary case where a set of data is either of high or low quality.
  - Cons: As described in section 3.3.2.4, the ECA rules are typically applied in a chained mode. This means that although the classification method is applicable to the case study, it may be the case that the Event-component is not used as intended. Also, the Alternative-Action-component of an ECA-rule proposed by Herbst in [Herbst1995] will not be necessary as the alternative action would be to approve the message for that specific rule. This is done implicitly and therefor does not need to be specified explicitly.

In conclusion, the most appropriate Business Rule classification method to be used in this case is the Event-Condition-Action classification.

For the choice of specification method, the following reasoning is used: one of the most commonly observed burdens of Business Rule authoring is the difficulty of authoring rules in a difficult to comprehend rule language. As Business Rules are commonly applied to specific domains, not all domain experts are IT experts and vice versa. This portrays the need for a rule specification language that non-IT-personnel can use to author Business Rules.

This excludes the XML-based rule authoring methods as well as the classic specification method. The AceRules, near-English and rule-engine specific methods (such as ILOG's IntelliRule format) are possible candidates. The final choice does not depend on the criteria described above, but on the Rule Engine's compatibility.

4 Combination method

This chapter discusses the proposed method for combining the data quality checks discussed in chapter 2 with the Business Rules concept discussed in chapter 3 and also discusses how the flexibility of the data validation process is increased.

4.1 Assuring data quality using Business Rules

The combination of data quality concepts with the Business Rules approach results in a method to assess the quality of data by translating the complete set of necessary data quality checks to a set of Business Rules. Each rule represents an exception to the norm. In other words, an indicator that a certain piece of the data set does not conform to the appropriate standard.

As described in section 3.6, the ECA-rules classification method is used in the combination method. The basic operation of every rule is the Condition-Action pair. The Condition-part describes what property or properties have to be true to execute the Action-part of a rule. This means that when the Condition-element of a rule is executed and found to be true (the rule is 'fired'), the quality of the data goes down. The Action-part contains the amount by which the quality of the data decreases when that particular rule fires. The more rules are fired for a single set of data, the more the quality of that set is lowered.

In a pure binary system, each fired rule reduces the quality by 1. When at the end of the execution of some (sub)set of rules the quality reduction is higher than 0, the data is labeled as 'low quality'. In other cases, the amounts of quality reduction are added up to come to a conclusion about the degree of quality of the data. This degree can then be used to determine if the quality of the data is enough to pass the validation.

4.2 Creating flexibility in Data Quality validation processes

As described in section 1.2.4, the flexibility of a process is increased by decreasing the time that is necessary to change the process, decreasing the cost that is necessary to change the process, decreasing the number of items in the process that have to be changed and/or decreasing the number of steps that have to be taken to transform the requirements of a process to an implementation. The solution to the research problem proposed in the previous section increases the flexibility of data quality assurance processes by attacking these four dimensions of flexibility.

Table 4.1 shows the feasibility of the solution. The score of the combination method on the flexibility criterion is further discussed in chapter 6.

Flexibility dimension	Improvement
Time	The time it takes to locate and change the validation checks is lowered because the rules are stored in a central repository. This makes it easier to find the rules that have to be changed. Time is also saved due to the fact that testing the rule set is done more efficient. Errors that exists in the rule set can easily be singled out instead of having to dig through lots of code.
Cost	The cost that is incurred per changeover is lowered because now IT-people are not the only people who can write and change the rules. Less IT personnel is needed to be able to implement the changes in the rules because the rules have become much easier to read and write. Also, because the rules are stated outside the application, no IT personnel is needed to be able to locate the rules.
Ease	The ease with which the changes in the rules are made is increased because the number of items that have to be changed is decreased. This is because the rules now support an inheritance function. Rules that apply to more than one standard or more than one version of a standard can now be 'recycled', which lowers the amount of places the rule must be changed. The ease is also increased because the number of necessary steps to transform the requirements to some implementation is decreased. This is because the rules can now be stated in a language that not only IT personnel can understand. Domain experts can write the rules themselves, which causes a decrease in implementation steps.

Table 4.1: Flexibility dimensions in combination method

4.3 Other researches

Vasilecas and Lebedys[Vasilecas2007] created an approach to extract domain knowledge and accompanying Business Rules from models created in UML to provide data validation. The motivation behind this research was the lack of tools that support both system modeling and data validation. Although their findings are very interesting for future research within this case, the prototype they developed is only partly applicable to this thesis because at this stage the need for an automated rule extraction method is not present.

In [Chanana2007], Chanana and Koronios identify the need for good data quality and propose a method to use the Business Rules approach to verify the quality of data and correct poor data quality. The authors identify a number of identifiable flexibility benefits an organisation could receive when it uses the Business Rule approach in their data quality validation applications. This list is very comprehensive and is partly included in summary:

1. *Data quality:* Data can be validated when data enters the system or on data that already exists in the system. This creates flexibility in the moment the checks are

performed.

2. *Data consistency:* The rules that are used to validate data are stored in a central repository, which makes them applicable to every instance of the data set anywhere in the organisation's domain. This creates flexibility in the placement of the validation processes[1].

3. *Business agility:* organisations are under pressure to respond to the rapidly changing environment. This creates flexibility in the way the organisation interacts with the environment, thus allowing it to change more easily

4. *Control of business processes:* The control of the business process moves from the IT people, who generally don't have a lot of business knowledge, to the people who do: the business people. This creates flexibility by gaining direct control over the business knowledge instead of having to go through the IT-department to add or change knowledge.

5. *Persistence of knowledge:* the central rule repository becomes a knowledge base for the whole organisation. This creates flexibility by allowing more people to access and contribute to the repository.

6. *Ease of testing:* the centralised storing of rules allows for a greater testing effort. This creates flexibility by allowing more testing methods and testing moments in the development process of business rule applications.

7. *Fool-proof:* as Business Rules are declarative, not procedural, the catching of errors and contradiction in the rules has become very easy. This creates flexibility by allowing the rules to be stated in a non-procedural fashion, because the BRE itself determines the correct order in which the rules will be evaluated.

The listing above shows several benefits of using the Business Rules approach in a data quality validation context. Although this research is not used in any quantifiable means in this thesis, it provides more insight in the aspects of flexibility in a data validation context.

4.4 Summary

The ECA-rule concept can effectively be used in a data quality validation process to assess the quality of data. The Event-component determines which rules have to be executed for which standard(s), the Condition-component states under which condition(s) the data quality is lowered and the Action-component states by how much the quality of the data is lowered and for what reason.

The solution is feasible because it increases the flexibility of the process by decreasing the time, cost, number of changed items and number of steps between requirements and

[1] i.e. the data quality validation processes do not have to be centralised, as long as the rules that are used in these processes are stored in a central place.

implementation when a change must occur. Another research project describes seven indicators of increased flexibility.

5 Prototype

This chapter discusses the design and implementation of the prototype based on the proposed solution discussed in chapter 4.

5.1 Design

This chapter describes the design of the prototype use to demonstrate the combination of data quality validation with the Business Rules concept described in chapter 4. It shows the User Interface, the used data model in a UML class diagram, a UML sequence diagram and the UML class diagram of the application itself.

As described in the Introduction chapter of this thesis, there exist 13 different message standards. The prototype concentrates on one of these message standards, since applying the combination concept to all of the message standards is a long and difficult task and will not demonstrate the applicability and feasibility any further.

The chosen standard is the HA304-standard for General Practitioners, because a lot of messages of this type already exist in the database of the Claim Factory and a project is currently being set up to automate an administrative process which only handles messages of the HA-type. This means that the results of this thesis provide a more direct insight into the applicability of the method to a real-life case.

5.1.1 Environment

As discussed in section 1.2 the validation process should be placed before the evaluation process in the Claim Factory and after the first transformation module, as the Business Rule engine performs its execution in an Object Oriented fashion and it is the responsibility of the validation process to ensure the quality of the message for the evaluation process.

5.1.2 User Interface

The User Interface (UI) design is relatively simple: it allows a user to specify a range of messages that exist in the database to be checked using the validation process. This validation process is responsible for starting up the Rule Engine, loading the rule sets, ordering the Rule Engine to execute the request and to return the results of the request back to the UI and is described later in this chapter.

This UI is only used for testing and demonstration purposes. In an actual production environment, the validation process will most likely be implemented as a background

(web)service or an in-line process. Both do not require any user input to complete their tasks.

5.1.3 Object Model

As the messages are validated one message at a time and using the message as a whole, an Object model has to be created that represents the contents of a message. This model is different for every type of message but can have some overlap in their properties. The Object model for the HA-message is designed to represent a HA-message as it exists in the SQL server. The following classes are defined:

Bericht The representation of the HA-message; corresponds to the VecozoBaseBericht table in the database

Voorloop The representation of the Voorloop- or Opening-record within a Message object; corresponds to the VecozoVoorloopRecord table in the database

Verzekerde The representation of the Verzekerde(Patient)-record; corresponds to the VecozoVerzekerdeRecord table in the database

Debiteur The representation of the Debiteur(Debit)-record; corresponds to the VecozoDebiteurRecord table in the database

Prestatie The representation of the Prestatie(Treatment)-record; corresponds to the VecozoPrestatieRecord table in the database

Sluit The representation of the Sluit(Closing)-record; corresponds to the VecozosluitRecord table in the database

Helper This class provides some additional methods and functions that are not present in the rule specification language of ILOG

The above description results in the class diagram shown in figure 5.1[1]

5.1.4 Rules

The rules themselves are based on the ECA-rules classification. Each rule is defined with a condition that indicates if the part of the message (data element) that rule was written for does not correspond to what the applying message standard dictates about that particular part of the message. The action component consists of a rejection-operation and the addition of a *reason for rejecting* the message. This reason is added to the message object, so the return message that is eventually sent can be constructed with any reason for rejecting the message.

[1]There exists one additional record type: the Commentaar (Comment) record. This record can be attached to at most one other record, and provides additional comment. This record type is omitted from the model and thus from the prototype, as it only provides possible background information and there are no interesting constraints on the information contained in this record. Additionally, it unnecessarily complicates the model and thus the prototype.

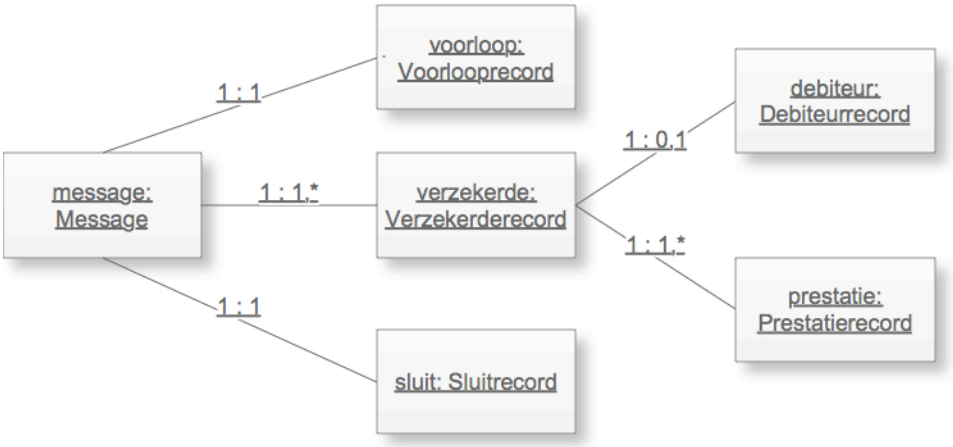

Figure 5.1: Prototype class diagram (UML class diagram notation)

The Event-part of the ECA-rule is 'extracted' from the rule, so that the individual rules are no longer responsible for deciding if they should be fired. Some other flow mechanism is used for this task. The reason for this is explained in section 5.1.5.

An abstract example using two data elements:

IF ('property of data element A does not hold')

AND ('property of data element B does not hold')

THEN {

- REJECT('message');

- REASON('both A and B do not hold');

}

5.1.5 Rule sets and -flow

The rules are all member of some rule set. These sets are defined because it creates a good overview of which rules apply to which rule set and thus to which standard. It is easier to specify one rule set for one message standard (where one rule can exist in more than one rule set) than to define which message standard the rule applies to in the Event-part of *every* rule.

These rule sets can then be divided into subsets to create flexibility in the rules that are executed at certain points in the validation process. Logic dictates that for every type of record found in the message, a subset of rules is defined. This means that, in the case of the HA message standard, there are 5 distinct subsets of rules:

1. Voorloop
2. Verzekerde

3. Debiteur

4. Prestatie

5. Sluit

Because performance is an important issue, the order in which different parts of the message are checked is vital. For example, when the rules of the Voorloop-subset are executed and the result causes the message to be denied, there is no point in executing the rest of the subsets as this will not change the result and thus use up vital resources without any additional advantages[2]. This implies the use of some execution flow to control which subsets are executed at which point in time.

In this test-setting, there exists only one rule set for one message standard. In a production setting, this 'ruleflow' can also determine which rule set should be loaded according to the type and version of the received message. As described in section 1.1.2, a HA-message can have multiple Patient(Verzekerde)-records with each Patient-record having multiple Treatment(Prestatie)-records. This causes the flow diagram to contain two iteration-loops: one to iterate over Patient-records in a Message and one to iterate over Treatment-records in a Patient-record.

When the rule flow is defined, the individual rules can be implemented. The basis for this implementation lies in the chosen combination of data quality validation methods and Business Rules classification described in section 4.1. The identified rules listed in Appendices A and B are translated to one of the rule specification languages supported by the chosen BRE.

5.1.6 System components

A number of system components are necessary to be able to validate messages using the Business Rules approach:

Data source The messages that are to be validated must be stored somewhere, preferably in a database.

Business Rule Engine As described in section 3.4.4.1, the BRE is responsible for executing a given rule set on a given set of data. It must also be able to maintain sessions to store the rule set(s) that will be executed, decide on which rules to execute according to the rule flow as well as maintaining session variables (such as results)during the execution.

[2]This is not always the case. For example when the application resides on the side of the Care chain where messages are created instead of evaluated, it might be important to uncover every possible error in the message before sending it out. In this case, every rule in the rule set should be executed, regardless of the intermediary result. Furthermore, a simple performance test shows almost no difference in execution time between messages that were approved and messages that were denied. This means that the overhead that is created by executing all subsets that apply to a message standard instead of just a few is not that big and thus does not result in a very big performance lag.

Rule Repository The repository must be able to store Business Rules in sets. It must also be integrated with the BRE, so that the BRE can request a given rule set directly from the repository when the validation application calls this function.

5.1.7 Validator class

This class represents the overall controlling class of the validation process. It is responsible for the following actions:

1. Loading the message(s) which is/are to be validated from the database and putting the records into their intended objects
2. Creating sessions with the BRE
3. Issuing the command to load the correct rule set into the BRE
4. Loading the message object(s) into the BRE
5. Setting the environment variables
6. Issuing the execution-command
7. Returning the results of the execution to the UI

5.1.8 Execution sequence

When taking all the information in this section into account, the sequence diagram shown in figure 5.2 can be constructed. This diagram shows the sequence of activities that take place when the user wants to validate a range of message using their unique identification numbers (IDs) and at which component that action takes place.

The ability to specify a range of messages is implemented so that the prototype requires minimal user input. This increases the ease with which the various tests were conducted. The iteration, shown in figure 5.2 by the return arrow with label 'next message', ensures that every message in the specified range of message ID's is validated by the BRE.

5.2 Implementation

This section explains the steps taken to build a prototype of the prototype discussed in the previous section in order to demonstrate the applicability and feasibility of applying the Business Rule approach to data quality validation processes.

5.2.1 Environment

Section 5.1.1 suggests that the prototype should be placed 'in line' of the Claim Factory project. This means that the Claim Factory would have to be broken up to be able to test the proposed method. Luckily this is not the case, as the messages themselves exist in some Object-structure in a database. This means that the prototype can exist next

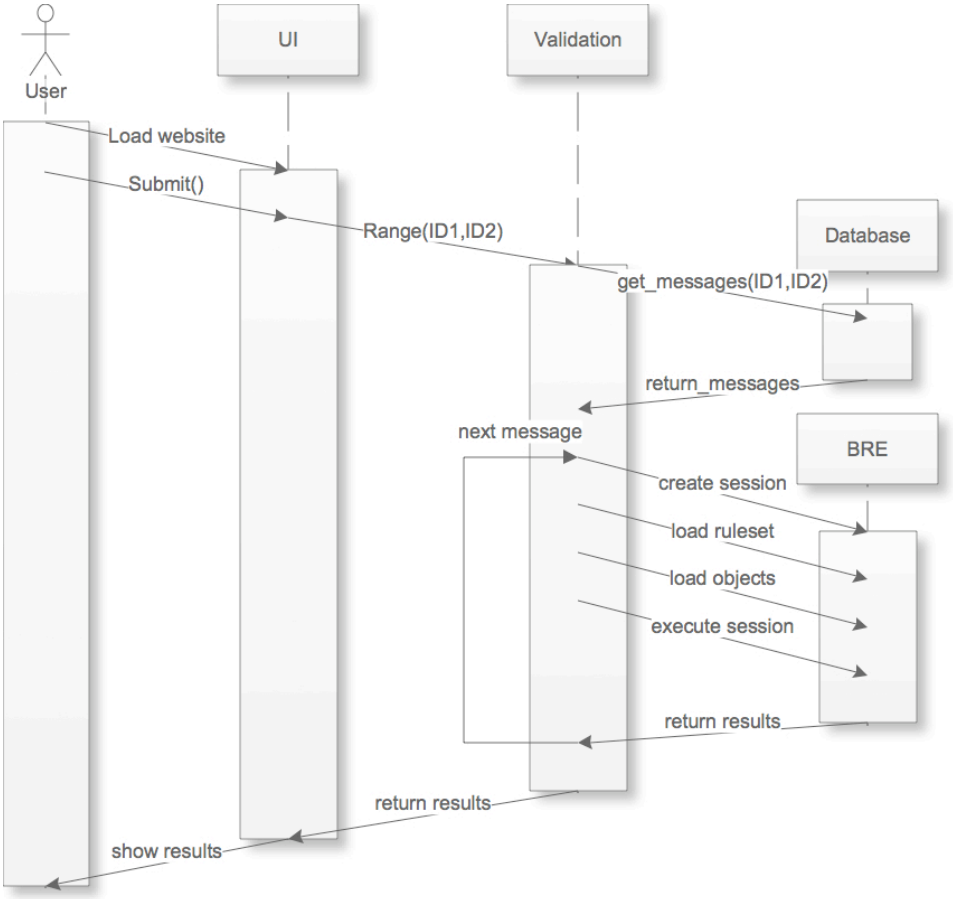

Figure 5.2: Message evaluation sequence diagram (UML sequence diagram notation)

to the Claim Factory instead of inside it. Messages that are validated can simply be 'flagged' as being validated by adding a column to the database indicating this flag. This method is preferred because in this case no changes will have to be made to the original Claim Factory processes.

The placement of the prototype within the Claim Factory is displayed in figure 5.3. It shows the added check in the second transformation module, which only allows the execution of the transformation action if the claim message is validated. Parallel to the Claim Factory itself is the validation process, which updates the status of a message when that message is validated and sends the message to the exporter module if the message is not validated.

5.2.2 System components

Data source The prototype is based on an SQL 2005 Database Server as data source. The SQL 2005 server was already available and was filled with useful test-data.

BRE and Repository ILOGs Rule Studio for .NET was chosen as Business Rule Engine and Repository. The ILOG Rule Studio for .NET was chosen because of the rule specification language, the extended amount of available documentation and the fact that it allows business users to use their favorite document editor to author the rules. Also, the .NET programming language is extensively used in applications developed by Topicus, which allows for a easier integration into new or existing projects.

The first step in the implementation process is to install and configure ILOG Rule Studio to run on a Windows machine and to be integrated with Microsoft's Visual Studio. Due to a limitation by ILOG, the Rule Studio software cannot be integrated in the newer version of Visual Studio. The installation and configuration did not cause any problems as the process is well documented.

5.2.3 Object model

The second step is to define an Object model in Visual Studio that can be used in the Rule Execution service as a data model. Each class has its own private properties which correspond to (a number of) the columns in the database tables. These properties can be retrieved by the usual Get-methods. Some classes contain additional methods, these are described later on in this section.

This introduces the third step: creating a so-called 'vocabulary'. A 'word' in the vocabulary is created for each property of each class in the Object model. These words can be used in the specification of the condition and action part of the ECA rules. The vocabulary can be edited using ILOGs Business Object Model view, a snap-in view for Visual Studio.

Not every property or method needs to be in the vocabulary, because some properties and methods only serve a purpose for object itself and do not specify any information about the state of the object which is relevant to the business user. The properties and methods that are not relevant can be hidden from the business user by specifying it

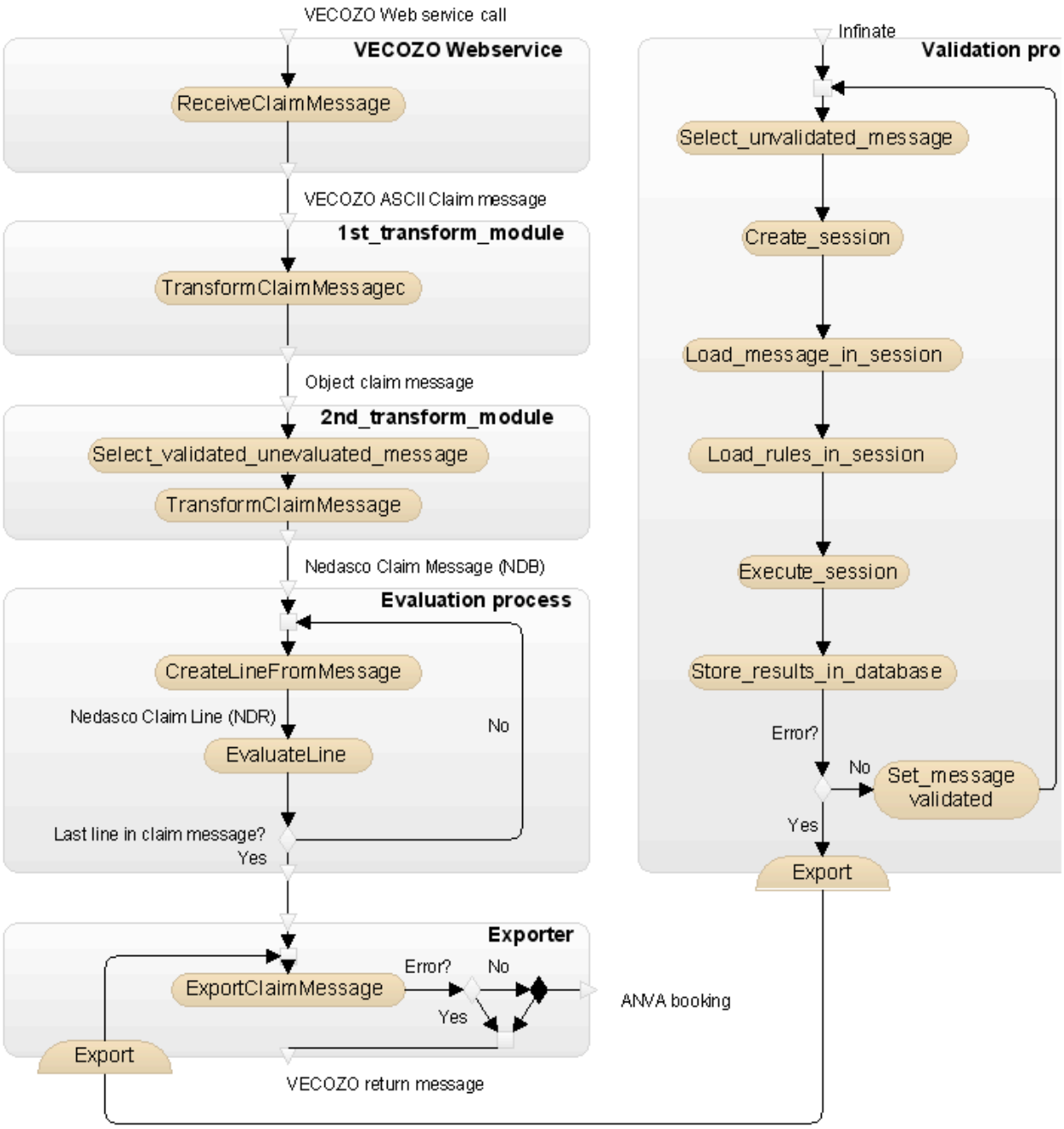

Figure 5.3: Inner processes of Claim Factory with validation process (BizzDesigner / ISDL notation)

as 'hidden' in the vocabulary. The business user can also use the vocabulary to write, edit and audit the rules outside the programming environment (for example in Microsoft Word) by using the export and import functions of the rule authoring tool.

5.2.4 Rule project

The next step is the creation of a rule project. This project is used to define the rule sets and their subsets, the rule flow and the so-called 'rule set parameters'.

5.2.4.1 Rule set parameters

These parameters define which object(s) from the Object model can be used by the Rule Execution Server as input, output or in- and output variables. For example, the Bericht-object is added as an in- and output variable, as it represents the (only) input data and is also used to provide feedback about the status of the message (i.e. whether it is approved or rejected, the reason for rejection etc).

5.2.4.2 Rules

When the vocabulary and the rule project are created, the rules can be written. This is done using the easy-to-use ILOG IntelliRule Editor. This editor uses the vocabulary in an 'intelligent' way: if the rule writer wants to add a rule which uses one variable of type *integer*, the editor will only display entries in the vocabulary which are of the same type.

When this function is used, the error rate in the authoring of Business Rules is greatly improved due to the fact that no wrong choices in operations, values etc can be made. When the author decides to not to use the function or to completely switch this function off and he or she writes the name of a variable wrong, the editor will display a red line under the variable the same way most modern text editors would do to indicate that variable was incorrectly used. When all rules are written, they are inserted in the repository under the name 'VecozoRules'.

Figures 5.4 through 5.7 show rule examples from the implementation. These rules are created using the IntelliRule rule editor from ILOG. The checks that are stated in the IntelliRule format are all derived from the data quality determination checks stated in section 2.3.

Figure 5.4 shows an example for the Range-test check

Figure 5.5 shows an example for the If-then-test check

Figure 5.6 shows an example for the Cross check

Figure 5.7 shows an example for the Zero control-test check

The rules that apply to a specific record in a claim message are grouped inside a rule subset. Rules that use the vocabulary from multiple records are placed in the subset of the first condition element of the rule. Rules that apply to the message as a whole are either placed in the Voorloop or in the Sluit subset.

```
VecozoRules.Record.Voorloop.BETALING_AAN
if
    it is not true that the BETALING AAN of voorlooprecord of 'the
current message' exists in CODE_BETALING_AAN of 'the current helper'
then
    Reject 'the current message' ;
    add "Voorloop: " + the database ID of voorlooprecord of 'the current
message' + ": CODE_BETALING_AAN not valid" to 'the current message' ;
```

Figure 5.4: 'Payment to' Range check rule (IntelliRule format)

```
VecozoRules.Record.Voorloop.IS_CODE_VERSIE
if
    the CODE IS of voorlooprecord of 'the current message' is not empty
using 'the current helper'
and the CODE IS of voorlooprecord of 'the current message' is not only
zeroes using 'the current helper'
and the VERSIE IS of voorlooprecord of 'the current message' is empty
then
    Reject 'the current message' ;
    add "Voorloop: " + the database ID of voorlooprecord of 'the current
message' + ": VERSION INFORMATION SYSTEM not present" to 'the current
message' ;
```

Figure 5.5: 'Information System code' If-then rule (IntelliRule format)

```
VecozoRules.Record.Voorloop.BEGIN_EIND_DATUM
if
    the EIND DECL PERIODE of voorlooprecord of 'the current message' is
before the BEGIN DECL PERIODE of voorlooprecord of 'the current message'
or the BEGIN DECL PERIODE of voorlooprecord of 'the current message' is
after the current date using 'the current helper'
then
    add "Voorloop: " + the database ID of voorlooprecord of 'the current
message' + ": End date claim period is before begin date claim period or
begin date claim period is after current date" to 'the current message' ;
    Reject 'the current message' ;
```

Figure 5.6: 'Begin-End-date claim period' Cross-check rule (IntelliRule format)

```
VecozoRules.Record.Sluit.TOTAAL_BEDRAG
if
    the totaalbedrag of 'the current helper' does not equal the TOTAAL
DECLARATIE BEDRAG of sluitrecord of 'the current message'
then
    Reject 'the current message' ;
    add "Sluit: " + the database ID of sluitrecord of 'the current
message' + " : The total amount of the claim present in the claim does
not correspond to the calculated amount" to 'the current message' ;
else
    add " Sluit: sluitrecord total amount is" + the TOTAAL DECLARATIE
BEDRAG of sluitrecord of 'the current message' + ", calculated amount is
" + the totaalbedrag of 'the current helper' to the debug-messages of
'the current message' ;
```

Figure 5.7: 'Total amount' Zero-control rule (IntelliRule format)

5.2.4.3 Rule Flow

Defining the rule flow is the next task. ILOG Rule Studio provides an easy-to-use flow editor as another snap-in view for Visual Studio. Every rule flow has its own start- and endpoints and a number of 'Rule tasks' and 'Action tasks' between them, connected by directed arrows modeling transitions between the tasks. Each Rule task executes a subset of the rule set. An Action-task can only perform actions, it does not evaluate any rules. Guided by the flow diagram in figure 5.2, the resulting rule flow is created and displayed in figure 5.8.

When the rule project is completed, a UI is created. This UI is kept simple: it contains two drop-down boxes which contain the unique database identification numbers of the VecozoBaseBericht-table, and only the numbers of those messages that represent a HA-message. Beneath that, a Submit-button is created. The UI also contains a panel in which a table row is drawn displaying each message's result, the number of fired rules, the Rule Server execution time and, in the case of a rejected message, the reason for the rejection.

5.2.5 Validator class

The last step is to define a Validator class which is called upon when the Submit-button of the UI is pressed. The object of this Validator class iterates over the range of database Message IDs, for each ID doing the following:

1. Retrieve the information of the message from the database
2. Create a Message-object using the information on the message from the database
3. Create a Helper-object with a reference to the Message-object
4. Create a Voorlooprecord-object using the information on the Voorlooprecord from the database

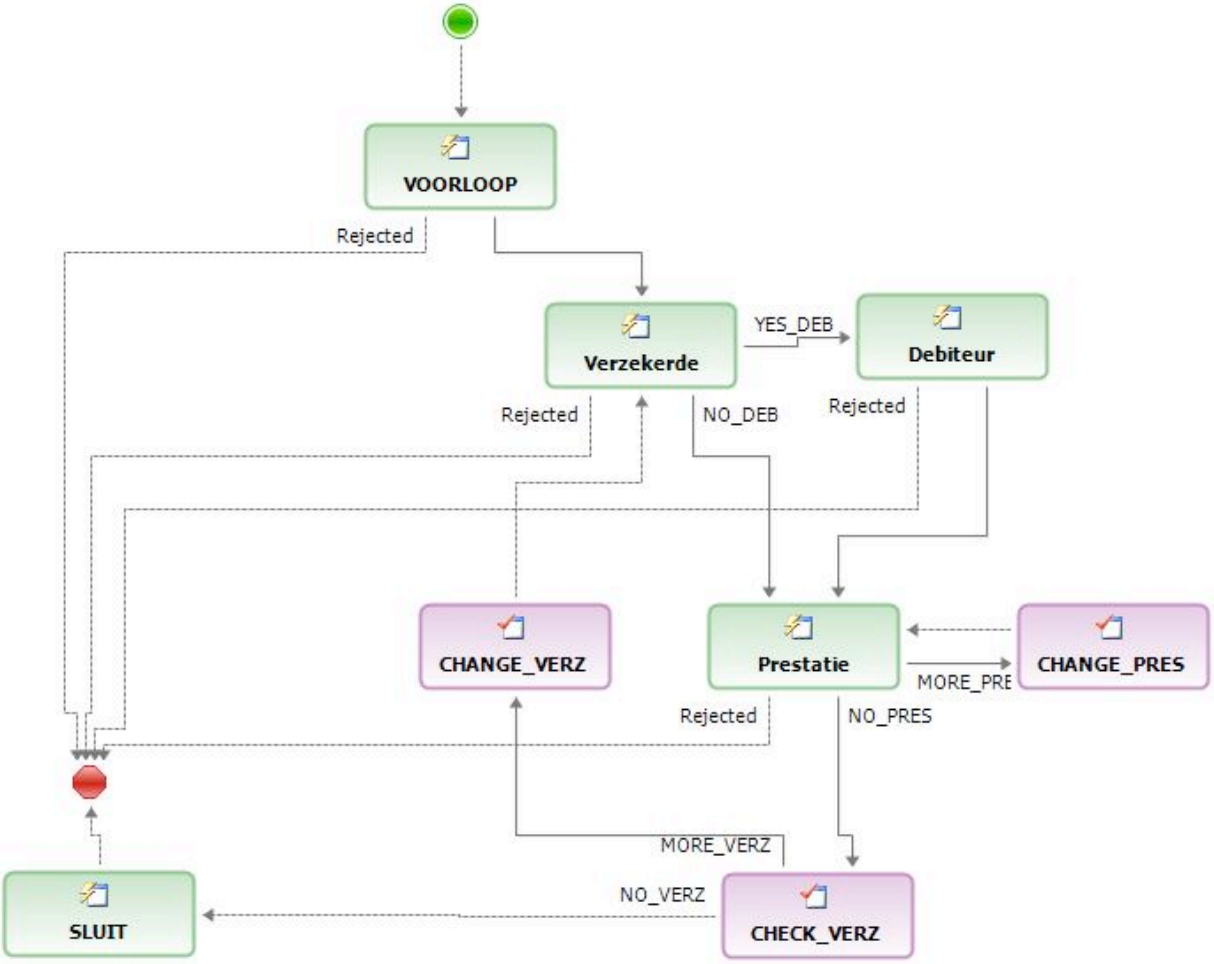

Figure 5.8: HA-message Rule flow (ILOGs RuleFlow editor notation)

5. For each Verzekerderecord in the message, do the following:
    a) create a Verzekerde-object using the information on that patient from the database and create a reference to it by adding the object to the Verzekerde-list in the Message-object
    b) check for a Debiteurrecord. If present, create a Debiteur-object and create a reference to it from the Verzekerde-object
    c) for each Prestatierecord attached to this patient record present in the database, create a Prestatie-object and create a reference to this by adding it to the Prestatie-list in the current Verzekerde-object
6. Create a Sluitrecord-object using the information on the Sluitrecord from the database
7. Create a reference to the current Verzekerderecord and the current Prestatierecord (used by the iteration process) by calling the 'initiate'-method of the Message-object.
8. Create a new session by calling the createSession()-method of the ExecutionEngine-object (which represents the ILOG Rule Execution Server service running in the background)
9. Load the appropriate rule set in the session
10. Load the rule set parameters 'the current message' and 'the current helper', pointing to the Message and the Helper-object respectively, using the session.SetParameterValue-method.
11. Start the execution time measurement
12. Execute the session
13. Stop the execution time measurement
14. Return the results

When the evaluation is completed for every ID in the range, the results are displayed in the table of the UI. An example of the UI displaying some results of the execution is shown in Appendix C. The C# code which is used to represent the Validator-class in the prototype is included in Appendix D.

5.3 Feasibility

The prototype demonstrates the feasibility of the solution by improving the three indicators of flexibility. The Time and Cost-indicators are both improved due to the fact that the rules are now stated in a language that not only specific IT personnel can understand. Also, the fact that the rules are stored in a single place where individual rules can easily

be identified improves the flexibility because in the current situation's case the checks are scattered throughout the application and will therefor take more time to be identified and changed

The Ease-indicator is improved because the number of items that have to be changed is lowered. Checks that are the same for every version of one message standard or for multiple standards (such as the check to validate if the correct record identifier is used) is only implemented once and therefore can easily be changed in the event a change in the message standard is made. In the current situation, the checks are explicitly stated for each version of the standard and for each standard.

The prototype only demonstrates that in a practical case the solution is feasible. The extent of the feasibility must be validated in further research. Chapter 6 discusses this issue in detail.

5.4 Summary

This section discusses the design and implementation of the solution proposed in chapter 4. It states that the solution can be placed in parallel with the current implementation of the validation process. The Object model that is to be used equals that of the records in a claim message. Rules are constructed using the properties of these objects. These rules are stored in a central repository and are loaded into the BRE by the Validator object. This object is also responsible for executing the validation-session created in the BRE and passing the results of the execution to the User Interface. The prototype demonstrates that the application of the Business Rules approach to validate electronic claim messages is feasible in the sense that it improves the flexibility of the validation process of the Claim Factory.

6 Validation proposition

This chapter discusses how an evaluation of the prototype proposed in chapter 5 can be performed. Due to limitations in time, this evaluation was not performed during this research project. The evaluation is therefor mentioned as an open issue in section 7.3.

6.1 Criteria

To be able to verify the validity of the solution proposed in chapters 4 and 5, a number of criteria must be reviewed:

1. The decrease of the average time that is spent per message in the evaluation process
2. The decrease in number of false positives and false negatives that occur in the evaluation process
3. The increase of the flexibility of the evaluation process

The first two criteria are conditions the prototype must satisfy in order to be feasible. The third criterion is a measure of performance, as the goal of this research was to increase the flexibility of a data validation process.

For the first criterion this means that the average time a message spends in the evaluation process must be equal or lower than in the current situation. The time increases because the validation step of the evaluation process is done more elaborate and thorough in the proposed solution than in the current situation. The time decreases because less 'bad' messages get evaluated when they should not have been. The total of the increase and decrease should indicate a decrease in order for the solution to be feasible.

The second criterion is an indicator of the quality of the rule set. The proposed solution must not only ensure that 'bad' messages *do not* pass the validation step, is must also ensure that 'good' messages *do* pass the validation step. The number of false positives and false negatives should be decreased for the solution to be feasible.

The third criterion is an indicator of the performance of the proposed solution. As described in section 1.2.4, flexibility is measured by the three indicators of flexibility. This means measuring the decrease in time and cost spent on a change in both the current situation and the proposed solution, the decrease in the number of items that have to be changed as well as the decrease of necessary transformation steps from requirements to implementation. This criterion is the most important aspect of the performance of the solution proposed in this thesis. It should therefor get the largest weight in the actual validation of the prototype.

6.2 Performance.

In order to assess the performance of the solution, the score of the proposed solution on the criteria described in section 6.1 must be measured. This section discusses what needs to be done to achieve this.

6.2.1 Indicators

For the first criterion, determining the performance of the solution on this criterion means measuring the average time that is spent evaluating a message in the current situation and in the situation where the proposed solution is implemented. This can be done by taking a set of claim messages where a known number of messages are of low quality and evaluating them using both the current implementation and the proposed solution. The time that each message spends in the evaluation process is measured along with other information, such as number of errors detected in the messages. The score of the proposed solution when compared to the current implementation is determined by the difference in the average time a message spends in the claim message system.

The score of the proposed solution on the second criterion can be determined by two different methods:

1. Empirical study: create a large set of test messages with known errors and determine the ratio of erroneous messages that do get validated against the total number of messages and the ratio of correct messages that do not get validated against the total number of messages. These ratios estimate the percentage of false positives and false negatives of both the current implementation and the solution prototype.

2. Analytical proof: analyse the message standard by stating it in some mathematical model and prove the properties of the rule set for that message mathematically. This results in a mathematical proof of the validity of the prototype on the second criterion.

The score of the proposed solution on the third criterion can be determined by measuring the three aspects of flexibility in business processes as described in section 1.2.4. The measurement of these three aspects is more difficult than the measurement of the first two criteria.

Each of three aspects depends on a *change* in a business process. This change can be simulated by implementing a previous version of a message standard in both the current implementation and the proposed prototype. Using this situation as a basis, the Time, Cost and Ease-aspects of flexibility are measured when both implementations are updated to a new version of the standard.

To measure the score of the proposed solution on the first aspect of flexibility would mean setting up a representative test set of messages and modifying the current implementation of the Claim Factory to measure the message's time spent in the evaluation process. This operation takes too much time and can therefor not be performed as part of this research.

Because of the sheer size of the claim message standards there are hundreds of different errors and combinations of errors that can exist in a claim message. The construction of the set of test messages to measure the second criterion takes up more time than is available within this thesis. The increase in flexibility is already discussed in section 1.2.4, the calculation of the actual amount of increase in flexibility falls outside the timeframe for this research. That is why the actual measurement of the score of the proposed solution on the three criteria is listed as an open issues in section 7.3.

6.3 Summary

To be able to say anything about the performance of the proposed solution, the solution itself must be validated. This can be done by defining criteria to which the current and proposed situations will be scored. These criteria are time spent in evaluation process, number of false positives and false negatives and the amount of gained flexibility. These can be measured by implementing several measurement methods. The actual implementation and execution of these measurements lies beyond the available time for this thesis.

7 Conclusions and recommendations

This chapter summarises the previous chapters by providing conclusions about the research, giving recommendations about the provided solution and listing open issues concerning the use of this solution in the future.

7.1 Conclusions

In the previous chapters, the foundation was laid to answer the main research question. The question was:

> *How can Business Rules be used in a process that validates messages constructed using some standard to increase the flexibility in handling changes in this standard?*

Combining the knowledge gained in the previous chapters, the answer to the central research question can be answered as follows:

> *The flexibility of a process that validates messages constructed using some standard can be increased by implementing the various data quality validation checks with the ECA-rules Business Rule classification concept to explicitly state and verify the data quality assessment logic outside the main application.*

Data quality The first part of the research was on the concept of data quality. Data quality in the context of this case study is defined as the conformance of a set of data to a quality standard agreed upon by the producer and consumer of that data. The determination of the quality of data is done using a set of checks based on the quality standard which measures the degree of conformance of a set of data to that standard. When the quality of a message standard is to be validated, a set of quality checks must be implemented based on the requirements of the quality standard to ensure that message is constructed based on the correct standard.

Business Rules The second part of the research was on the concept of Business Rules, which are used to explicitly state business process decision logic outside the process itself. The usual components in a Business Rule system are the rule engine, the rule repository and the rule authoring tool. The rule engine is used to execute a set of rules written in the rule authoring tool and stored in and loaded from the rule repository.

Literature on Business Rules indicate that they can be classified by a number of different schemes, for example the fundamental, Integrity Maintenance, Service composition and Event-Condition-Action schemes.

The specification of Business Rules can be done in a number of different ways. There are specification methods based on natural language, such as ACE-rules and SBVR English, methods based on XML such as RuleML and R2ML and methods which are specific to the used rule engine.

Using Business Rules techniques in data quality validation By using the ECA-rules Business Rules concept to explicitly model the necessary checks to measure the quality of a message, the rules that are used to verify the quality of that message can be changed more easily and with more accuracy when the demands on that message described in the standard of the message are changed. This greatly increases the flexibility of the validation process.

Data quality validation using Business Rules in practice The prototype was implemented using specific Rule Engine software with a specific rule specification language, but the concept is generic enough to be applied using different Business Rules implementations. The concept can also be used in different production environments. For example the Business Rule data quality concept is not limited to just claim message but can also be incorporated into other data validation applications, such as in the case of mortgage acceptance processes.

The prototype shows that the incorporation of data quality validation logic using Business Rules is very easy and powerful at the same time, where changes in the validation logic can be made fast, easy and with less errors than when the logic is embedded in the programming code of the validation process . The changes can be made by non-IT personnel, which lowers the number of steps to translate the requirements to an implementation.

In the test setting, a performance of 2 message validations per second was measured, independent of the size of a claim message. This measurement was taken on a non-dedicated standard workstation machine with a normal network connection to the database server. Due to these factors, the actual production performance is likely to increase when a faster, dedicated machine is used which has a faster connection to the database.

The average time a message spends in the Claim Factory in the current situation is roughly 15 seconds. This means that when the current prototype is used in the evaluation process of the Claim Factory, the added overhead is roughly 5 percent. According to the project manager of the Claim Factory, this number is acceptable, considering the likely increase in flexibility and decrease of average time spent in the Claim Factory evaluation process.

The prototype also detected that there was a serious deviation from the standard in a large number of messages. These messages had 'double patient records', meaning one message contained two or more Patient-records that represented the same Real-world patient. The message standard does not allow this. These messages were not detected by the current implementation of the Claim Factory's message validation process.

The quality and completeness of the rule set being used to validate the messages as well as the increase in flexibility provided by the proposed solution is still to be determined

by either an empirical or analytical study.

7.2 Recommendations

- During the prototyping phase, it became evident that the responsibilities of each part of the message evaluation process are not strictly separated. Incorporating the Business Rules approach in future projects will create a proper foundation for the distribution of responsibilities on data quality between different parts of the process.
- A rule repository should be used to centrally store all rules. This will cause a decrease in overlapping work performed on modeling and implementing process logic as well as help create awareness of the existing rules and constraints throughout the whole organisation.
- It is debatable if the approach can be incorporated in current projects and those that lie in the near future before a better understanding and appreciation is created within the organisation. Before any decision is made about this matter, point 1, 2 and 3 of section 7.3 must be resolved.

7.3 Open issues

1. Validation: As stated in chapter 6, steps must be taken to validate the proposed solution. This is a very important issue, because no guarantees can be made on the validity and thus the actual increase in flexibility of the proposed solution without having determined its validity. This research has demonstrated that the application of Business Rules is feasible in a practical case, the actual increase in flexibility must be determined in a future study.
2. Change Management: When the Business Rules approach is incorporated in the organisation and its projects and a rule repository is created, keeping the repository 'clean and organised' as well as getting people to adopt this different method of process logic administration is going to be difficult. A further study into change management in the context of the Business Rule approach has to be undertaken to fully grasp the consequences of this shift in process design.
3. Performance: The current prototype shows that the rule execution server can validate roughly two messages per second. Although this sounds like a low figure, it has to be put into perspective by calculating the savings the application of the Business Rules approach will (hopefully) produce. This means that the actual impact on the throughput of the Claim Factory has to be determined before the proposed solution can be used within the Claim Factory.
   Since the prototype was developed and tested on a simple non-dedicated machine, the number of validated messages is likely to increase. The number of validated

messages per second can be increased even further when multiple rule execution servers are used. Because both the rules and the data is stored in a central place, an unlimited number of execution servers can be used to be able to comply to performance requirements.

4. Rule quality: One of the burdens of data quality validation, whether it is implemented using Business Rules or using some other method, is the quality of the validation rules. The lack of overlap and the level of conflicts between rules is easily determined, but the rigorousity and completeness of the rules are relevant to the case in which the validation rules are placed. This means that research must be undertaken to study methods to improve and guarantee the quality of a rule set.
   During this thesis, several methods were discovered to improve the quality of a rule set. These are just some basic, non-holistic empirical methods to improve rule quality, but they can have a potentially large impact and can be used as a basis for future research. The distinction is made between 'sound practices' (first method) and 'appropriate validation methods' (second method):

   - One way to create rule quality is to evaluate the quality of the rules in the repository by some auditing principle. For example, the ' 4-eyes-method', where every rule that is about to be added to or changed in the repository has to be audited by at least one other person before it is accepted, can be applied to the authoring process to further increase the quality of the rule set. Again, this method cannot give any guarantees about the quality of the rule set.
   - Another way to to create rule quality is the use of 'decision tables' or 'truth tables'. These tables enumerate every possible combination of input variables so that every possible situation is covered by a rule. This approach also forces the rule author to think about every possible combination which could lead to more insight in the way different situations are handled. Unfortunately, these tables are not applicable to every situation. They also do not provide any guarantee about combinations of the variables that are present in the truth table with variables that are not.
     A downside of the truth tables is that it is often not feasible to enumerate every combination where only a few result in a desired or undesired situation. Example: if there exist five Boolean input variables and only three combinations of these variables result in a desired situation, it is easier to specify the rule like 'if the combination is not desired' than to specify 32 rules that enumerate every combination where 29 of them have the same outcome.

Bibliography

[Bandini2002] *Knowledge Based Environmental Data Validation*, S. Bandini, D. Bogni, S. Manzoni, Proceedings of the Biennial Meeting of the International Environmental modeling and Software Society, Lugano, Vol. 3, 2002, 330-335

[BizTalk] *Microsoft's BizTalk website*, http://www.microsoft.com/biztalk/

[Boley2001] *Design rationale for ruleML: A markup language for semantic web rules*, Boley, H., Tabet, S., and Wagner, G. (2001). In SWWS, pages 381–401.

[BRG2000] *Defining Business Rules ~ What Are They Really?*, Business Rule Group, 2000, http://www.businessrulesgroup.org/first_paper/

[Chanana2007] *Data Quality through Business Rules*, Vivek Chanana, Andy Koronios, School of Computer and Information Science, University of Australia, in: International Conference on Information and Communication Technology, March 2007, Bangladesh.

[Charfi2004] *Hybrid web service composition: business processes meet business rules*, Charfi, A. and Mezini, M. (2004b). In ICSOC '04: Pro- ceedings of the 2nd international conference on Service oriented computing, pages 30–38, New York, NY, USA. ACM Press.

[Dobyns1991] *Quality or Else: The Revolution in World Business*, L. Dobyns, C. Crawford-Mason, 1991, Houghton Mifflin

[Egvedi2006] *Standard compliant, but incompatible?!*, Tineke M. Egyedi, Computer Standards and Interfaces 29, 2007, pages 605-613

[Eijndhoven2008] *Increasing flexibility by combining business processes with business rules,* T.E. van Eijndhoven, January 6, 2008

[Fuchs1990] *Attempto controlled english - not just another logic specification language*, N.E. Fuchs, U. Schwertel and R. Schwitter, 1990, In LOPSTR '98: Proceedings of the 8th International Workshop on Logic Programming Synthesis and Transformation, pages 1–20, London, UK. Springer-Verlag.

[Herbst1995] *Business Rules in systems analysis: a meta-model and repository system*, H. Herbst, Institute of Information Systems, Research Unit 'Information Engineering', University of Bern, Switzerland. In Information Systems Vol. 21, No.2, pp. 141-166, 1996, Elsevier Science

[Herbst1994] *The Specification Of Business Rules: A Comparison Of Selected Methodologies*, H. Herbst, G. Knolmayer. Methods and Associated Tools for the Information Systems Life Cycle, Maastricht, The Netherlands, 1994, 29-46.

[Herzog2007] *Data Quality and Record Linkage Techniques*, Herzog, Scheuren, Winkler, 2007, ISBN: 978-0-387-69502-0 0-387-69505-2 Copyright 2007 ISBN 978-0-387-69502-0

[ILOG] ILOG website, http://www.ilog.com

[IQDS] *Information Quality / Data Quality Glossery*, International Association for Information and Data Quality, http://iaidq.org/main/glossary.shtml

[JavaRuleEngine] *Java Rule Engine website*, http://java.sun.com/developer/technicalArticles/J2SE/JavaRu

[JRules] *ILOG JRules website*, http://www.ilog.com/products/jrules/index.cfm

[Juran1999] *Juran's Quality Handbook (5th Edition),* 1999, ISBN: 978-0-07-034003-9

[Juran1989] *Juran on Leadership for Quality: An Executive Handbook*, J.M. Juran, The Free Press, 1989

[Juran1980] *Quality Planning and Analysis, 2d ed.,* J.M. Juran and F.M. Gryna, McGraw-Hill, 1980.

[Kasi2005] Design attributes and per- formance outcomes: A framework for comparing business processes, V. Kasi and X. Tang, In Proceedings of the Eighth Annual Conference of the Southern Association of Infor- mation Systems (SAIS), pages 226 – 232, (2005). Savannah, Georgia, USA.

[Knolmayer1993] *Business Rules*, Knolmayer, G., Herbst, H., in: Wirtschaftsinformatik 35 (1993) 4, pp. 386 - 390.

[Kuhn2007] *AceRules: Executing Rules in Controlled Natural Language*, T. Kuhn, 2007, in Proceedings of First International Conference on Web Reasoning and Rule Systems, Innsbruck, Austria (7th–8th June 2007), LNCS.

[McKnight2004] *Overall Approach to Data Quality ROI,* W. McKnight. White Paper, Firstlogic Inc., 2004. URL: http://www.oracle.com/technology/products/warehous e/pdf/Overall%20, Approach%20to%20Data%20

[Mendix] *Mendix website,* http://www.mendix.com/

[OMG2006] *Semantics of business vocabulary and business rules (sbvr)*, Object modeling Group specification, http://www.omg.org/cgi-bin/doc?dtc/ 2006-08-05

[OpenRules] *OpenRules website*, http://www.openrules.com/

[Orriens2003] *A framework for business rule driven service composition*, Orriens, B., Yang, J., and Papazoglou, M. P. (2003). In TES, pages 14– 27.

[Portes] *PORTES - Instructie*, http://ei.vektis.nl/portes/Help/instructie.html, November 2008

[RuleML] *RuleML inituative website*, http://www.ruleml.org

[SOASuite] *Oracle's SOA Suite website*, http://www.oracle.com/technology/tech/soa/index.html

[Taveter2001] *Agent-oriented enterprise modeling based on business rules*, Taveter, K. and Wagner, G. (2001). In ER '01: Proceedings of the 20th International Conference on Conceptual modeling, pages 527–540, London, UK. Springer-Verlag.

[Urban1992] *The Implementation and Evaluation of Integrity Maintenance Rules in an Object-Oriented Database*, Urban, S.D., Karadimce, A.P., Nannapaneni, R.B.,in: IEEE Computer Society (ed.), Proceedings Eighth International Conference oComputer Society Press, Los Alamitos 1992, pp. 565 - 572.

[Vasilecas2007] *Application of business rules for data validation,* Vasilecas and Lebedys, Information Technology and Control, 2007, Vol.36, No.3, ISSN 1392-124X

[VECOZO] *VECOZO website*, http://www.vecozo.nl

[Vektis] *Vektis 'Externe Integratie' website*, http://ei.vektis.nl

[Verschuren] *Het ontwerpen van een onderzoek*, Verschuren, P. and Doorewaard, H. (2005). LEMMA BV, Utrecht, The Nether- lands, third edition.

[vonHalle2002] *Business Rules Applied: Buidling better systems using Business Rules Approach*, B. von Halle, John Wiley & Sons inc, New York, 2002.

[Wagner2002] *How to design a general rule markup language*, G. Wagner, 2002, In XSW, pages 19–37.

[Wagner2004] *The abstract syntax of ruleml - towards a general web rule language framework*, Wagner, G., Antoniou, G., Tabet, S., and Boley, H. (2004). In WI '04: Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence, pages 628–631, Washington, DC, USA. IEEE Computer Society.

[Wagner2006] *A usable interchange format for rich syntax rules integrating ocl, ruleml and swrl* , Wagner, G., Giurca, A., and Lukichev, S. (2006). In Hitzler, P., Wache, H., and Eiter, T., RoW2006 Reasoning on the Web Workshop at WWW2006.

[Wand1996] *Anchoring Data Quality dimensions in ontological foundations*, Yair Wand and Richard Y. Wang, Communications of the ACM, November 1996, Vol.39, No.11

[Wang1995] *A framework for analysis of data quality research,* R.Y. Wang, V.C. Storey and C.P. Firth, IEEE transactions on Knowledge Data Engineering 7, 1995, pages 623-640

[Wang1996] *Beyond Accuracy: What Data Quality Means to Data Consumers,* R.Y. Wang, D.M. Strong, 1996, Journal of Management Information Systems I, Vol. 12, No. 4, pages 5-34

Part I

Appendices

Appendix A

This appendix contains a list of HA specific level 5 checks, taken from [Portes]. This list is defined by Vektis and states which rules a HA304-message should comply to.

1. If field 0106 "Code informationsystem softwarevendor" is filled in, then field 0107 "Version informationsystem softwarevendor" is mandatory.
2. If field 0106 "Code informationsystem softwarevendor" is not filled in, then field 0107 "Version informationsystem softwarevendor" is not allowed.
3. If field 0111 "Practicecode" is filled in, then field 0110 "Care provider code" is mandatory.
4. If field 0112 "Institutionscode" is filled in then field 0110 "Care provider code" is not allowed.
5. If field 0112 "Institutioncode" is filled in then field 0111 "Practicecode" is not allowed.
6. If field 0113 "Identificationcode payment to" is filled in with value 01 (= service-bureau) then field 0109 "Code servicebureau" is mandatory.
7. If field 0113 "identificationcode payment to" is filled in with value 02 (= 'care provider') then field 0110 "Care provider code" is mandatory.
8. If field 0113 "identificationcode payment to" is filled in with value 03 (= practice) then field 0111 "Practicecode" is mandatory.
9. If field 0113 "identificationcode payment to" is filled in with value 04 (= institution) then field 0112 "Institutioncode" is mandatory.
10. The value of field 0115 "Enddate claim period" must be greater or equal to the value of field 0114 "Begindate claim period".
11. The value of field 0205 "Patient insurance number" must be unique in the message.
12. If field 0212 "Namecode/use of name (02)" is filled in with value 2, then field 0213 "Name patient (02)" is mandatory.
13. The value of field 0222 "Debetnumber" must be equal to the value of field 0303 "Debitnumber".

14. If field 0218 "Postal code outside NL" is filled in then field 0221 "Countrycode patient" is mandatory.

15. If field 0309 "Namecode/use of name (02)" is filled in with value 2, then field 0310 "Name patient (02)" is mandatory.

16. If field 0223 "Indication client deceased" is filled in with value 1, then field 0326 "Relation type debit account" is mandatory.

17. If field 0319 "Postal code outside NL" is filled in then field 0321 "Countrycode patient" is mandatory.

18. The value of field 0403 "Social Security Number of the patient" must be equal to the value of field 0203 "Social Security Number of the patient".

19. The value of field 0404 "UZOVI-number" must be equal to the value of field 0204 "UZOVI-number".

20. The value of field 0405 "Patient insurance number" must be equal to the value of field 0205 "Patient insurance number".

21. The value of field 0409 "Enddate treatment" must be greater or equal to field 0408 "Begindate prestatie".

22. The value of field 0420 "Referencenumber this treatment record" must be unique in the message.

23. The value of field 0416 "Indication debit/credit (01)" must be equal to the value of field 0419 "Indication debit/credit (02)".

24. The value of field 0421 "Referencenumber previous related treatment record" is mandatory in the case field 0419 "Indication debit/credit (02)" is filled in with value 'C'.

25. If field 0112 "Institutionscode" is not filled in, then field 0110 "Zorgverlenerscode" and field 0111 "Praktijkcode" must both be filled.

26. If field 0112 "Institutionscode" is not filled in, then field 0110 "Zorgverlenerscode" and field 0111 "Praktijkcode" must both be filled..

27. If field 9907 "Total claim amount" equals zero then field 9908 "Indication debit/credit" must be filled with value 'D'.

28. The value of field 0415 "Calculated amount (VAT incl.) must be equal or greater then the value of field 0418 "Claim amount (VAT incl.)".

29. If the value of field 0108 "UZOVI-number" is not zero then the value of field 0407 "Forwarding allowed" must be 1 (yes).

Appendix B

This appendix contains a list of additional implemented checks for the HA304 message standard. Because it has been determined during the research that the checks defined by Vektis in [Portes] did not cover all the necessary checks, a list of additional checks is defined and implemented in the prototype. There is onegeneral rule: If the value of field xxxx should be an element of a code list, that value MUST must exists in that code list. Because a lot of fields contain values which should be elements of some code list, this check is stated as a common rule.

1. The value of field 0117 'Begindate Claim' is equal or less than the current date.
2. If field 0118 'VAT number' contains a value that is not null, the first two characters of the value must equal 'NL' and the character on position 12 must be 'B' and the rest of the positions in the string must not be characters
3. If the value of field 0203 'Social Security Number' equals '999999999' then field 0205 'patient number' must be filled.
4. No two Patient-records can exist in one message that hold the same value of field 0203 'Social Security Number' or hold the same combination of values of fields 0204 'UZOVI number' and 0205 'Patient number'.[1]
5. The value of field 0207 'Date of Birth' must be less or equal to the value of field 0117 'Begindate Claim'
6. The value of field 0209 'Namecode 01' must equal '1'
7. The value of field 0212 'Namecode 02' must equal either '0' or '2'
8. If the value of field 0303 'Debet number' is not null (a Debit-record is present) then the value of field 0222 'Debet number' must equal the value of field 0303.
9. If the value of field 0222 'Debet number' does not equal 0, the value of field 0303 'Debet number' (a Debit-record is present) must equal the value of field 0222
10. The value of field 0415 'Calculated amount must be greater of equal to the value of field 0412 'number of treatments' times the value of field 0414 'treatment tariff'

[1] this rule encapsulates rule 11 of Appendix A, as the UZOVI number is unique within a message.

11. The value of field 0421 'Referencenumber previous related treatment record' must not equal the value of field 0420 'Referencenumber this treatment record' (for any combination of treatment records in the message)

12. The value of field 9902 'Total number of patient records' must equal the number of patient records in this message

13. The value of field 9903 'Total number of debit records' must equal the number of debit records in this message

14. The value of field 9904 'Total number of treatment records' must equal the number of treatment records in this message

15. The value of field 9905 'Total number of comment records' must equal the number of comment records in this message

16. The value of field 9906 'Total number of detail records' must equal the number of patient, debit, treatment and comment records in this message

17. The value of field 9907 'Total claim amount must equal the sum of the values of field 0418 'Claim amount for every treatment record with field 0419 'Indication Debit/Credit 02' filled in with 'D' minus the sum of the values of field 0418 'Claim amount for every treatment record with field 0419 'Indication Debit/Credit 02' filled in with 'C'

Appendix C

This appendix shows a screenshot of the User Interface with the results of executing the ruleset defined in the prototype on some messages from the Claim Factory. The Status column shows the message ID and the status (Approved/Rejected) of the message. The Duration column shows the time that was spent to execute the rule set for that message. The Messages column shows the reason for the rejection (if any). The Debug Messages column shows some debug information.

Submit

302893

302712

Total duration

00:00:42.1748439

Status	Rules Fired	Duration	Messages	Debug Messages
302890 : Approved	6	00:00:00.4062786		ZA_ZV from practice : 1A Flow : Verzekerde passed Flow: Prestatie passed Flow: Check for next Prestatierecord Flow: Check for next Verzekerderecord -- Flow: Next Verzekerderecord Flow : Verzekerde passed Flow: Prestatie passed Flow: Check for next Prestatierecord Flow: Check for next Verzekerderecord Sluit: total amount is 2940 calculated total amount is 2940
302852 : Rejected	2	00:00:00.3750264	Voorloop: 192983: The message contains two Verzekerderecords which represent the same natural person (BSN-number or Patient ID number is the same). This is not allowed	ZA_ZV from practice : 1A

Appendix D

This code snippet, stated in C# from the VecozoValidationUtil class and representing the Validator-class, is included as an appendix to demonstrate the invocation of the ILOG Rule Execution server as well as to show the use of ruleset parameters in the prototype.

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using VecozoBOM.Berichten;
using VecozoBOM.Records;
// Namespace for Rule Execution Server.
using ILOG.Rules.ExecutionServer.Session;

public sealed class VecozoValidationUtil
{
    public static int validateVecozoMessage(ref HA304Bericht bericht,
  Helper helper) {
        int ruleInstanceCount = 0;

        // Get the local rule session provider.
        LocalRuleSessionProvider provider = LocalRuleSession-
      Provider.Provider;

        // Create a local rule session.
        IRuleSessionContract session = provider.CreateSession();

        // Create an execution request.
        // The path of the ruleset that is going to be executed
      is passed as a parameter.
        // Since no version is mentioned either for the ruleset
      or for the RuleApp,
```

```
            // the ruleset that will be executed actually is the high-
        est version of the
            // ruleset contained in the highest version of the RuleApp.
            // This ruleset validates the claim message. It takes the
        message and a helper object as
            // input and returns the message as output.
            ExecutionRequest request = new ExecutionRequest("/VecozoRuleApp/VecozoRules");

            // Set the values of the input ruleset parameters.
            request.SetParameterValue("the current message", bericht);
            request.SetParameterValue("the current helper", helper);

            // Set the ruleflow to be used.
            request.RuleflowFullName = "VecozoRules.VecozoRuleFlow";

            //starting time of execution:
            DateTime startTime = DateTime.Now;

            // Execute the ruleset.
            ExecutionResponse response = session.Execute(request);

            //stop time of execution
            DateTime stopTime = DateTime.Now;

            //resulting timespan:
            TimeSpan duration = stopTime - startTime;

            // Get the value of the output ruleset parameter and
        put this value in the loan parameter.
            bericht = (HA304Bericht)response.GetParameterValue("the
        current message");
            bericht.Duration = duration;

            // Get the count of fired rules.
            ruleInstanceCount = response.RuleInstanceCount;

            // Release the rule session.
            provider.ReleaseSession(session);

            // Return the fired rules value to the calling process
            return ruleInstanceCount;
        }
    }
```