University of Twente
Enschede - The Netherlands

– Master's thesis –

Evaluation of requirements management tools with support for traceability-based change impact analysis

B.J.M. Abma

September 10, 2009

Committee
Prof. dr. ir. M. Aksit
Dr. ir. K.G. van den Berg (1st supervisor)
Dr. I. Kurtev

Research group
Software Engineering
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente

Abstract

The environment in which most software systems operate changes constantly. As a result, software systems need to be adapted to be able to fulfill their purpose in the changing environment. Since the specification and management of the requirements is one of the first and therefore most important steps of each software engineering project, good requirements management is essential to cope with these changes. Requirements management tools can support managing requirements. These tools provide features to create and update requirements, specify the relations between them and use this information to perform impact analysis to support change. Four of these tools, IBM's RequisitePro, Borland's CaliberRM, TopTeam's Analyst and TeleLogic's DOORS are evaluated to determine their support for change.

Previous research on change management has been used to gather a number of criteria to test if and to what extend tools have features to support change impact analysis. Some of the properties tested for are information model support, options to import and export artifacts, integration with other tools used in the software life cycle, graphical representation of the artifacts and their relations, coverage analysis and impact analysis. A case study has been used to determine the impact analysis support of the four tools selected tools. Based on these criteria none of the tools turned out to be a clear winner; all tools had poor impact analysis features, meaning most work still had to be done by hand. The differences between the tools appeared in their supporting features for the actual impact analysis process. DOORS offered the best basis for impact analysis through its extended information model support, providing custom link types and custom attributes for links.

To quantify the performance of impact analysis methods in requirements management tools a number of methods are available. Most of these methods measure the performance of the change itself instead of the performance of the impact analysis method. We show that impact analysis methods can be seen as binary classifiers and therefore the accompanying metrics can be used to determine their performance. This method has been applied to two of the four tools using a case study with change scenarios. Again, there is no clear winner; the differences in the impact analysis methods used by both tools are in some cases an advantage and others a disadvantage.

The results of this evaluation of requirements management tools with support for traceability-based change impact analysis, shows the basics for impact analysis are present in current tools through the support for information models, different link types, custom link types, link attributes, graphical representations, etc. A lot of this information is however not used during the impact analysis itself, leading to poor results. Therefore, the impact analysis in current requirements management tools is very limited and methods that are more effective are needed.

Acknowledgements

I would like to thank everyone who has supported me during this research project.

Klaas van den Berg, Ivan Kurtev and Mehmet Aksit for guiding me through the project all this time. I would especially like to thank Klaas, his encouragement and support made it possible for me to go on and his constructive feedback made it possible to conclude this report.

The hard-working fellow students in 5066, previously known as 5070, for the productive working environment and interesting discussions. I also would like to thank everyone regularly joining the “4-uur Cup-a-Choco” sessions, for the relaxing experience at the end of each day.

Sander Bockting for refocusing my attention on binary classifiers while discussing his research results. It was of great help during the evaluation of current impact analysis performance methods.

Roderick van Domburg for providing a better view on the change management process in the real world and our discussions on the application of BC-based metrics to evaluate impact analysis techniques.

My parents Henk and Thea, for making it possible to study at Twente University in many ways. Without your support and encouragement, this would not have been possible. Thank you mom and dad, I love you.

Matthijs Abma, September 2009, Enschede

Contents

Table of figures

Table of tables

List of abbreviations

BC	Binary Classifier
CIA	Change Impact Analysis
CMS	Course Management System
FN	False Negative
FP	False Positive
IA	Impact Analysis
MOF	Meta-Object Facility
SRS	Software requirements specification
TN	True Negative
TP	True Positive
UML	Unified Modeling Language
WASP	Web Architecture for Service Platforms
XML	Extensible Markup Language

1 Introduction

``Everything changes, nothing remains without change.''

Hindu Prince Gautama Siddharta, the founder of Buddhism, 563-483 B.C.

1.1 Context

The quote at the beginning of the chapter shows that it has been known for over 2,500 years that change is universal. In our modern society this is more the case then ever before. Companies, the market in which they operate, economies, governments, they all change constantly. To cope with these changes and sustain their purpose, the software systems supporting these changing organizations must be modified too. In general these software systems change sooner then later and the earlier a in the software life cycle a change is performed, the lower the costs.

During the software life cycle, many different artifacts are created, used, modified and removed. Artifacts can be scenarios, use cases, functional requirements, architectures, designs, code or other byproducts generated during the creation of the final product. When a change is made to the system, a number of these artifacts and artifact types will be affected. O'Neil states IBM's Santa Teresa Laboratory reported that an average of 25 percent of the requirements for an average project will change before completion of the project [OC01]. Costs for these changes are even higher, Cleland-Huang shows that change can account for 40 to 90 percent of the total development costs [CHC03]. To manage these artifacts and support the change process, tools can be used. In the case of requirements, such a tools is called a requirements management tool.

A **requirements management tool** is an application to create, retrieve, update and remove requirements. This is only the most basic functionality needed. To make a requirements management more useful, additional functionality is needed. For example the option to import and export requirements, integrate with tools covering artifacts in other stages of the software life cycle and specify relations between requirements, called traceability.

Traceability is the ability to make relations between artifacts explicit. These relations can be used for example to determine what scenario is the source of a specific use case. It can also be used between different stages in the life cycle, for example to show what sections of code are responsible for the implementation of a specific step in a use case. Besides manually inspecting these traces, they can also be used by the tool to perform coverage analysis, to determine if all requirements have been fulfilled and every artifact is linked to at least one requirement to prevent unnecessary code. Another use for traces is impact analysis.

Change impact analysis identifies all artifacts that have to be changed as a direct or indirect result of a proposed change. Information obtained during impact analysis can be used to predict the number and type of artifacts affected to establish the costs of a change and to decide to perform a change or not. At the same time, it can be used to get a clear image of all effected artifacts to make sure none of the systems other functionality is affected by the change.

Requirements management tools with support for traceability and impact analysis can be helpful in an environment with constantly changing requirements, where the effects of these changes must propagate to all other artifacts in the system, without breaking it.

1.2 Problem statement

Since changes are such an important part of modern software development and maintenance, support for impact analysis appears to be an essential feature for tools used during the software life cycle. Since a systems requirements are the origin and reason for all other artifacts and therefore often less structured and harder to automatically analyze then artifacts created later on in the life cycle. To evaluate the status of current tools, a list of criteria for these tools is needed, leading to our first research question.

> *What features in traceability based requirements management tools are needed to perform change impact analysis?*

The features for impact analysis in tools give a good idea about what options are available, but they do not show the tools actual impact analysis performance. To measure a tools impact analysis performance its method must be quantified in an objective way. This leads to our second research question.

> *How can the performance of change impact analysis support in current tools be quantified?*

Finally, these criteria and metrics to determine an impact analysis methods performance have to be applied to requirements management tools to evaluate their status, leading to our final question.

> *What is the status of impact analysis in current tools, based on these questions?*

The rest of this document will be used to find the answers to these questions.

1.3 Approach

To answer the questions defined in the previous section, first current literature is inspected to see what research results are already available. Based on the results a set of criteria of metrics is composed to evaluate our selection of tools representing current requirements management tools.

To inspect the features for impact analysis in current requirements management tools, literature on the requirements for these tools is used. Several studies on the requirements of requirements management tools have been performed, using cases from the industry. The results could provide a good foundation for a set of criteria to evaluate our tools.

The same approach is used for the quantification of the impact analysis methods performance. Current literature on the evaluation of impact analysis methods will be inspected to see if it is sufficient to evaluate our tools. If this is not the case, a new method will be developed.

Once the lists of criteria and metrics are composed, they will be applied to our selection of tools and the results will be discussed.

1.4 Contributions

By answering the research questions, a number of contributions have been made. First of all a list of criteria to evaluate the features for impact analysis available in requirements management tools is presented. Next to the actual impact analysis support, most of the criteria are related to features needed to make it possible to perform impact analysis in the first place.

Besides investigating the features for impact analysis also a set of metrics has been presented to quantify an impact analysis methods performance. These metrics make it possible to show the relative number of misses and false alarms affecting the quality of the impact analysis. These numbers can be used determine what change scenarios a tool support well or does not support at all.

Applying these criteria and metrics to a selection of requirements management tools showed that the basics needed for impact analysis are presented in some of the tools, but the actual impact analysis feature is really limited in all tools, still needing a lot of manual work. As a result, the performance of the impact analysis methods was also not perfect. Even for a simple example case, each tool missed or falsely suspected at least one artifact during each change scenario. None of the tools performed better then the others, since their different properties were an advantage during some of the changes, but a drawback for the rest of the changes.

1.5 Outline

The report has been divided into two parts as shown in Figure 1. Part A will cover the features to support impact analysis present in current requirements management tools and part B will cover the quantification of impact analysis performance.

The tools evaluated in this report are introduced in Chapter 2. Besides the tools used, also a short summary is given of some of the other interesting tools that could not be evaluated in this report. Chapter 2 will discuss the basic concepts relevant to requirements management, traceability and change impact analysis. After this introduction, a set of criteria will be presented based on needs from the industry presented in the literature. These criteria are used in Chapter 4 to evaluate the tools in our selection of requirements management tools using the requirements in the WASP case.

Chapter 5 discusses the current methods to quantify and measure the performance of impact analysis methods. The current methods appear to have some limitations, so a new method to measure the performance of impact analysis methods is introduced. In Chapter 6 all methods introduced in the previous chapter are applied to two of the four tools. A set of sample changes based on a Course Management System is used as change scenario.

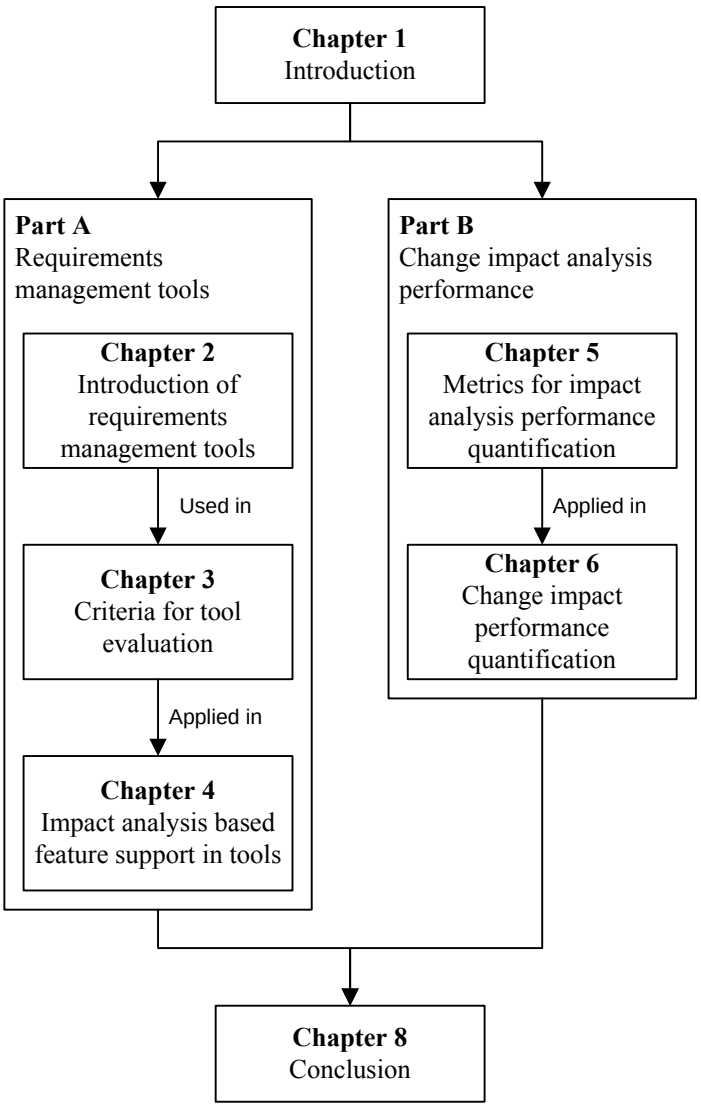

Figure 1: Document outline

Part A: Requirements management tools features

The first part of this report will cover the support of impact analysis related features in current requirements management tools.

Chapter 2 is used to present the four tools used as a representation of the current requirements management tools. In addition, a number of tools that did not make the selection, but still have some interesting features, are discussed.

Chapter 2 will cover the basic concepts related of requirements management, traceability and impact analysis so it is clear what each of them means in the context of this thesis. Once the definitions are clear, a set of criteria is specified to test what features for the support of impact analysis are present in the tools. These criteria will cover options like the specification of information models, import and export, tool integration, options to visualize requirements and their relations and of course the options for the impact analysis itself. Since not all these criteria are as relevant for impact analysis, a ranking is used to determine the weight of each criterion. The requirements document for the WASP case will be used to test if and to what extend the features covered by the criteria are covered.

In Chapter 4 all criteria are applied to the four tools selected in Chapter 2: IBM Rational RequisitePro, Borland CaliberRM, TopTeam Analyst and TeleLogic DOORS. In the end, none of the tools is a clear winner when it comes to impact analysis. The basic features to support impact analysis are there, like support for an information model, custom link types, link attributes, etc, but actual impact analysis is often limited to simply following the direct incoming or outgoing links or performing transitive closure. When looking at these supporting features, DOORS appears to be the winner, as it has extended support for custom trace links and link attributes.

2 Introduction of requirements management tools

``There are many ways of going forward, but only one way of standing still.''

Franklin D. Roosevelt, American 32nd US President, 1882-1945

One of the research questions is to determine the status of the options for and performance of change impact analysis in current requirements tools. A selection of four tools will represent the current requirements tools.

2.1 Selected tools

The selection of these tools is based on what tools are being used by partners of the QuadREAD project and tools mentioned in the literature. The tools being used by the partners are IBM Rational RequisitePro, Borland CaliberRM and TopTeam Analyst. The final tool selected is Telelogic DOORS, since several papers describe it as one of the more extended traceability tools available [ARN06, CHC03, DP98].

2.2 Other tools

When looking for information on requirements management many tools to support this process were discussed. Some of them were commercial and existed for years while others were just created as a proof of concept. A selection of interesting tools that did not make it into the final selection is discussed here.

Sparx Enterprise Architect
This tool is being used by one of the partners in the QuadREAD project and was therefore on our short list. The tool has support for both requirements and models and uses UML relations, like aggregation, composition and nesting, to link requirements and model elements. The models are used to create new artifacts through model transformations and this process is also used to immediately create traces between the source and target models [SS09].

It seems like there is no dedicated impact analysis method available in Architect. All options to use links and traces are manual processes using matrix views, hierarchies, reports for specific trace types and an auditing view showing all changes made by certain users.

In the end the tools was not selected since there were already enough tools used by partners in the selection and this tool was introduced quite late in the evaluation process.

HiVe (Hierarchical Verification Environment)
This tool is being developed by the Australian Department of Defense to manage requirements. Not a lot is known about this tool, since only a comparison with DOORS has been released [CMS06].

HiVe supports formal checking of requirements using a generic theorem prover called Isabelle. By using a front-end language called Isar and a default set of theorems and objects, this generic prover can be used for requirements. Since the proof tool is

interactive, user input is needed for each step of the proof. At the moment this solution is used, since fully automated proof can be hard to comprehend by humans.

The requirements are modeled in a Z-like schema calculus, which has proved to be a powerful tool for naming, structuring and reusing complex system requirements. The results of this tool seem promising, but since it is not available, it could not be included in our survey.

Microsoft Visual Studio
The latest version of Visual Studio has also support for model transformations. These transformations can be used to create and maintain traces between the various levels of models [GSC04]. Since this is a well-known and often used development platform, it would have been interesting to make it part of our survey. However when the tools were tested, the beta version had just been removed and was therefore no longer available.

3 Criteria for tool evaluation

``Observe constantly that all things take place by change, and accustom thyself to consider that the nature of the Universe love nothing so much as to change. The Universe is change.''

Marcus Aurelius, Roman emperor, 121-180 A.D.

In this chapter a list of criteria will be composed to be able to determine if and to what extend it is possible to perform change management and especially change impact analysis in requirements management tools. The criteria will be selected based on previous research on the needs of functionality in requirements management tools. To test if these criteria are met by our selection of tools, a sample case is also introduced.

3.1 Basic concepts for feature support

This section will describe the basic concepts needed for the evaluation of the features related to impact analysis support. Section 3.1.1 will handle requirements management, Section 3.1.2 will handle traceability and Section 3.1.3 will use these previous two sections for change management and impact analysis. The final section, Section 3.1.4, will discuss the models and modeling techniques used to model the features of the tools.

3.1.1 Requirements management

During the development of a software system many different documents, models, fragments of source code, test cases and executables are produced. All these elements are called artifacts. Cleland-Huang defines an **artifact** as a piece of information produced or modified as part of the software engineering process [CHC03].

One of the first type of artifacts produced during the development of a new software system, is the requirement. Pfleeger describes a **requirement** as a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose [Pfl98]. This definition is focused on the presence of features and capabilities, but does not mention any restrictions on the system. Wieringa proposed a more neutral definition: **requirements** of a system are the desired system properties needed to achieve the desired properties of the composite system. The requirements are motivated by the properties of the composite system [Wie03]. Features and limitations are however not mentioned explicitly. Robertson and Robertson define a **requirement** in their Volere template as follows: a measurable statement of intent about something that the product must do: or a property that the product must have: or a constraint on the system [RR97]. Since this definition clearly states that requirements can contain both features and restrictions, it will be used from now on.

Requirements come in different granularities, describing different levels of detail. The highest level used in this work, will be the **scenario**. Merriam-Webster describes a **scenario** as an account or synopsis of a possible course of action or events. This describes indeed the essence of the scenario as requirements type. ,To target it more towards software engineering a **scenario** describes in a simple way a possible course

of several events performed by a user, external system or other entity that must be supported by the system.

The next level will describe the events presented in the scenario in more detail. In this context these events are called use cases and Pfleeger defined them as: a **use case** describes particular functionality that a system is supposed to perform or exhibit by modeling the dialog that a user, external system, or other entity will have with the system to be developed [Pfl98]. A more compact version is given by Aizenbud-Reshef et al: a **use case** is defined as a sequence of actions that returns value back to an actor [ARN06].

The finest level of requirements we will use are the functional and non-functional requirements. Pfleeger defines a **functional requirement** as the description of an interaction between the system and its environment. A **non-functional requirement** describes a restriction on the system that limits our choices for constructing a solution to the problem [Pfl98]. A non-functional requirement is also called a constraint: **constraints** are the desired system properties imposed by the environment; these are the bounds within which the requirements must be realized [Wie03].

Besides the type and content of a requirement, the other information can be a valuable addition to a requirement too. Knethen described some of these additional attributes based on experiences from the industry. Difficulty of developing the requirement until it is validated, who finances the implementation of a requirement, various measures the requirement should satisfy when tested (absolute minimum, measured in the contract, planned measure, desired measure), priority, release in which the requirement should be implemented and status in the development cycle according to a predefined set of states are some of these attributes [Kne02].

Managing these different levels, types and attributes of requirements throughout the software life cycle is called **requirements management**. Hoffmann et al define **requirements management** as the structuring and administration of information from elicitation, derivation, analysis, coordination, versioning and tracking of requirements during the complete product lifecycle [HKW04]. A more compact definition is provided by Sommerville, but it lacks a reference to the total life cycle of the project: **requirements management** is the process of understanding and controlling changes to system requirements [Som89]. Cant et al present a definition that mentions both the management and time aspect: **requirements management** is the process of capturing, analyzing and tracking system requirements as they evolve through the system lifecycle [CMS06].

Most steps in the software life cycle have tool support to make working with the various artifacts more efficient. In the case of requirements these tools are called **requirements management tools** and we define them as tools to create, retrieve, update and remove requirements and everything related like relations, reference documents, etc. This is a very basic definition, but the features provided by the various tools differ a lot, and this definition covers them all.

3.1.2 Traceability

All artifacts in a software system are related. Each artifact type is the result of a previous more generic level and results in a more detailed artifact type. These links

between artifacts are called traceability links or traces: **traceability links** define the relationships that exist between the various artifacts of the software engineering process [CHC03].

Everything related to creating and maintaining traces is called traceability. There are however a lot of definitions describing traceability each in a different way. Gotel and Finkelstein identified four different types of traceability [GF94]:

- **Purpose-driven**, defined in terms of what it should do: "...the ability to adhere to the business position, project scope and key requirements that have been signed off".
- **Solution-driven**, defined in terms of how it should do it: "...the ability of tracing from one entity to another based on given semantic relations".
- **Information-driven**, emphasizing traceable information: "...the ability to link between functions, data, requirements and any text in the statement of requirements that refers to them".
- **Direction driven**, emphasizing traceability direction: "...the ability to follow a specific item at input of a phase of the software lifecycle to a specific item at the output of that phase".

Each definition has its own priorities and scope and none of them covers all aspects. Therefore, we decided to use the following definition, without any priorities or limiting scopes: **traceability** is the ability to make relations between artifacts explicit.

The current definition for traceability is very generic and to be able to specify more precise what scope a certain traceability operation has, additional definitions have been proposed. The direction of traceability information is called forward or backward traceability. Wieringa describes **forward traceability** as the ability to trace a requirement to components of a design or implementation and **backward traceability** as the ability to trace a requirement to its source, i.e. to a person, institution, law, argument, etc [Wie95]. Pre- and post-requirements traceability seem to describe the same scope: **pre-SRS** traceability refers to those aspects of a requirement's life prior to its inclusion in the requirement specification and **post-RS** traceability refers to those aspects of a requirement's life that result from inclusion in the requirement specification [GF94].

When only the traces between artifacts of a specific type are taken in account, it is called **inter-artifact traceability**. When traces between different types are relevant, it is called **extra-artifact traceability**. Pinheiro defined these scopes specifically for requirements: **inter-requirements traceability** refers to the relationships between requirements and **extra-requirements traceability** refers to the relationships between requirements and other artifacts [Pin03]. Pfleeger made the same distinction, but he called it horizontal and vertical traceability: for each artifact type, **vertical traceability** expresses the relationships among the artifacts of the artifact type. **Horizontal traceability** addresses the relationships of the components across collections of artifacts [Pfl98].

Besides the orientation and scope of traceability, the general type of traceability is also relevant. Both Ramesh and Lavazza make a distinction between product-related and process-related traceability [LV00, RJ01]. **Product-related** traceability links and mechanisms describe properties and relationships of design objects independent of how they were created. **Process-related** traceability links and mechanisms are being

captured by only looking at the history of actions taken in the process itself and are not recoverable from the product relationships alone. Product-related traceability is mainly used when the use of traceability information is not yet common in an organization or project group. Generalization and aggregation abstractions are common in this traceability model. When they get more experienced with traceability the process leading to artifacts becomes more important. Instead of depending on other artifacts, artifacts evolve into other artifacts according to a certain rationale. More advanced traceability users use more of these process-related traces.

Creating relations with one generic trace type results in a set of traces that will be very hard to use in all but the smallest projects. Several papers describe different types of traces to cope with this problem [ARN06, LV00, PG96, RJ01].

3.1.3 Change impact analysis

Since a majority of the software systems has to change during their operational life span, it helps a lot when they are developed with this need for change in mind. The online dictionary Merriam-Webster defines a **change** as "to make different in some particular" [MW09]. In a software engineering context a more strict definition is provided by Ecklund Jr et al: a **change** is a requirements-level expression of an intended revision [EDF96]. This shows again the importance of solid requirements management, just as the initial project, changes also start with the requirements.

To cope with change the most important aspect is the evolvability of the software system itself. Rowe et al describe a high-level definition of evolvability as "a system's ability to accept change". Based on the framework of a quality attributes model a number of change categories can be identified, like generality, adaptability, scalability and extensibility. Using these categories they provide a refined definition of **evolvability**: an attribute that bears on the ability of a system to accommodate change in its requirements throughout the system's lifespan with the least possible cost while maintaining architectural integrity [RLL98].

When the design and architecture of a software system make it hard to perform any changes at all, nothing is going to change that. However, when a system was designed with evolvability in mind, a good change process can improve the implementation of changes. Such a process is called **change management** by Bohner and Arnold and provides a common communication channel between maintenance staff, users, project managers and operations staff, and it provides a directory of changes to the system, for status reporting, project management, auditing and quality control [BA96].

According to Moreton change management needs at least the following steps: impact analysis, system release planning, change design, implementation, testing and system release and integration [Mor96]. We are interested in the first stage: impact analysis. Bohner and Arnold describe an **impact** as a part determined to be affected, and therefore worthy of inspection [BA96]. Based on this definition of an impact and the context of software engineering leads to their definition of **impact analysis**: the activity of identifying what to modify to accomplish a change, or of identifying the potential consequences of a change. Another definition is provided by Pfleeger as the evaluation of the many risks associated with the change, including estimates of effects on resources, effort, and schedule [Pfl98]. An important aspect all definitions have in

common is that impact analysis normally does not change anything but our understanding of the system and the effect of the proposed change [AB93].

Both these definitions do not only focus on the artifacts possibly affected, but also on the consequences and risks. Since we will evaluate the impact analysis methods in requirements management tools, that primarily support the identification of suspect artifacts, we define **change impact analysis** as identifying all artifacts that have to be changed as a direct or indirect result of a proposed change.

Arnold and Bohner identified two types of impact analysis being supported in tools. The first type works based on analyzing the source code using techniques like program slicing, cross-referencing and control flow analyzing. The second type is more life-cycle-document oriented and is based on, creating relations between artifacts from the beginning of the project, instead of the end. Using traceability relationships, configuration management systems and consulting designs and specifications are techniques used for this second type of impact analysis. According to Arnold and Bohner, the first type is more mature and provides a finer grained analysis of the impacts, while the second one is less mature but provides a broader analysis [AB93]. Since we focus our evaluation on requirements management tools, we will work with impact analysis methods of the second type.

Over time, Bohner has identified five different search algorithms to perform impact analysis. Each of these algorithms has been presented in at least one paper [Boh02]:

- **Semantically guided**: the analysis is performed using predetermined semantics for objects and relationships.
- **Heuristically guided**: identification is done by using predetermined rules or heuristics to suggest possible paths to inspect or impossible ones to ignore.
- **Stochastically guided**: analysis is performed using probabilities derived for the situation, using characteristics of the artifacts.
- **Hybrid guidance**: identification is done using a combination of the methods described above; examples are program slicing and ripple-effect analysis.
- **Unguided/exhaustive**: impacts are identified using a brute force solution using simple traceability information, for example transitivity closure algorithms.

Based on the same literature Bohner classified three key challenges in modern impact analysis:

- **Information source volume**: Many information sources have to be analyzed and related.
- **Change semantics**: the ways to describe software change relationships are limited for the range of software artifacts.
- **Analysis methods**: the methods to analyze the dependencies and traceability structures of software work products have not been fully explored for current products (web services and COTS components).

These three challenges will be taken in account when evaluating the impact analysis features present in the current tools.

3.1.4 Models and modeling

The traceability options of tools can be described using text, but it takes a lot of text to describe all options and limitations. Therefore, a model will be presented to depict the traceability options available in each tool.

The definition of a **model**, according to the very general description of Cambridge, is "a representation of something" [Cam09]. Dictionary.com is more specific, offering multiple definitions of which "a simplified representation of a system or phenomenon, as in the sciences or economics, with any hypotheses required to describe the system or explain the phenomenon, often mathematically" seems to fit best in this case [Dic07]. The Free On-line Dictionary of Computing is more aimed at computer science and describes a model as "a description of observed behavior, simplified by ignoring certain details. Models allow complex systems to be understood and their behavior predicted within the scope of the model, but may give incorrect descriptions and predictions for situations outside the realm of their intended use. A model may be used as the basis for simulation" [How07].

A compact definition of a model targeted at computer science is provided by Kurtev: a **model** represents a part of the reality called the object system and is expressed in a modeling language. A model provides knowledge for a certain purpose that can be interpreted in terms of the object system [Kur05]. So part of the objects existing in reality is expressed in a model using a modeling language. Just as with normal languages, like English, Dutch or German, something expressed in a language only has meaning when the reader is able to associate it with the concept of the corresponding real world object in his mind. Figure 2 shows the meaning triangle of Ogden and Richards [OR23]. It shows how a real world object and a real world symbol, expressed using a language, are related through the concept of these elements in ones mind. The same holds for models and the modeling languages used to create them. They use symbols to represent real world entities and relations and are only understood when the observer is aware of their meaning in the real world.

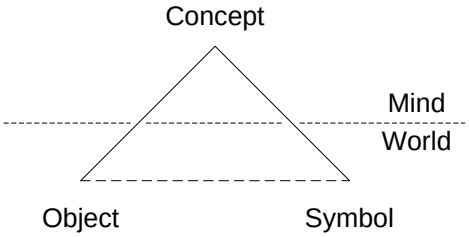

Figure 2: The meaning triangle (based on [OR23])

The requirements management tools mentioned in Section 3.1.1 contain a model expressing the actual requirements and their relations in the real world. This model is created using a modeling language and just as the requirements themselves, it is also possible to create a model to express this modeling language. The model for this modeling language is called a **metamodel** [Kur05]. As a result, each model is an instance of its metamodel.

Figure 3 shows an example of this metamodeling architecture. $Model_L$ represents a part of reality using a modeling language called L. The subscript part of the model

name indicates the language it is expressed in. It is possible to create a model of modeling language L, called a metamodel. This metamodel is expressed in a metalanguage ML and therefore called $LModel_{ML}$. Again, it is possible to create a model of meta modeling language ML, called a metametamodel. This metametamodel can be expressed in a new metametalanguage MML, this language can again be modeled and this process can be repeated an infinite number of times. In practice, three modeling levels are often enough. The amount of levels can be limited by expressing the metametamodel in the language ML itself. In this case, the metametamodel will be called $MLModel_{ML}$. Language L is a language used to model the real world and language ML is a language used to model a language model. So ML cannot only be used to model $LModel_{ML}$, but also $MLModel_{ML}$, since it is just another model. This makes the top-level a self-reflective model.

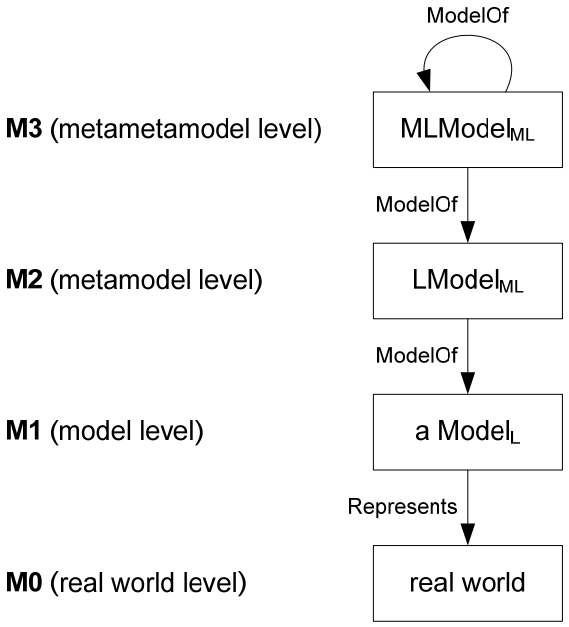

Figure 3: Metamodeling architecture (based on [Kur05])

Atkinson identified two separate dimensions for metamodeling. This leads to two different forms of instantiation. The first dimension handles the language definition and as a result uses linguistic instantiation, just as the original instantiation. The second dimension handles the domain definition and therefore used the ontological instantiation [AK03]. The **linguistic metamodel** is a model to define a modeling language that can be used for the linguistic instantiation. The **ontological metamodel** is a model to describe what concepts exist in a certain domain and what properties and relations they have and is used for the ontological instantiation. It relates user concepts to their domain and allows for the creation of types of types or domain metatypes. These two forms occur at the same time and allow us to locate a model element precisely in the language-ontology space.

The real world artifacts and their relations will be modeled in the requirements management tools. Therefore, the real world artifacts will be in the real world level or M0 and the model of these elements will be in the model level or M1, as depicted in Figure 3. If we want to create a model of the modeling features of the tools, we need a model of the language used to create the model: a metamodel. Therefore, the model for the tools modeling features will be in the metamodel level or M2. The metamodel

will be expressed in the language modeled in the metametamodel in the metametamodel level or M3. In this case, the metametamodel will be the Meta-Object Facility (MOF), the same metametamodel used to model the language used to express UML.

The standard linguistic metamodel architecture combined with the ontological metamodels applied to requirements management has been depicted in Figure 4. The various model levels contain the following models:

M3 Linguistic metametamodel for the metamodel, modeling the language used to describe the modeling features of the tool. In this case the metametamodel is the MOF.

M2 Linguistic metamodel for the model, modeling the language for modeling the concrete artifacts, the relations and the information model with their corresponding types. This metamodel will be used to describe the modeling capabilities of the various requirements management tools.

M1 O1 Model representing the general concepts of artifacts and their relations, also called artifact types and relation types in the requirements management tool. At the same time it is also the ontological metamodel of the concrete artifacts model M1 O0, describing the various types of artifacts and relations used there. This is the **information model** for a project.

M1 O0 Model representing the concrete artifacts and their relations in the requirements management tool.

M0 ‘O1’ General concepts of real world artifacts and their relations.

M0 ‘O0’ Concrete traceability project with real world artifacts and their relations.

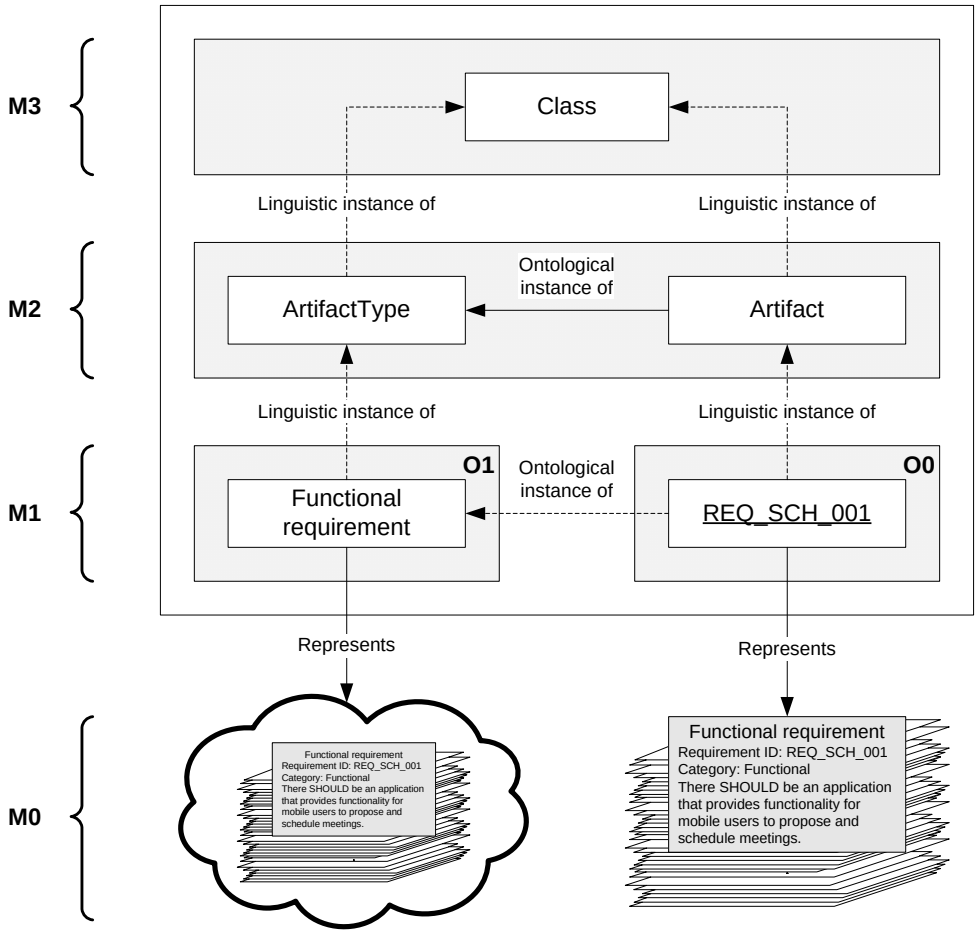

Figure 4: Models used to model features of requirements management tools

3.2 Criteria for evaluating requirements management tools

To evaluate what features are available in the requirements tools for change impact analysis a series of criteria are composed. The criteria will not only focus on the impact analysis method itself, but also on the features providing the basis to perform any impact analysis at all. There are criteria to determine the options to model an information model, to import and export artifacts, to integrate with other tools, etc. Not all criteria are as important for impact analysis, therefore each criterion gets a rating to show its relative weight compared to the others.

3.2.1 Criteria related to traceability

To determine which requirements are relevant for requirements management tools in general and traceability in particular, a number of papers on this subject have been consulted. Hoffmann composed a list of requirements for requirements tools [HKW04]. Most of the requirements are based on projects at Daimler Chrysler. So it is not supported by many cases, but in general it is a good basis for a list of requirements for traceability in these requirements management tools. Dömges and Pohl wrote a paper on adjusting traceability environments to project specific needs [DP98]. It describes a number of requirements to perform traceability, why they are needed and how they could be implemented in a tool. Their core requirements for adaptable environments are the need for project-specific data types (which data types

must be captured) and project-specific trace strategies (in which situations, how and by whom must these types be captured). Aizenbud-Refesh et al. describe some traceability support problems in current requirements management tools and give some reasons why these could (partly) solved by using model driven development [ARN06]. They point out problems with the lack of a metamodel, links to artifacts in external tools, the automated creation of links, maintenance of traceability relationships and the poor support for methodologies.

Based on the requirements and problems discussed in these publications the following list of criteria, with functional requirements sought in modern traceability environments, is created.

Criterion (1)	**Information model support** [HKW04]
Description	Tools shall allow the definition of requirements management information models.
Motivation	Requirements management information models are metadata models used in requirements management projects to formally define objects of different types, their attributes, and links between them [WW02]. This helps the engineers to understand this information and how they should systematically manage their specifications. The model is created beforehand and is independent of the tools, so the tool has to support free modeling of the requirements so the model can be used.
How is it measured	Is free modeling possible, yes or no? If it is possible, what is exactly possible? Is it only possible to create general requirements with general links between them, or is it possible to clearly specify the types of the requirements (use case, functional, non-functional)? What is the model of these possibilities?

Criterion (2)	**Links detached from artifacts** [ARN06]
Description	Information about the link should not be stored in the artifact it originates from or points to.
Motivation	If the trace information is stored in the artifact, the requirement management tool has to be able to recognize and manipulate all the artifacts involved. Therefore, it needs explicit support for all the artifacts or it has to tread them as plain text and leave the understanding of the semantics to the user. If the trace information is only in the source, it is not known at the target. However, if it is stored in both the source and the target, updating will be more cumbersome since the info is in two locations. Therefore, the link information should not be stored in the actual artifacts.
How is it measured	Is the tool able to store information in the artifacts, or has it an internal representation of the artifact that can be used by the link?

Criterion (3)	**Uniquely identifiable reference objects** [ARN06]
Description	The tools must be able to uniquely identify a reference object in space and time.
Motivation	If the trace information is not stored in the artifact itself, the artifact has to be identifiable. Even if it has been moved, merged with another artifact or changed the link must point to the right artifact.
How is it measured	Does the tool store an identifier in the reference object, yes or no? If it does, how is this identifier constructed?

Criterion (4)	**Traceability link types** [ARN06]
Description	It must be possible to use different types of traceability links, each with their own meaning.
Motivation	Not all relations between artifacts have the same meaning. A relation indicating that an artifact satisfies another artifact has a different effect on change in the requirements then a relation indicating that one artifact evolved into another one. Therefore, to give more meaning to the relations between artifacts, different types of traceability links are required.
How is it measured	Is it possible to use more then one type of trace link? How many types can be used, is it possible to add own types? How does the tool use the different trace links?

Criterion (5)	**Assign attributes to links** [HKW04]
Description	It must be possible to assign attributes to links.
Motivation	In a real world software project, not every relation between two artifacts has the same properties. Therefore, the availability of link attributes is necessary to express these characteristics in the traceability model. When the relations between the various artifacts are used for analysis or filtering later on, the link attributes can be used to query the right ones.
How is it measured	Is it possible to add attributes to links, yes or no? Is it also possible to create custom attributes for links?

Criterion (6)	**Automatic link detection** [ARN06]
Description	The tool should be able to automatically detect relations between artifacts through for example information retrieval techniques, monitoring of change history, naming schemas or model transformations.
Motivation	The increased effort that is needed to manually create and maintain traceability information causes the lack of general acceptance. Therefore, the automatic detection of relations between artifacts can help a lot.
How is it measured	Is there any way to automatically determine if new traces are required or existing ones need revisioning? What technique is used to determine these relationships?

Criterion (7)	**Enforce traceability links** [HKW04]
Description	The tool must be able to enforce the creation and change of certain traceability links.
Motivation	If the tool is able to enforce the creation or change of traces if a requirement is altered, the traceability information stays consistent. If the changes in the traceability information, because of a change in the requirements, have to be made manually, it is possible the changes are not carried out or they are carried out wrongly, causing an inconsistent trace model. If any other changes are made with this inconsistent system as a basis, the inconsistency will spread, eventually resulting in barely usable trace data.
How is it measured	Is there a way to enforce the creation or change of traces, through for example scripting? If so, in which cases can the enforcement be used and what link types can be created and or changed as a result?

Criterion (8)	**Requirements documentation support** [HKW04]
Description	The tool needs to be able to read and import requirements from requirement specification documents created outside the tool, in for example Microsoft Word. At the same time, it needs to support the creation of documents describing the requirements in the tool.
Motivation	For a development team all the necessary requirements information is stored in the requirements management tool. Therefore, in their case the need for separate requirements specification documents is not that urgent anymore. However there are still parties with no or limited access to the tool, for example the customer or other companies working on the same project. For them it is important that the tool support a way to import and export the requirements information in the tool.
How is it measured	Is it possible to import or export any requirements information from or to a document? When importing is it possible to recognize requirements, how are they detected, can link information be identified? What information can be exported, what document formats are supported, is there support for document templates, is it possible to export meta information like history or ownership?

Criterion (9)	**Tool integration** [HKW04]
Description	The tool needs to be able to interface with other tools in the product life cycle to make information stored in them visible and linkable.

Motivation	There are many tools involved in the software product life cycle, each with their own artifacts. Because these artifacts are the result of one or more requirements, they have to be represented in the requirement management tool. If this is not the case, the consistency of the system's representation in the requirements tool will be insufficient and the traceability information will deteriorate. To keep the information in the requirements management tool consistent during the various product phases and the traceability information complete, the information in the artifacts has to be visible and linkable for the tool. This can be realized by interfaces between the various tools.
How is it measured	Is it possible to link to external artifacts? What types of artifacts? Is the interface active, so the requirements management tool is aware of changes in the artifact? What is the granularity of the smallest possible structure and is there any redundant data because of the interface?

Criterion (10)	**Graphical representation traces** [HKW04]
Description	The tool must provide graphical user-friendly representation of traces.
Motivation	To determine the relations between artifacts and to analyze the interdependencies in case of a change, some form of visualization of the trace info is needed. The traces can be shown using for example matrices, trees or graphs, so the information can always be shown in an appropriate form.
How is it measured	What visualization forms to represent traces are present in the tool?

Criterion (11)	**Coverage analysis support** [HKW04]
Description	The tool must be able to produce an overview of the artifacts that have no traces to previous or subsequent artifacts.
Motivation	When artifacts appear without traces to artifacts produced earlier in the software life cycle, this can mean that unnecessary functionality is introduced. If artifacts have no traces to artifacts realizing the requirements, this can mean they are not being implemented and functionality is missing. An overview of all artifacts not being traced to previous or subsequent artifacts in the software life cycle can help to prevent this.
How is it measured	Is there a view to list all artifacts without traces to or from the neighboring stages in the software life cycle? Can all types of artifacts be shown in one view with both missing traced from and traced to links, or are several views needed?

Criterion (12)	**Impact analysis support** [HKW04]
Description	The tool must provide a way to perform an impact analysis to determine the effected artifacts.

Motivation	To support the evolvability of the system, a change strategy is needed to make sure all changes are performed in such a way that risks to the system are minimized. An analysis of the impact of the change can help to create an overview of what to modify to accomplish the proposed change or to identify potential consequences of the change. These related artifacts can be located with the help of the traceability information.
How is it measured	Is it possible to propose a change and get an overview of artifacts directly or indirectly affected? What is the granularity and extend of this information?

3.2.2 Non-related criteria

The literature mentioned in the previous section also discusses some requirements for requirements traceability tools being less relevant for the traceability support or the determination of the implicit metamodels of the selected tools.

For example, the different views specifically for requirements are less relevant for traceability, since the relevant views for traceability are already discussed. The generation of requirements documents is not that relevant for traceability either. Maybe it uses some of the traceability information to resolve the relations between requirements, but the generation itself has nothing to do with traces. Neither have collaborative working on the same task, checking out for offline use or web access to the requirements. Administrative tasks like central installation and administration; management of users, roles and rights and workflow management are of no relevance to the tracing process either, hence they will be not evaluated.

3.2.3 Rating

After the criteria have been evaluated for each tool, an overview table is presented after each evaluation with for each criterion a rating for its performance and a small outline describing important aspects present and missing. The interpretation of the rating given to each criterion is explained in Table 1. To make it possible to do a quick comparison of the three tools, a table is presented showing only the rating each tool gets for each criterion, after the individual tool evaluations.

Table 1: Interpretation of criterion ratings

Score	Interpretation
+	Important aspects of a criterion are supported.
□	Number of important aspects is available, but some are also missing.
-	Most important aspects of a criterion are missing.

3.2.4 Conclusion

In this chapter a selection of criteria is presented, to create an overview of the tracing capabilities of the various requirements management tools. This tracing capabilities overview will be created during the tool evaluation and used a basis for the implementation of the case study.

The whole information structure to store and trace requirements is represented in these criteria. It starts with the setup of a traceability model that can be used as a

reference for the rest of the project. In what detail can the reference objects be described, is it possible to specify different types of relations and can the relations be enriched with attributes? Then the creation of traces at the start of the project is covered. Is there support for automation to create links between certain artifacts automatically and is there a way to enforce the creation of traces between certain artifacts? Together with the creation of the links, the support for maintenance of the links in case of changes in one of the artifacts is investigated. And finally yet importantly, the usability of the links to improve the development process is examined. What views to inspect the traces are provided, is it possible to do a change impact analysis of the trace information and is it possible to inspect the link structure for inconsistencies?

3.3 Case study WASP

When investigating how each requirements management tool performs on each of the criteria, a requirements document with several layers of requirements will be useful. The requirements description for Web Architecture for Service Platforms (WASP) describes different scenarios that make use of context-aware mobile services to supply the end-user with information and services [EKS02].

The requirements for WASP are described using three requirements types. Scenarios give a high-level overview of typical situations the system should be able to handle. From these scenarios use cases are derived describing specific conditions mentioned in the scenarios. These use cases are used to create the final list of concrete functional and non-functional requirements. So this case describes several levels of requirements, as depicted in Figure 5, and is therefore ideal for the evaluation of the criteria.

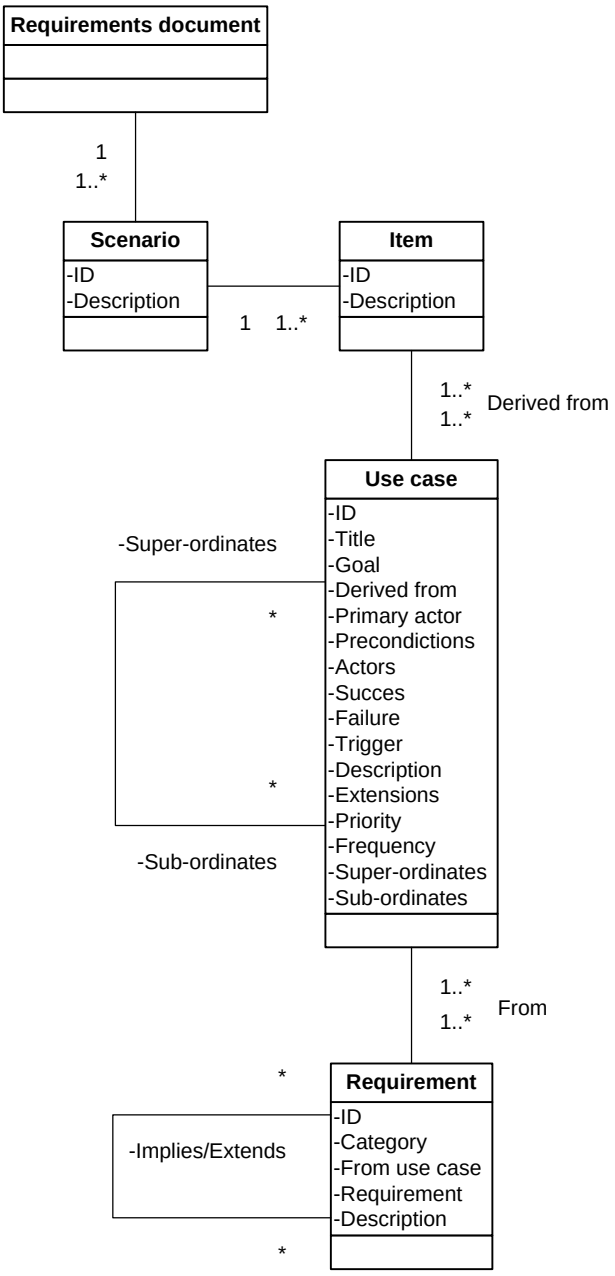

Figure 5: Information model for WASP 2.1

3.4 Conclusion

The following list of criteria has been composed to test for different aspects if change management and especially impact analysis:

Table 2: Final set of criteria

Criteria	Weight
1. Information model support	+
2. Links detached from artifacts	□
3. Uniquely identifiable reference objects	–
4. Traceability link types	+
5. Assign attributes to links	+
6. Automatic link detection	□
7. Enforce traceability links	–
8. Requirements documentation support	–
9. Tool integration	–
10. Graphical representation traces	+
11. Coverage analysis support	+
12. Impact analysis support	+

The case that will be used for these criteria is the WASP case, describing different scenarios that make use of context-aware mobile services to supply the end-user with information and services. This case has a clear information structure, with different types and levels of artifacts.

4 Impact analysis based feature support in tools

``There is nothing permanent, except change.''

Heraclitus of Ephesus, Greek philosopher, 540-480 B.C.

By implementing the case studies in the tool and looking at the underlying data structures the criteria, as described in section 3.2.1, have been evaluated and the results are presented per tool.

4.1 IBM Rational RequisitePro

IBM Rational RequisitePro is a requirements management tool that is part of IBM's Rationale Suite. This tool can be used to model requirements and store them in a relational database. A version history of the changes made to the requirements is maintained, so it is clear for everybody what has been changed and when. Requirements and artifacts can be linked to each other through traces. These traces make it possible to do a simple impact analysis to determine what other artifacts are effected when one of the artifacts is changed.

The tool also integrates with Microsoft Word, so requirements and other artifacts can be specified in Word and RequisitePro will be able to monitor them for changes or update them with information from the tool. Besides Word, it also integrates with some of the other tools in the Rational suite.

Since RequisitePro is in essence a requirements management tool, all artifact types and artifacts that are created are called requirement types and requirements by RequisitePro. Since both requirement types and requirements are completely customizable, they can be used to represent almost any artifact in the software development process. Therefore the word artifact will be used in this section for what is called a requirement in RequisitePro. The version used for this evaluation is IBM Rational RequisitePro 7.0.1.

4.1.1 Implementing case study

To determine how RequisitePro handles the various criteria defined in section 3.2.1, the case study from section 3.3 is being implemented. In this section the results of the implementation on all of the criteria are described.

4.1.1.1 Information model support

Tools shall allow the definition of requirements management information models.

The requirements management model as described in section 3.3 can be modeled partially in RequisitePro. Modeling the custom artifact types is completely supported. It is possible to create a type and assign a set of custom attributes to it. These attributes can be valued using either a user defined list or a free text field. The options for the list can be specified when creating the attribute and it is possible to indicate if only one of the values in the list can be selected or if it is possible to select multiple items at the same time.

Since RequisitePro has the option to import and monitor Word documents for attributes, it is also possible to specify document types as part of the information model. Such a document type contains a custom template so new attributes can be specified using a standard layout. A document type also has a default artifact type, which will be selected when a new attribute is marked in a document of this type.

Since it is possible to define custom representation objects for artifacts, all objects and their attributes in the WASP requirements management information model can be freely modeled. When multiple projects use the same information model, it is possible to create a template from an existing project. Such a template can contain only the various types (documents, requirements, attributes) of elements or also the content of the project for reuse in other projects.

Specifying the possible links between the various artifact types in the information model is not possible in RequisitePro.

Information metamodel

Through the implementation of the WASP 2.1 case study, a lot of information about the structure of RequisitePro was gathered. Part of this information is described in the evaluation of the criteria in the previous section. However not all the information related to RequisitePro's capabilities to store information to manage requirements is expressed in these criteria.

To show the exact possibilities of expressing requirements information, especially the traces among requirements and other artifacts, we created a model describing the tools modeling capabilities in Figure 6. The model is based on the results of the criteria evaluation and an inspection of the data structures used by RequisitePro.

As described in criterion two, the links are detached from the actual artifacts. They are not located inside the Word documents, but in the tool itself. The link between two artifacts can be of one of two types, a very specific hierarchical link or a more general trace link.

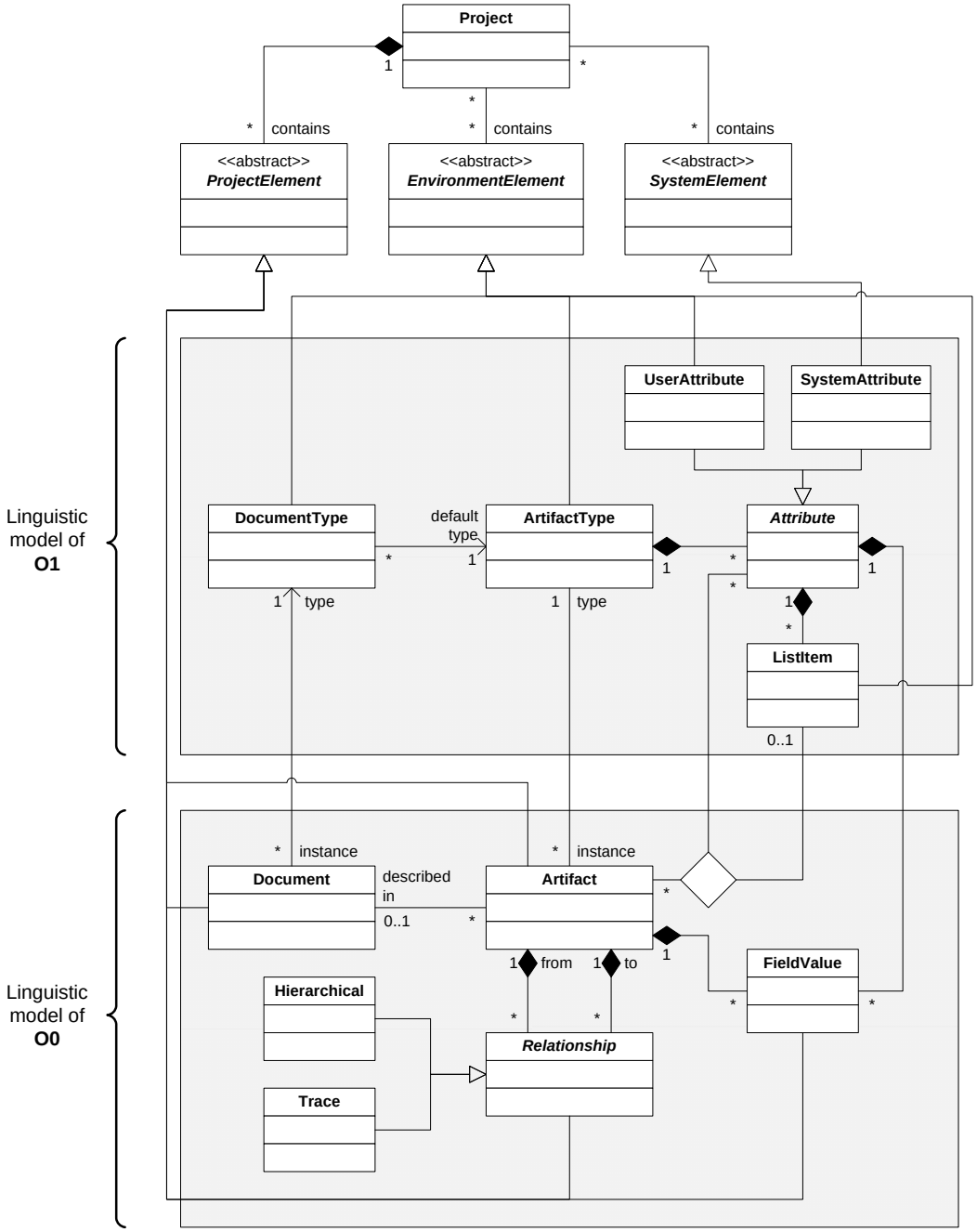

Figure 6: Metamodel for language to model requirements in RequisitePro

Example project

Although all artifact types in RequisitePro can be defined by hand, to match the project as good as possible, the tool is also provided with a number of default projects. Since RequisitePro is focused on the requirements management process, all the default projects only contain the requirements specification phase.

The example based on use cases as provided by RequisitePro is depicted in the object diagram in Figure 7. It contains a selection of the information model used to describe the requirements when using use cases. It starts with a vision document type, describing the high-level features that have to be delivered to the stakeholders and investors. Each feature in the document will become an artifact of the type product feature.

From this vision document, a number of documents containing more specific functional use cases and non-functional supplementary requirements are derived. Every requirement in one of these documents will be represented in RequisitePro, using an artifact of the default type of the document they are described in.

Eventually each artifact type has a number of attributes and if the attribute has a list as data type it also has a number of list items to choose from.

This example shows the flexibility of the customizable artifact types in RequisitePro. The requirements artifact types of both the WASP 2.1 case study and this RUP use case example can be modeled. It is however not possible to specify the structure of the various types of artifacts, since there is no way to indicate if it is possible or not to create relations between a number of artifact types.

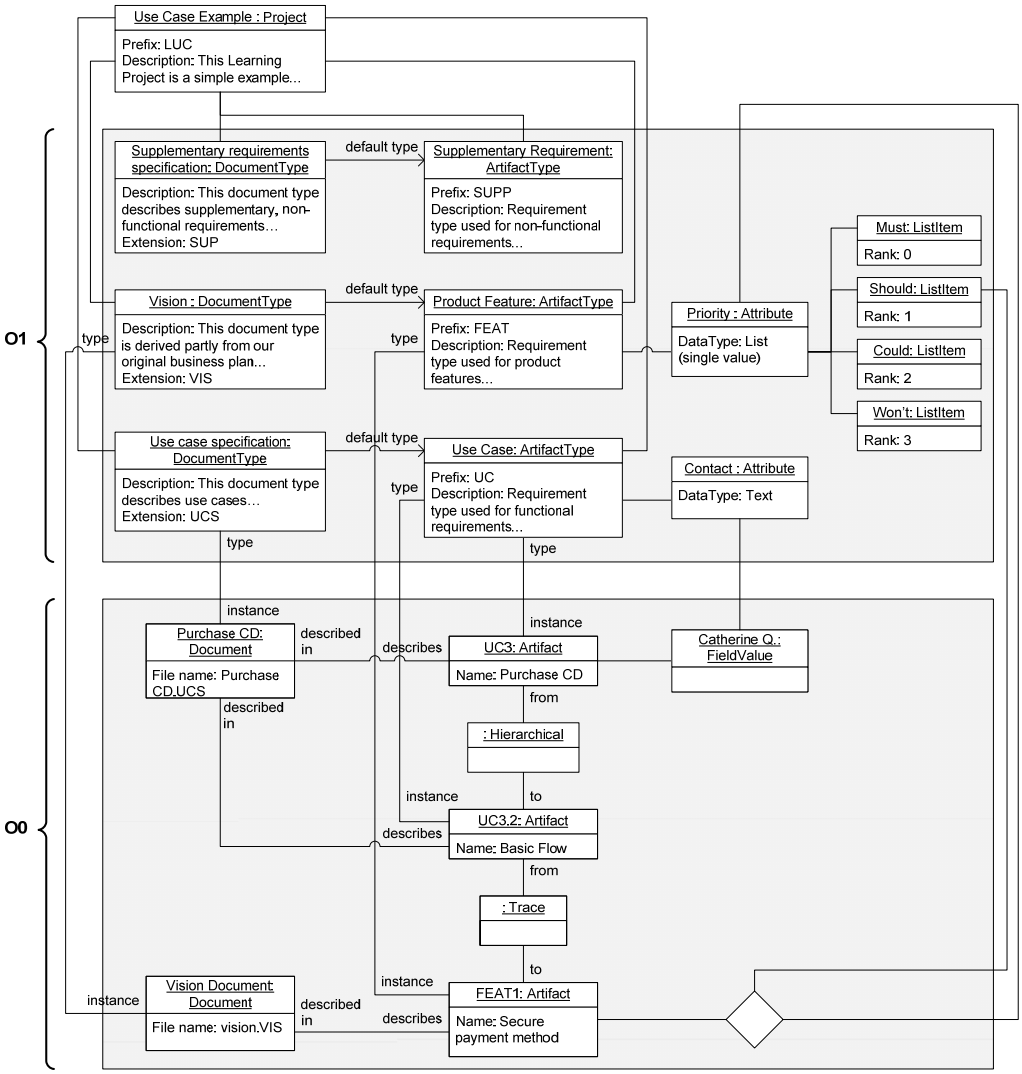

Figure 7 Object model for use case example in RequisitePro

4.1.1.2 Links detached from artifacts

Information about the link should not be stored in the artifact it originates from or points to.

RequisitePro stores no direct trace information in the artifact itself. It creates a representation of the artifact in RequisitePro and creates the links between these representations. However, there is some link between the actual artifact and its representation needed, so RequisitePro can monitor it for changes. This link is realized by assigning a unique identifier, as described in Section 4.1.1.3, to the requirement in the artifact and store this identifier in RequisitePro. This way RequisitePro can track the requirement even when it moves to another document, without keeping any link information in the artifact.

4.1.1.3 Uniquely identifiable reference objects

The tools must be able to uniquely identify a reference object in space and time.

Normally RequisitePro stores the Word documents in its own format, so they cannot be altered outside the application. However, this can be turned off. When the requirements document is altered without RequisitePro running, it can still detect changes in the requirement when it is started again. It is even possible to move a requirement from one document into another and RequisitePro will detect the change, recognize the requirement and will offer several options to solve the difference between RequisitePro and the document (Figure 8).

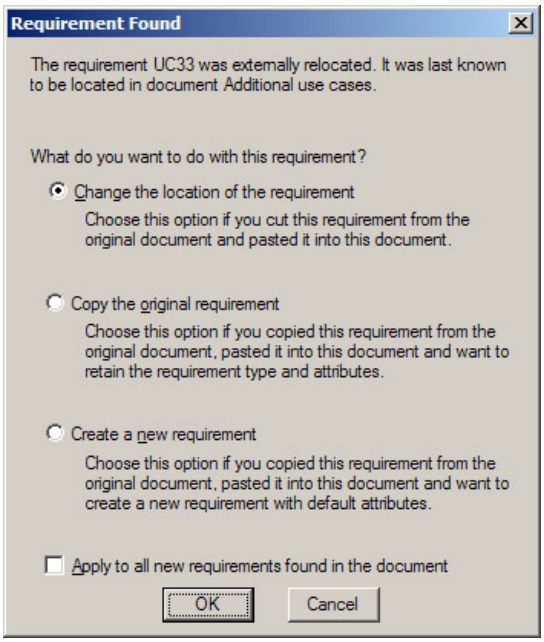

Figure 8: Requirement moved to another document

RequisitePro realizes this functionality by storing an identifier in the requirement using Microsoft Word's Bookmarks tag. This identifier is also stored in the requirement's representation in RequisitePro making it possible to link the original requirement to its representation regardless the document the requirement is in.

4.1.1.4 Traceability link types

It must be possible to use different types of traceability links, each with their own meaning.

In RequisitePro it is possible to use two link types to connect artifacts. The first link type is the general trace link. It can be used to link any two artifacts represented in RequisitePro. It can be created from the origin artifact as a 'trace to' link and from the target as a 'traced from' link. Besides the direction of the link, no additional information can be stored in a link.

The second link type is the hierarchical relationship. Hierarchical relationships are parent-child relationships between artifacts of the same type. Hierarchical relationships can be used to subdivide a general artifact into more explicit artifacts. Child artifacts provide additional detail for their parent artifact [IBM06]. This means each parent artifact can have multiple child artifacts of the same type as the parent. Every child artifact can have only one parent, but a child can be the parent of other artifacts.

When artifacts are changed, the links with the directly related artifacts are marked as ‘suspect’ (see section 4.1.1.12 for details). If two artifacts are linked to each other through an indirect link between one or more other artifacts, this indirect link is marked ‘indirect suspect’ if one or more of the individual links are marked ‘suspect’. However if one of the links in this indirect link is a hierarchical relationship, the indirect link will not be marked as ‘indirect suspect’.

The limitations of the hierarchical link make it quite specific and only usable in a limited number of relations. The WASP 2.1 requirements specification uses several relations between artifacts, some examples: derived from, super-ordinates, sub-ordinates, extension of and implied by. None of these can be represented using the hierarchical relationship, since it is not used during impact analysis.

Besides the hierarchical and trace links, it is not possible to create any other custom link type.

4.1.1.5 Assign attributes to links

It must be possible to assign attributes to links.

As described in criterion 4, the only two link types possible are the hierarchical link and the general trace link. For both it is not possible to add any additional information like attributes. The only information stored with a link is its direction and status.

4.1.1.6 Automatic link detection

The tool should be able to automatically detect relations between artifacts through for example information retrieval techniques, examining active code during test scenarios, monitoring of change history, naming schemas or model transformations.

There is no option in RequisitePro to automatically detect links between artifacts. It does maintain links when artifacts are moved from document to document as described in criterion 3. But since only the external documents change in this case and

not the internal state of RequisitePro, it is questionable if this is really maintaining the links.

4.1.1.7 Enforce traceability links

The tool must be able to enforce the creation and change of certain traceability links.

As described at the evaluation of the first criterion, it is not possible to specify between which types of artifacts links are allowed. So RequisitePro is not able to enforce the creation of any links between artifacts. All relations have to be created by hand. There are some limitations to where links can be created. It is for example not possible to create cycles in the links. A parent can have only children of the same type of artifact, they have to be described in the same document and these children can have only one parent. The last limitation is that it is not possible to create a normal trace link, when a hierarchical link is already in place between two artifacts.

4.1.1.8 Requirements documentation support

The tool needs to be able to read and import requirements from requirement specification documents created outside the tool, in for example Microsoft Word. At the same time it needs to support the creation of documents describing the requirements in the tool.

RequisitePro is able to import Word documents containing requirements. The individual requirements are recognized by formatting style or special character string denoting the beginning and end of a requirement. It is not possible to detect or import any link information that may be present in the specification of the requirements.

RequisitePro also supports the exporting of requirements to a document. This is however limited to one type of requirements at time and it only creates a table containing the names of the requirements and their attributes. It is not possible to structure the output with for example a custom template. Besides the requirements themselves, it is also possible to export the views, like the completeness and impact analysis, on the requirements. However this will not be presented in a Word document, but in a plain text file as comma separated values.

4.1.1.9 Tool integration

The tool needs to be able to interface with other tools in the product life cycle to make information stored in them visible and linkable.

By default, RequisitePro integrates with Microsoft Word. All requirements written in Word can be imported in RequisitePro and once that has been done, RequisitePro monitors these requirements for changes. If there are differences between RequisitePro and the document, it asks what happened and which of the two versions should be kept. RequisitePro is also able to manipulate the Word document, so changes made directly in RequisitePro are also present in the originating Word document.

Since RequisitePro is part of IBM's Rational suite, it is able to exchange requirements information with some of the other application in the suite. The integration with Rational ClearQuest, a change request management tool, looks most useful. It allows

the user to link change requests to requirements in RequisitePro, making it easier to determine the impact of a request. Since ClearQuest is not available right now, the integration options with this tool has not been tested.

Besides the integration with Word and maybe Rational, there are no external tools RequisitePro can exchange requirements information with. There is also a RequisitePro Extensibility server (RPX), which can be used to access the objects in the database. The GUI library can be used to control the user interface.

4.1.1.10 Graphical representation traces

The tool must provide graphical user-friendly representation of traces.

RequisitePro has several ways to present relations. The hierarchical relations are shown in the three representing the various types of artifacts, documents and views. By default only the parent containing the children is shown. The parent can be expanded to show the children directly linked to the parent. Children that are also the parent for other children can be expanded in the same way.

The general trace links can be shown in two ways. The first is by using a matrix view. The rows of the matrix are one type of artifact and the columns can be the same or another type of artifacts. When there is a direct trace link between the two types of artifacts, an arrow indicates the direction of the trace on the intersection of the row and the column. It is also possible to show the indirect trace links between the two types of artifacts. These are represented using semi-transparent arrows.

The second way to show general traces is with a tree. This tree shows all artifacts traced to or from a specified type of artifact. It shows only direct links, but it does this for multiple levels of artifacts. Both these views and their settings can be stored in the project for easy reuse.

4.1.1.11 Coverage analysis support

The tool must be able to produce an overview of the artifacts that have no traces to previous or subsequent artifacts.

RequisitePro supports a matrix view showing two types of artifacts, type A in the rows of the matrix and type B in the columns (which can be of the same type). Each type can be filtered for certain attribute values using a query. To show if there are any requirements without traces to subsequent requirements, only the artifacts of type A are enlisted with no direct traces to any of the artifacts of type B. In the same view it is possible to enlist all artifacts of type B without traces from type A in the columns of the matrix.

It is not possible to show the missing relations between more then two types in the same view. So an individual view is needed for each pair of types. If the traces between the two types are not all in the same direction (all relations trace from the scenarios to the use cases) there are two views needed. One for the traces from type A to type B and one for traces the other way around.

4.1.1.12 Impact analysis support

The tool must provide a way to perform an impact analysis to determine the effected artifacts.

The criterion about graphical representations shows that there is a matrix view to show the traces and their direction between two types of artifacts. Therefore, it is possible to find the forward and backward traces between two phases. When an artifact in the source phase has been changed, this matrix view also shows which target artifacts can be affected by this change. There is however not a way to automatically determine which mappings have to be preserved. This has to be done by hand.

Normally a change affects artifacts in more then two phases, since the artifacts in the target phase of the first mapping, often act as the source phase for the next phase in the development cycle. Therefore, the system, to detect mappings that have to be changed or preserved between two phases, has to be extended so it takes in account the mappings in and between multiple phases.

RequisitePro supports limited impact analysis between multiple phases. Using the matrix view it is possible to show the indirect traces between two types of artifacts, created through the transitivity of traces from and to the phases in between. Both forward and backward traceability are used to create the indirect links. It is however not possible to determine which of the suspected links indicate if the mapping between the artifacts has to be changed or preserved. Since it is only possible to show the suspected links between two types of artifacts, a view for every pair of artifacts has to be created.

The multiphase impact analysis is therefore only useful as an indication of where to look for artifacts that have to be changed or preserved. It is no complete solution to find all mappings that have to be changed, preserved or that need special attention since they are affected by multiple mappings.

4.1.2 Summary

During the implementation of the WASP 2.1 case study, the criteria created to evaluate the capabilities of the various requirements management tools have been described for RequisitePro in section 4.1.1. The outcome of these criteria is summarized in Table 3.

Table 3 Summary of criteria evaluation for RequisitePro

Criteria	Rating	Result
1. Information model support	□	Specification of custom artifacts is possible; the specification of relations and custom traces is not.
2. Links detached from artifacts	+	Links are stored in RequisitePro and not in the Word documents themselves.
3. Uniquely identifiable reference objects	+	Every requirement in the Word document has a unique identifier, linking it to the reference object in RequisitePro.
4. Traceability link	□	• Trace relation: general trace link used in

types		most cases; • Hierarchical relation: link type that can be used in only a very limited number of cases.
5. Assign attributes to links	-	It is not possible to assign any attributes to links.
6. Automatic link detection	-	There are no special techniques to detect links.
7. Enforce traceability links	-	Since it is not possible to specify any possible links in the information model, it is not possible to enforce them.
8. Requirements documentation support	□	Limited to Word documents. During import no trace information can be detected. Exporting only possible according to standard structure.
9. Tool integration	□	Limited to Microsoft Word and other IBM Rational products.
10. Graphical representation traces	+	A matrix view and a tree view are available.
11. Coverage analysis support	+	Direct coverage between two types of requirements.
12. Impact analysis support	□	It is possible to create a list showing only artifacts directly related to the changed artifact, or it is possible to show all indirectly effected artifacts in the matrix view.

4.2 Borland CaliberRM

Borland's requirements management tool is called Caliber Analyst. It consists of two parts: Caliber DefineIT, a tool to define and refine requirements at the beginning of a project, and CaliberRM, a tool to manage requirements and help with collaboration, impact analysis and communication during the software lifecycle.

CaliberRM is the real requirements management tool we are interested in. It maintains traceability information between various requirements in Caliber, but also between artifacts in other tools. It also offers a number of ways to keep an overview of all requirements and it combines these features to analyze the impact of change proposals. At the same time, it maintains a history of all changes to both artifacts and the links between them, so it is possible to go back in time to see what changes have been made previously to the requirements. The version tested for this evaluation was Borland CaliberRM 2006 9.0.847.0

4.2.1 Implementing case study

To determine how Caliber handles the various criteria defined in section 3.2.1, the case study from section 3.3 is being implemented. In this section, the results of the implementation on all of the criteria are described.

4.2.1.1 Information model support

Tools shall allow the definition of requirements management information models.

The WASP 2.1 requirements management model, as described in section 3.3, can be modeled in Caliber to some extend. It is possible to create custom artifact types to match the ones present in the requirements management model. Besides the name of this type of artifact, a suffix can be specified that will be used as the start of the artifact identifier. To set the various artifact types apart, it is possible to assign attributes to an artifact and presented them in different groups. The rest of the properties for an artifact type have to do with user rights and access levels. These describe for example who can remove a specific artifact type (no one, the owner or everybody) and who can change which attributes. They also determine if artifacts of this type are hidden and if they can be created or only viewed.

The custom artifact types, specified when implementing the information model for a project, are also available in other projects. This means they can be reused in new projects. So specifying these types can be seen as a good investment for the future, if most of the software development projects are based on the same artifact structure.

The attributes for the artifacts can be created in the same way as the artifacts themselves. Attributes have a name, a type (all possible types are enlisted in Figure 9) and a description. The effect of this attribute on the major version number and the status of the trace links of an artifact in case of a change is specified next. As with the artifacts, the last set of options can be used to specify the user rights. For each artifact type it is possible to specify who can change this artifact (no one, the owner or everybody).

Beside these attributes specified by the user, there are also a number of system attributes that are created by default for each artifact type. They specify for each artifact for example the owner, the id, the version number, the status and the priority. In case of the priority and status it is possible to change the list entries so they match the project being modeled.

It is not possible to define any information about the links that can exist between the various types of artifact types. The original WASP information model allows only specific link types between certain types of artifacts. It is however not possible to define this information when specifying the information model in Caliber. So when modeling the actual requirements, users can specify links between any two artifacts, which conflicts with the original information model.

Caliber supports the specification of different types of requirements, each with their own set of attributes, but it is not possible to use or specify different link types or between which types of requirements links are or are not allowed.

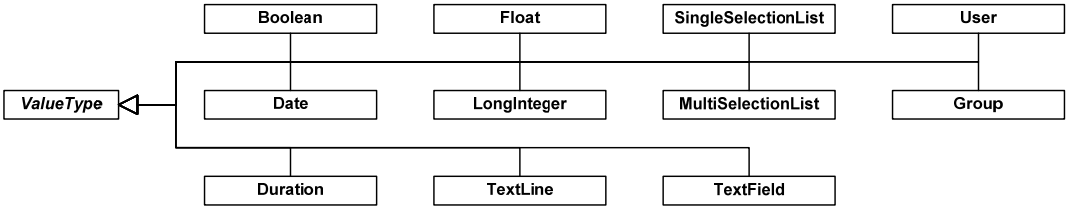

Figure 9: Attribute types in Caliber

Information metamodel

Just as for RequisitePro, a lot of information about Caliber was gathered through the implementation of the WASP 2.1 case study. Not all this information could be described using the criteria, so the information metamodel for Caliber is depicted in Figure 10. The model is based on the results of our criteria evaluation and an inspection of the data structures used by Caliber.

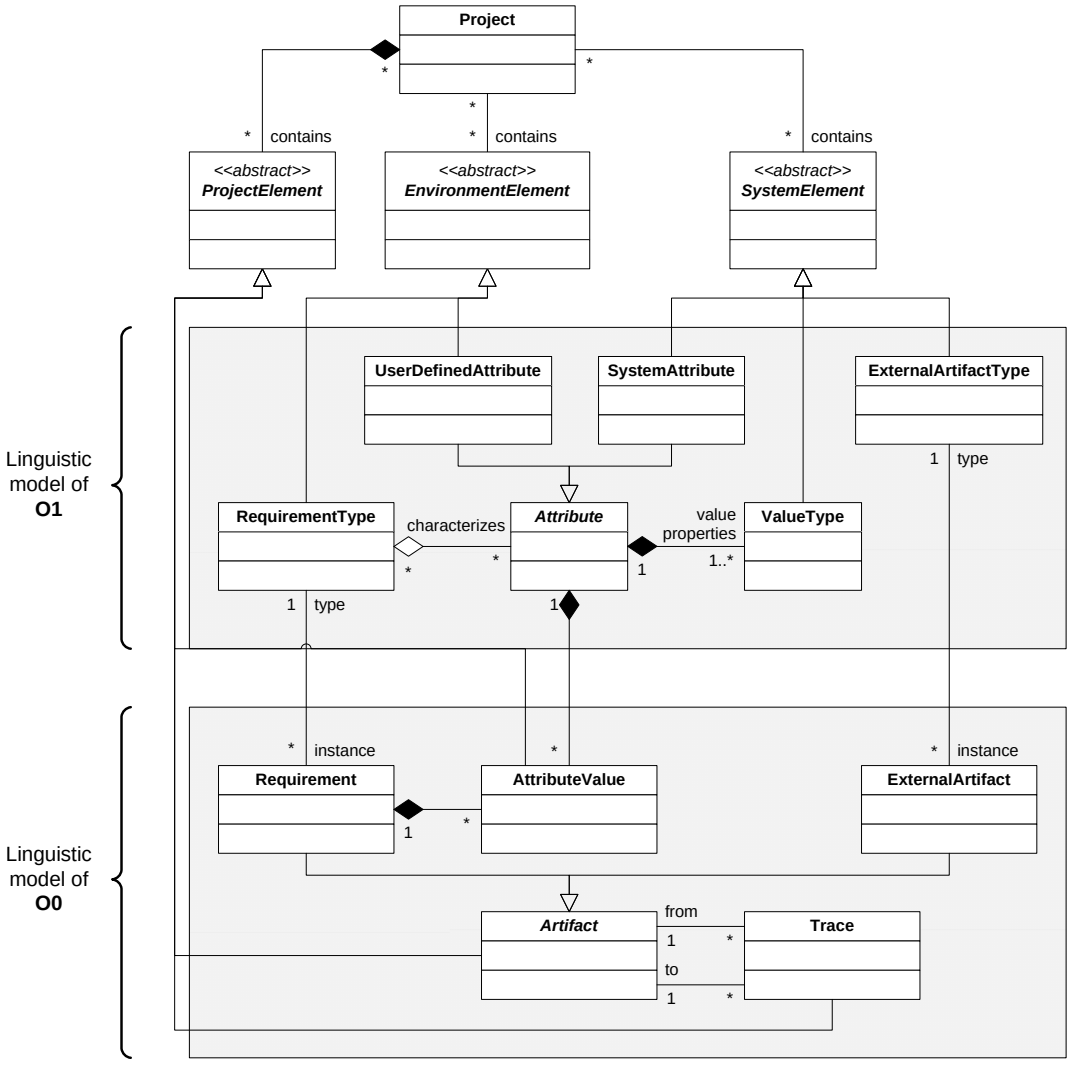

Figure 10: Metamodel for language to model requirements in Caliber

4.2.1.2 Links detached from artifacts

Information about the link should not be stored in the artifact it originates from or points to.

The artifacts have to be imported into the tool before any traceability links can be created to or from any other artifacts. So both the artifact and the link information between them is always stored in the tool itself and not in the artifact documents.

It is possible to create a traceability link between an artifact in the tool and its originating document. This link is only maintained in the tool, so here the link is also detached from the document.

4.2.1.3 Uniquely identifiable reference objects

The tools must be able to uniquely identify a reference object in space and time.

When using Caliber, this tool is supposed to be the main repository of artifacts. The link with the originating document is lost after importing the document. Afterwards it is possible to create a reference or traceability link to the originating document, but it is not possible to specify more detail then the location of the document in the file system. Once the artifacts are imported in Caliber, they get their own unique identification number, so the representations of the original artifacts are uniquely identifiable.

4.2.1.4 Traceability link types

It must be possible to use different types of traceability links, each with their own meaning.

Caliber supports one general type of traceability links between artifacts. It is just a normal traceability link that can be established between any two artifacts indicating that there is some relation between two artifacts. These artifacts can be of the same type or different types and even external artifacts, like files, UML elements or test cases.

The relation between a parent artifact and its child in the hierarchy, supported by Caliber, could be seen as a link too. This relation is however not used for any traceability purposes like the determination of indirect links, so it is not seen as a separate traceability link type.

4.2.1.5 Assign attributes to links

It must be possible to assign attributes to links.

A traceability link in Caliber stores which artifacts are linked together and the direction of the link. Furthermore, it maintains the status of the link, showing if the link is suspect or not as a result of a change in one of the two related artifacts. Besides the status attribute, it is not possible to assign any additional attributes to traces. Nor is it possible to customize a link in any other way.

4.2.1.6 Automatic link detection

The tool should be able to automatically detect relations between artifacts through for example information retrieval techniques, examining active code during test scenarios, monitoring of change history, naming schemas or model transformations.

It looks like Caliber has no option to automatically detect links. The first option to create links is by specifying them in de traceability section of an artifact's details. It is possible to select a target artifact from the project three or one of the external artifact source and create a from- or to-link to this selected artifact. The second way to create links is using the traceability matrix. By selecting the intersection between two requirements, it is possible to create a link in the desired direction. By selecting multiple intersections at once, it is possible to create or change multiple links at the same time. The matrix view can also be used to change the direction of links or to remove them.

There is however no way to create links in an automated or even semi-automated way. It is not possible to assign certain identifiers to a special attribute during import, to create links, nor is it possible to automatically create links to source artifacts when importing.

4.2.1.7 Enforce traceability links

The tool must be able to enforce the creation and change of certain traceability links.

It is not possible to enforce or prevent the creation of links between two types of artifacts. So once the actual artifacts are being modeled in Caliber, it is possible to create links between any two types of artifacts.

4.2.1.8 Requirements documentation support

The tool needs to be able to read and import requirements from requirement specification documents created outside the tool, in for example Microsoft Word. At the same time it needs to support the creation of documents describing the requirements in the tool.

Since Caliber maintains all requirements inside the tool, without monitoring the originating document, extensive import and export techniques are needed to prevent data deterioration.

Import
Calibers import option only supports Microsoft Word documents at the moment. Before a document can be imported, the artifact type of the artifacts contained in the document has to be specified in the project, since the imported artifacts will be created as the children of a type. Once a Word document has been specified, the styles in the document are retrieved and can be used to specify which styles contain the names and descriptions of the artifacts. If the artifacts are structured in a hierarchy, this hierarchy can be preserved by selecting all styles containing the names of artifacts in the various levels present in the document. Besides specifying a style to select the artifacts description, it is also possible to select all the content between two artifact names, between two delimiters or containing a number of keywords. Besides text, Caliber also supports images and tables in the content. It is however not possible to

capture more details besides the artifact description. So it is not possible to automatically assign values to user defined attributes based on keywords, delimiters or other information available in de content of the artifact.

Once the import tool has retrieved all artifacts in the document, a preview showing the names, the description and the structure is presented. In this preview it is possible to reorder artifacts on the same level and to change artifact names. Once the order and names are correct, the location where the artifacts will be inserted into the existing structure can be specified.

When all the artifacts are extracted from the document and imported in Caliber, there is no link with the originating document anymore. Therefore, in the case that the artifacts in the document are altered, the tool will not be aware of this in any way. It is possible to create a traceability link from the document to the representation of the artifact in Caliber, but when the document changes, this is not detected by the tool either. Even when someone wants to update the artifact descriptions in Caliber by telling it the document has to be imported again, this is not possible. Caliber will import the artifacts as a new set of artifacts, with no option to inherit the user defined attributes or traceability links from the original artifacts.

Once the document with the artifacts is imported, the artifacts should be managed from Caliber and the document should not be used anymore to make changes.

Export
Caliber has a several options to export the artifacts in projects. The simplest options just export (part of) the Caliber database into another format. Caliber works with its own Microsoft SQL Server or Oracle Database. Project information from all these databases can be exported to Microsoft Access. When only a subsection of the information is needed, for example a traceability matrix, it is also possible to export this information in a simpler Microsoft Excel sheet or an XML file. The same holds for the requirements grid, an overview of all artifacts and their attributes, it can be exported to an XML, a comma separated or a plain text file.

When a more detailed overview of the current project is needed, Caliber offers a number of reports that can be generated from the artifacts in the current project. These reports can be focused on the artifact details, status or responsibility. The details report shows a list of all artifacts with their attributes. The status report shows all artifacts and their system attributes and groups them by their status. The responsibility report shows for each registered user what artifacts he or she is responsible for. The quick reports will be exported as XML files.

If previous reports are not sufficient and even more specific details or a predefined structure are needed, the Caliber document factory can be used. This tool supports the creation of templates for Microsoft Word, plain text and rich text documents that can be used to represent the artifact information present in the tool. The templates use commands to structure the artifact data and fields to insert the actual information about the artifacts into the document. The commands to structure the artifacts can be divided into formatting, sorting, definition and filtering of scopes.

A scope is used to select the information that has to be shown. This can be a set of artifacts of one type, one particular artifact or even an attribute belonging to a particular artifact. It is possible to specify a custom scope with its own filter to select a type or concrete artifact or attribute, but it is also possible to use one of the default scopes containing often-used information. These default scopes enclose information like the discussions, history and traces for artifacts or even the whole project.

Inside one of these scopes, it is possible to apply several sorting options to present artifacts in an order relevant to the type of document containing them. It is possible to sort by artifact identifier or the hierarchy as present in the tool, but it is also possible to sort by artifact type, priority, status or even a number of the user-defined attribute types. A scope is also used to specify the formatting of the artifacts it contains. Most formatting can be done when the template for the document is created. By applying markup on the keywords used to insert the actual information, the actual information in the final document will have the same markup.

The import options provided by Caliber are limited to Microsoft Word documents only. The options for exporting information are more extensive. Not only is it possible to export the database to various other database formats, it is also possible to export this type of data to Microsoft Excel, XML, comma separated or plain text files. For more detailed and specific reports about the current status of requirements, a series of reports can be generated by Caliber. When even more control about the details, order and formatting is needed, it is even possible to create export templates to export information to Microsoft Word documents.

4.2.1.9 Tool integration

The tool needs to be able to interface with other tools in the product life cycle to make information stored in them visible and linkable.

Caliber offers a number of connectivity options with other tools. Most of these tools are also made by Borland, but there is also support for tools from other suppliers. According to the external traceability options, the following tools are supported by Caliber: files in the files system, Borland Together, Borland SilkCentral, Borland StarTeam, HP Quality Center and Microsoft Visual Studio Team System for tests and work items.

Borland Together

Since there are several tools that should integrate with Caliber, the connectivity of one tool is tested. Since the connectivity with one of Borland's own tools is probably the best, Borland Together was chosen as it is also the next step in the software development life cycle and available on the systems at the university. Together is a Java IDE combined with an UML modeling tool and based on the UML models from the modeler it is possible to generate a Java framework in which the final code can be entered.

Besides using information from Caliber, it is also possible to connect Together with RequisitePro. The requirements in either tool can be linked to almost any UML element in Together: classes, attributes, procedures, packages, interfaces, but also activities in activity diagrams, etc. Only the links between elements cannot be linked to. The communication about link information is performed by the Together *trace*

synchronizer. This tool shows all traces to and from objects in Together. When something changed in Caliber or Together the information between both tools has to be synced manually.

It is not clear what has to be changed in Together to mark an object changed and the link with Caliber suspect. Changing the name of an object in Together caused no suspect links and the name change was performed in Caliber without a warning. Adding an attribute to an object did not result in any suspect links either. Both cases should not cause any problems in their related requirements, so the absence of the suspect status can be explained. However when the type of an attribute is changed, the links to the corresponding requirements are still not marked suspect. When this is an important attribute, this behavior could lead to serious problems.

Besides this limited suspect marking, the communication between both tools is also limited. For example when a class was removed in Together, the link to the requirement was also removed in Together. After synchronization with Caliber, it was still present in Caliber and its status was still normal and not suspect. When the description of a requirement in Caliber was changed, the link to the Together element was marked suspect in Caliber. After synchronization, the same link in Together was not marked suspect. Restarting all tools and reconnecting them, made the suspect status appear in Together too. However, when the status was cleared in Together, it was still marked suspect in Caliber after synchronization.

The communication between both tools seems to be not very robust. On multiple occasions one of the applications stopped responding completely during a synchronization action. After several hours, it was still unresponsive and performing a hard reset on the application was the only way to get it running again.

If the communication with the other tools is as limited as with Together, these tool integration options are limited.

Other tools
Besides integration with Borland's other tools, it is also possible to export requirements to a select number of applications from other manufacturers. The first application to export to is Borland StarTeam. This is a version system for artifacts in a software project. It is mainly targeted at source code, but it is also supports requirements. There are even some options to create links between the supported artifacts. The second application to export to is Mercury or HP Quality Center. This is a test management tool that uses requirements to see if all requirements are validated by tests and that these tests test what the requirement describes. At the same time it is also used to make sure there are no tests without a requirement and thus testing code that is probably not needed or not linked to the right requirement.

4.2.1.10 Graphical representation traces

The tool must provide graphical user friendly representation of traces.

When viewing artifacts in the main view, the hierarchy of the artifacts can be seen in the left column. The hierarchical links are however not used for the traceability of this relation. To let the hierarchy have any effect a normal link must be created between the parent and the child. Normal traces to other artifacts can be found in the

traceability tab of an artifact. There is a separate list for the direct in- and outgoing links showing the target artifacts, the status of the links and in which project or file the target artifacts resides. Besides inspecting the traces, it is also possible to modify the links from this view. It is possible to remove links, change their target or their status. There is however no indirect link information available in this tab.

Traceability matrix
For a clear overview of all links between artifacts, Caliber offers two traceability views. The first view is presented using a matrix with a selection of artifacts in the rows and columns of the matrix. Only the artifacts present in Caliber are shown in this view, artifacts in other tools or the file system and the links to or from them are not shown. When there is a direct link between two of these artifacts, an arrow in the direction of the link will be shown on their intersection. It is also possible to show indirect links between two artifacts, these will be shown as dashed arrows. There is however no easy way to see which direct links are responsible for an indirect link. When a change is made to an artifact, it is direct in- and outgoing links are marked suspect by showing a question mark next to the arrow. Besides indirect links, it is also possible to mark artifacts with no links at all.

By default, all artifacts for the active project will be shown in both the rows and the columns. In case of a serious project, the amount of artifacts and thus the size of the matrix can become overwhelming. Therefore, the matrix view offers an option to filter the row and column artifacts based on project, artifact type, link direction, link status and even attribute value. To make these filters even more specific they can be combined using logical operators.

All the settings for a traceability matrix can be stored in a so-called view. These views are stored as files in the file system and to enable them again, they have to be loaded from the file system too. Therefore, they are not as easily accessible as for example in RequisitePro, where they are part of the projects hierarchy. The filters used to select the artifacts for the rows and columns are stored in Caliber itself and are therefore easier accessible then the views.

Traceability diagram
The second view for traceability data is the traceability graph. It shows the selected artifact in the middle, all artifacts with direct incoming links on the left and all artifacts with direct outgoing links on the right. All incoming artifacts on the left also show all artifacts with direct links to them and this happens until all artifacts with direct links to the selected artifact or any of its predecessors are shown. The same happens for the artifacts with outgoing links on the right of the selected artifact. All the artifacts they link to using outgoing links are shown too and this is step is repeated again for these new artifacts.

The traceability matrix is a good help to determine which artifacts are related and this traceability graph helps to see how artifacts are related. It makes it possible to see which direct links are responsible for an indirect link in the traceability matrix. When a link between two artifacts is marked suspect, the arrow in the graph becomes red with a question mark next to it. The diagram also supports links to and from artifacts in external artifacts and tools.

There are three ways to depict links between artifacts in Caliber. The traceability information shown at the artifact details, gives a detailed overview of all links related to a particular artifact. The matrix view gives a quick overview of all links between a large group of artifacts, including indirect and suspect links. At the same time it can be used to quickly create multiple links between artifacts. The diagram view shows a graph showing all in- and outgoing links for a particular artifact and all artifacts linking indirectly to this artifact. It can therefore be used to inspect indirect traces and the possible effect of a change on any artifacts before or after this one.

4.2.1.11 Coverage analysis support

The tool must be able to produce an overview of the artifacts that have no traces to previous or subsequent artifacts.

In the traceability matrix there is a button to mark all artifacts without any in- or outgoing links. This option does however not allow filtering for artifacts with only missing ingoing links or missing outgoing links. A useful option to detect artifacts that could represent an unnecessary feature (no incoming links) or a feature not being implemented (no outgoing links).

Using the custom filters available for the traceability matrix it is possible select only artifacts with *no trace from* relations, to detect missing incoming links, or *no trace to* relations, to detect missing outgoing links. Since all types of artifacts are shown in the same grid by default, it is possible to get an overall view of all artifacts with missing links. When there are too many artifacts present in Caliber for this view to work, it is also possible to filter for certain types or artifacts or even specific attributes and their values.

4.2.1.12 Impact analysis support

The tool must provide a way to perform an impact analysis to determine the effected artifacts.

All impact analysis in Caliber must be done by hand. The three ways to inspect links, as described in section 4.2.1.10 about graphical representation, are also the tools used for impact analysis. The details view for a particular artifact can be used see which other artifacts are directly linked to this one. By doing this again for all these linked artifacts and their links it is possible to analyzing the impact. This is however very time consuming and by using the other two views, it can be done much faster.

The traceability matrix has the option to show indirect links. These indirect links show which predecessors the links to a selected artifact have and which successors a link from this artifact has. When an artifact is changed only the direct in- and outgoing links for this artifact are marked suspect. So this cannot be used to determine indirect suspect links. The parent-child hierarchy is not used either in the calculation of indirect links. When the relation between a parent and a child is relevant for the analysis, a normal link should be created between them. Sometimes it is not easy to see what the intermediate steps of an indirect link are. To make these steps visible the traceability diagram can be used.

When the effect of new or changed links is investigated, the matrix view has some options to make the effects of this change visible. When a new link is created or an

existing link is changed in the traceability matrix, it is marked in another color and so are the indirect links that are a result of these new links

The indirect links from the matrix and diagram views are a good start, but to find all possible effected artifacts each relation from and to the changed artifact must be investigated manually. When a link becomes suspect as a result of this change, it can be marked suspect in the matrix view and all links from and to the other side of this link will be investigated in the same way. Using this strategy the impact of a change can be determined.

The links between artifacts have to create a structure between artifacts to make clear which artifacts are predecessors or successors of an artifact. This way it is possible to locate the source or end result of a particular artifact. To maintain this structure, it should not be possible to create cycles in the links between artifacts, since that would make it impossible to locate some of the sources or targets of artifacts in this cycle. In Caliber it is however possible to create cycles in the artifacts. An example is shown in Figure 11 where *user requirement 3* links to number 4, number 4 links to number 2 and number 2 links back to number 3. This behavior should not be allowed.

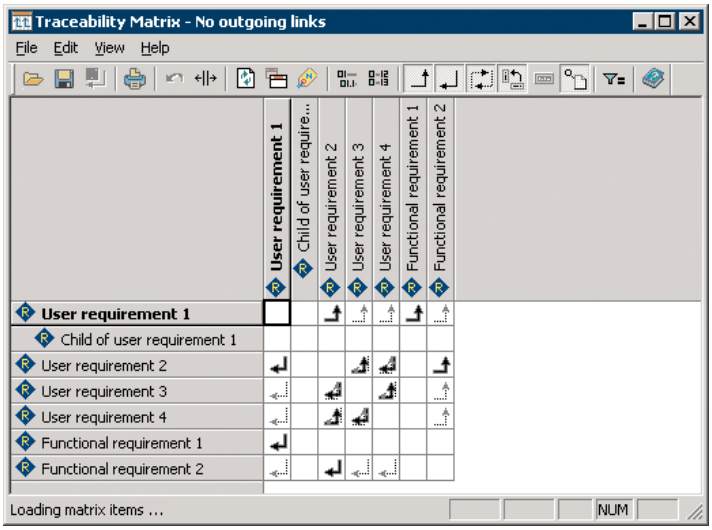

Figure 11: Cycle in links between artifacts

Caliber supports tools for manual impact analysis, but also allows the creation of cycles, making the evaluation of traces potentially difficult.

4.2.2 Summary

During the implementation of the WASP 2.1 case study, the criteria created to evaluate the capabilities of the various requirements management tools have been described for Caliber in section 4.2.1. The outcome of these criteria is summarized in Table 4.

Table 4: Summary of criteria evaluation for CaliberRM

Criteria	Rating	Result
1. Information model support	□	It is possible to create different reusable artifact types each with their own set of attributes. It is not possible to specify link types.

2. Links detached from artifacts	+	Links are not stored in the original artifacts, but in a separate table in Caliber.
3. Uniquely identifiable reference objects	+	Reference objects in other tools are addressed by their internal unique identifiers. Only reference objects in the file system cannot be identified uniquely.
4. Traceability link types	-	There is only one generic link type.
5. Assign attributes to links	-	Besides the direction and status, it is not possible to assign any attributes to links.
6. Automatic link detection	-	It is only possible to manually create and update links.
7. Enforce traceability links	-	There is no option to limit the creation of links between specific artifact types.
8. Requirements documentation support	+	Most import and export options are targeted at Microsoft Word, but the options to create documents and reports is quite extended.
9. Tool integration	□	Various tools can share traceability information with Caliber, but the quality of the information exchange is unreliable.
10. Graphical representation traces	+	Three possible ways to investigate links on different levels.
11. Coverage analysis support	+	It is possible to filter artifacts with no incoming, no outgoing or no links at all.
12. Impact analysis support	□	No way to do an automated analysis, everything has to be done by hand using the graphical representations. It is also possible to create cycles, something that should not be possible in a traceability tool.

4.3 TopTeam Analyst

TopTeam is the only tool in this survey with native support for more then only textual requirements. It also supports use cases and test cases to support the requirements specification process. Since these additional artifacts do have semantics, TopTeam is less flexible then some of the other tools. When a project uses UML in the way it has been implemented in TopTeam, it can be a great help. When a project is however not using UML as it has been implemented in TopTeam, or it even does not even use UML at all, its less flexible setup can become a problem.

The documentation was very limited in the version used for testing. There is a document explaining the higher-level concepts and there are a number of movies describing how to work with the use cases in the tool. There is however no general help system present. There are some links in the tool to websites that should provide, but now of testing these websites were not available. To make sure no functionality was missed, a demo given by Synergio, the Dutch distributor of TopTeam, was attended. The version tested for this evaluation was TopTeam Analyst 3.20.

4.3.1 Implementing case study

To determine how TopTeam Analyst handles the various criteria defined in section 3.2.1, the case study from section 3.3 is being implemented. In this section the results of the implementation on all of the criteria are described.

4.3.1.1 Information model support

Tools shall allow the definition of requirements management information models.

In contrast with some of the other tools, TopTeam Analyst supports more then only text-based requirements. It has also native support for use cases and test cases. For each project that is managed in TopTeam Analyst it is possible to specify which of these three types will be available. Besides primary elements, it is for example also possible to specify at the setup of the project who will be working on it, but since this is not directly related to requirements management, it will not be discussed. In the case of projects that are more complex it is possible to create one or more child projects and store them in the main project.

Once a project has been created in TopTeam Analyst, it is time to model the information model. The different artifact types used in a project are called record types in Analyst. By default, a number of parent record types are available, like Documents, Base requirement type, Context diagram, Diagrams, etc. representing basic artifacts. More specific record types can be created based on these parent types. Besides a parent record type, a name, prefix and description must be provided on creation of the new type.

Once the new record type is created, additional characteristics can be specified. States can be used to describe the current condition of a requirement. These states can be used in project rules, for example to warn a team member when a record gets in a certain state. Rules also specify allowed state transitions and what actions are allowed (modify the record, change the type, add or remove attachments or comments, etc). Next to states it is also possible to assign attributes, called fields in TopTeam Analyst, to each new record type. The last relevant option related to requirements management is the option to specify what type of links can be created to what types of records from this particular record type.

All new record types are stored centrally in TopTeam Analyst and are therefore immediately available to all new and existing projects for reuse.

Information metamodel

Just as for RequisitePro and Caliber, a lot of information about Analyst was gathered through the implementation of the WASP 2.1 case study. Not all this information could be described using the criteria, so the information metamodel for Analyst is depicted in Figure 12. The model is based on the results of our criteria evaluation and an inspection of the data structures used by Analyst.

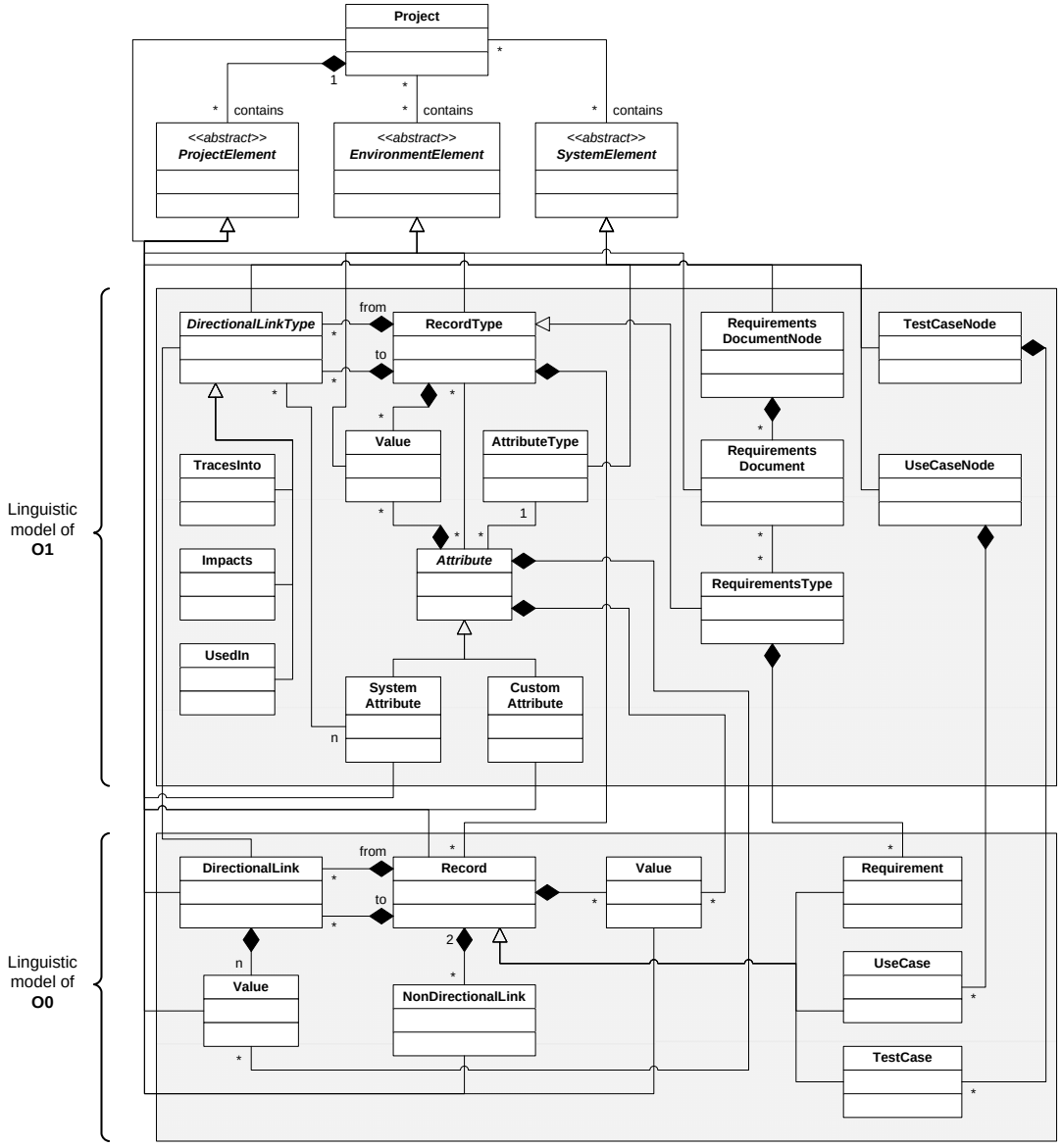

Figure 12: Metamodel for language to model requirements in Analyst

4.3.1.2 Links detached from artifacts

Information about the link should not be stored in the artifact it originates from or points to.

The real artifact, the original document with requirements, is only used during import; afterwards there is no link with the document anymore. So the links are detached from the original artifact by default.

Once the artifacts are stored in the tool, it seems the links are still detached. Changes to links are immediately visible at both the source and target of a link and no problems with asynchronous link information have been encountered during the investigation of the tool.

The smallest artifact to link to in TopTeam Analyst is a record. Therefore, it is not possible to create traceability links to elements in a record, for example a step in a use case or a model element. Non-traceability links, like the link between the reference to an actor in a use case and the actual actor, can exist between elements smaller then a record, but they are not part of the traceability model.

4.3.1.3 Uniquely identifiable reference objects

The tools must be able to uniquely identify a reference object in space and time.

The source of records, the original document, is not uniquely identifiable, since the links to files only refer to the location in the file system. So it is not possible to link to the specific location in the file, or to determine if the original was changed or moved in the file system.

The records in the tool itself, representing the various artifacts, are uniquely identifiable using a unique identifier composed of a type abbreviation and a number.

4.3.1.4 Traceability link types

It must be possible to use different types of traceability links, each with their own meaning.

TopTeam Analyst supports three directed trace types: traces into, impacts and used in. In the documentation these links are called *dependency* type relationships, but an individual description for each of the types is missing. During a TopTeam demo, it became clear that the three trace types present in the tool have no other semantics then there is a directed link between two records. Any difference between the links must be created, maintained and interpreted by the users of the tool. It was also confirmed that it is not possible to specify custom directed traces besides the three traces available in the tool.

Next to the three types of directed links, there is also a non-directed link available. This link is used when there is no clear dependency relation, for example between an actor and its representation in a use case. Non-directional links are not used in the traceability model.

Besides these four links described in the documentation and clearly visible in the tool, there are also a number of hidden links in the tool, behaving like traces. The <<includes>> and <<extends>> relations are only available for use cases, but they are directional and visible in the Repository explorer three as links. They are also shown in the traceability tab of the related records and it is possible to make these links suspect. When an <<includes>> relation is created in a use case diagram, but the use case being included is not described, the application shows a warning. These two relations are however not part of the traceability model.

There are also special links for screen mockups (navigates to/from), which are also available in the Repository explorer three as links. They are however not in the traceability tab of the related records and not part of the traceability model.

Just as with the directed links, it is not possible to create custom links with this kind of semantics.

4.3.1.5 Assign attributes to links

It must be possible to assign attributes to links.

The three default traceability links have a number of default attributes. The identifiers of the source and target records are logical attributes for a link. Next to a source and a target, a link in TopTeam Analyst also has two changeable attributes: a suspect attribute and a rationale text field. There are also a number of attributes not changeable by the users: the creator and updater of the link and the time and dates this happened.

It is not possible to add custom attributes to these links.

4.3.1.6 Automatic link detection

The tool should be able to automatically detect relations between artifacts through for example information retrieval techniques, examining active code during test scenarios, monitoring of change history, naming schemas or model transformations.

During the creation of the flow-of-events in a use case, it is possible to detect terms and actors automatically, since they are specified already. The links between the actors in the flow and the actual actor is not a real traceability link, but it makes navigation at least easier and it is the closest to automatic link detection until now.

4.3.1.7 Enforce traceability links

The tool must be able to enforce the creation and change of certain traceability links.

For each record type it is possible to specify to which other record types links can be created and which of the three directed links (traces into, impacts, used in) are allowed. When creating links between records only the selected record and link types are available. So it is possible to enforce the creation and changing of links according to the information model. It is however not possible to enforce at least one link between a specific record and another type.

4.3.1.8 Requirements documentation support

The tool needs to be able to read and import requirements from requirement specification documents created outside the tool, in for example Microsoft Word. At the same time it needs to support the creation of documents describing the requirements in the tool.

Import
TopTeam supports different ways to import requirements into the tool.

Style based
The first import method is based on the styles used in documents created with Microsoft Word (rtf or doc). Since the method needs Word to interpret the files, it only works when Word also installed on the system. To import requirements the location of the requirements document must be specified, together with the styles used for requirements titles and requirements descriptions. When the document is scanned, an overview showing all detected titles and their description is presented. This overview can be used to change the order, hierarchy level, type, title and description

of each individual requirement found during the scanning process. It is also possible to remove, merge and split requirements before actually importing them into TopTeam. The Word styles are stored and used in TopTeam Analyst and used again during export.

In theory this a well executed import system, but after importing several documents a number of problems appeared. After selecting the styles that had been used for the requirements descriptions, the import method also returned several sections with non-selected styles. When the name of a style contains a colon, it will not show up in the list of selectable styles during import. Renaming the style in Word solves the problem, but since a lot of custom styles have colons in their name, this is a cumbersome process. On another occasion, importing all titles in the document went ok, but random descriptions were missing. In the end it took three attempts to get all descriptions right. When the results of the scanned document are edited, before actually being importing in the database, there is no option to undo changes made to the results of the scan. Therefore, one has to be careful when performing operations where data is changed or removed.

It is not possible to assign certain styles from the document to attributes of the various requirement types.

Normal (legacy)
When the requirements are not available in a supported rich text format, it is possible to import it using the normal import method. After specifying the requirements document, it is shown in the same tool as the requirements after the scan of a document containing styles. Requirements are created by copying them from the original document and pasting them in a record. The same order and hierarchy tools are available as in the style based import tool, together with options to remove, merge and split requirements.

Import requirement from file (excel)
Requirements in tabular form (csv, txt, doc, xls or mdb) can also be imported. After specifying the file location and the type of requirements available in the document, a preview of the columns is presented. Each column can be mapped to a specific attribute of the selected requirement type, making it probably the best way to exchange requirements between tools. Once the columns have been mapped to attributes, the requirements that are actually being imported can be selected.

Import a use case from a document
Besides importing plain text requirements, there is also an option to import more structural requirements, like use cases, into TopTeam Analyst. When a text file containing the use cases is imported, it is presented next to an empty use case. Use case sections in the text document have to be moved manually into their respective elements in the TopTeam use case record. Terms and actor names mentioned in the use case steps are automatically linked to the instances with the same name in TopTeam Analyst. This procedure makes the importing of textual use cases into the tool as easy as possible.

Export
Records can be exported using a number of project or system wide templates. These export templates work in a similar way as the templates in Caliber. Using a simple scripting language a sub selection of records is fetched, for example all records of a specific type, and sorted based on an attribute, like its identification number, status or priority. The formatting of a TopTeam template can be done using Microsoft Word templates. This makes it possible to not only format the record title and description, using the styles they had during import, but also the rest of the attributes without formatting information from the import process. These export templates can be used for normal requirements, use cases and test cases.

For most graphical representations of record data, like trace representations or use case flow diagrams, it is possible to export them as images to the clipboard or a file. Various raster and vector formats to store these images are available, like jpeg, bmp, svg and emf. Use cases can also be exported as XML file or XMI (XML Metadata Interchange) file.

4.3.1.9 Tool integration

The tool needs to be able to interface with other tools in the product life cycle to make information stored in them visible and linkable.

Support for the integration with other tools is not mentioned in the documentation, the website, or the tool itself. Microsoft Word is used to import formatted documents and it is possible to export data in various formats, like XML; XMI; Excel; etc, for use in other tools. It is also possible to send e-mail messages when a record with a subscription is changed or information is needed from other users.

A demo showed that the interaction with other tools is indeed limited. Most important options are the export of test cases to Visual Studio and Quality Center and the option to export use cases to Visual Studio. There is however no support for the integration of traceability data with these tools.

TopTeam promotes Analyst as a tool supporting more then only text-based requirements, but also graphical use cases and testing information. Through the integration of these different artifacts into one tool, and not a whole suite of different application like some other venders do, they claim to offer an easy solution. This is probably also the reason for the little amount of integration with other tools.

TopTeam is however working on an API to make it possible to export data to other IDE’s then Visual Studio and Quality Center.

4.3.1.10 Graphical representation traces

The tool must provide graphical user-friendly representation of traces.

There are four ways to inspect traces.

Individual records
Each record has its own tab with all traces to and from other records. These traces can be used to navigate to the linked artifacts. The same information is also present in the

repository explorer, showing all requirements types and the various link types related to them, in a tree.

Traceability matrix
It is possible to select the type of artifact (business rules, software requirements, and use cases) and the specific project and document containing these artifacts in the rows and columns.

By default, the links presented in the matrix are only trace links. It is however also possible to use any other type of link available in the tool. This includes all UML links like forward traces as <<extend>>, <<include>>, Impacts, Includes, Navigates to, etc and reverse traces like Dependent on, extended in, included in, etc.

Once all these options are filled out, the matrix showing the links between artifacts from the rows to the columns is presented. From this view it is possible to create trace in and to links, remove links, make links suspect, clear the suspect status and add notes to links to add a form of rationale.

Traceability network diagram
Shows one or more threes of record types (features, use cases, screen prototypes) and the relations (traces into, impacts, used in) between them. It is possible to show four record types and links are only shown from a lower level number to a higher-level number. Relations are depicted as arrows and when a relation is selected details of the relation (type, source, target, suspect status, rationale) are shown. When a record is selected all records with direct links to or from this selected record are shown in a bold blue font and arrow.

To make the view more clear, it is possible to hide selected records. It is also possible to zoom in and out, change the width of the text for each record tree. Since it is not really possible to show indirect links, links from higher to lower record types and the number of record types in the diagram is limited to four, it is easy to miss relations between record types, making it useful for direct link analysis, but maybe less useful for higher level impact analysis.

Besides inspecting links, it is also possible to create links by specifying the type of link to be created, select the option to create new links, and start drawing from the source to the target record. It is also possible to mark/clear links as suspect, to add a rationale for a particular link or to permanently remove a link.

It seems there is not really a way to store a view with the selected record types so it can easily be reused.

Multi-level traceability report
A multi-level report shows all traces from a specified record type to a depth of a specified number of levels. It is also possible to show traces to the selected record type. It is also possible only selected record types in the network or only specific trace types. The report is text based and shows all identifiers and titles of the selected record type and indents when one of the records has one or more traces to other records, showing all records with the specified trace types. When one of these records has traces to other records, the same happens.

4.3.1.11 Coverage analysis support

The tool must be able to produce an overview of the artifacts that have no traces to previous or subsequent artifacts.

Like the other tools, TopTeam also supports the filtering of requirements based on their properties. The requirements tree shows the hierarchy of all requirements and also allows the use of various filters. The default set of available filters contains a filter to isolate requirements without any outgoing links (traces into, impacts, used in) and a filter to isolate only requirements without outgoing traces into relation. Using custom filters, it is also possible to isolate any of the other link types or requirements with no incoming links.

This works however only for the record types in the requirements group. For the use cases there is not really an option to easily filter for use cases without in- or outgoing links. There are however various ways to show for each use case (or other record type) if there are any in- or outgoing links. The object list can be used to filter a specific record type and by adding a traceability explorer or diagram, it can be easily seen if there are any links from or to this particular record. But all records must be scanned manually, which can be a problem in larger projects.

The Multi-level traceability report can be used for coverage analysis. It shows all outgoing or incoming links from one type of record to all other types or a sub selection of record types. This allows for easy manual analysis.

4.3.1.12 Impact analysis support

The tool must provide a way to perform an impact analysis to determine the effected artifacts.

Suspect links
When a record is changed, direct links are not automatically marked suspect. In the traceability tab it is however very easy to select all traces and mark them suspect all at once. This is however a manual task, that can be forgotten when many changes are made at once.

It is possible to subscribe to specific records. When one of these records is changed by someone else, the subscribers get a message. The message contains the name of the record, the user who changed the record and a link to the record for a quick inspection.

When the suspect status is added or removed, no rationale is asked.

Tracking items
Besides the normal records like requirements and use cases, TopTeam also supports so-called tracking items. These items are records to support the use of the normal records. The default tracking item types are: change request, issue, problem report and enhancement request. When a change request is created for a record, it is also possible to create items called impacts, describing the impact of this change request on other records. The original change request in divided in, for example, these impacts to make it easier to plan and manage the implementation of the request.

Such a system can be very useful for the management of changes to the project, but at the moment the tracking system does not seem to use any traceability information present in the project. There is support for the storage of the various impacts of a change request, but there is no real support for find these impacts using the tool.

Besides the traces, it is also possible to compare different baselines to see what has been changed since the last baseline moment.

4.3.2 Summary

During the implementation of the WASP 2.1 case study, the criteria created to evaluate the capabilities of the various requirements management tools have been described for Analyst in section 4.2.1. The outcome of these criteria is summarized in Table 5.

Table 5: Summary of criteria evaluation for TopTeam Analyst

Criteria	Rating	Result
1. Information model support	□	It is possible to create different reusable artifact types each with their own set of attributes. It is also possible to specify what link types are allowed between specific artifact types.
2. Links detached from artifacts	+	Links are not stored in the original artifacts, but in a separate table.
3. Uniquely identifiable reference objects	+	Artifacts in the tool can be identified using their unique identifiers. Reference objects in the file system cannot be identified uniquely.
4. Traceability link types	□	There are three traceability link types: traces into, impacts and used in. It is not possible to create custom link types.
5. Assign attributes to links	□	It is not possible to assign custom attributes to the links, but it is possible to mark links suspect and to specify a rationale for the link.
6. Automatic link detection	-	It is only possible to manually create and update links.
7. Enforce traceability links	□	It is possible to specify which link types can be created between what artifact types. There is however no way to enforce the checking of existing and new links after a change to an artifact has been made.
8. Requirements documentation support	+	Most import and export options are targeted at Microsoft Word, but the options to create documents and reports is quite extended using a script language to order and format artifacts.
9. Tool integration	-	It is only possible to export record data to tools. There is no back- and forward integration or option to synchronize traceability data.
10. Graphical representation traces	+	Various ways to investigate links on different levels are available.

11. Coverage analysis support	+	It is possible to filter artifacts with no incoming, no outgoing or no links at all.
12. Impact analysis support	-	At the moment it seems like impact analysis support is very limited. Links are not marked suspect after a change, it is not possible to show indirect links and it seems there is no way to use the *impacts* and *used in* link types in a useful way.

4.4 Telelogic DOORS

DOORS is the requirements application offered by Telelogic as part of their collection of tools to support the software engineering lifecycle. DOORS is mentioned in several papers and is often referred to as very capable requirements management tool [Dic02, DP98, Wie95].

DOORS is clearly targeted at organizations with multiple people working on requirements at the same time. A server maintains the database with all projects, requirements, links, etc. and users can login on this database using the client application. Based on their user rights they can view or change information in the database. The version used for this evaluation is TeleLogic DOORS 8.3.

4.4.1 Implementing case study

Using the criteria specified in section 3.2 Telelogic DOORS will be evaluated to see if it is as capable as claimed, using the case study from section 3.3. In this section, the results of the implementation on all of the criteria are described.

4.4.1.1 Information model support

Tools shall allow the definition of requirements management information models.

DOORS uses a database to store its entire project, just like the other tools. This database can contain multiple projects and each of these projects can have its own information model. A model contains a number of modules to store the links and objects representing the actual requirements.

Each project can contain three types of modules:

- **Formal modules** are used to store representations of the requirements artifacts. Each artifact is partitioned in smaller objects, which are the smallest linkable sections of a requirements document. These objects can be structured in a hierarchy to make navigation easier. Link modules are the second type and are used to store the links between two objects, to indicate that there is a certain relationship between them.
- A **link module** contains a linkset for each pair of formal modules with links between them. These linksets can be created during the specification of the information model to indicate between which formal modules links can and cannot be created and which link modules will be used for these links.
- The third and last type of module is called the **descriptive module** and is not really used anymore. Originally it was used to store the original artifact, but since

a representation of the artifact is now stored in the formal module, use of the descriptive module has no real advantage anymore.

The structure of these modules and their content is depicted in Figure 13. Since the descriptive module is not being used anymore, it is left out.

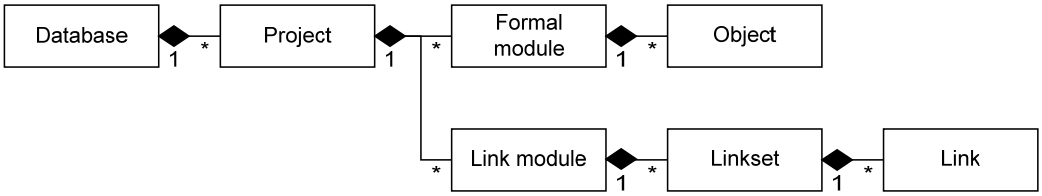

Figure 13: Overview of elements in DOORS

Each module, object and link has a number of system attributes to store properties needed by DOORS and it is possible to also specify custom attributes for these three elements. Such a custom attribute has a name and a type. There are a number of system types, like string; integer and date, but it is also possible to specify a custom type with additional restrictions. All system types have been depicted in Figure 14.

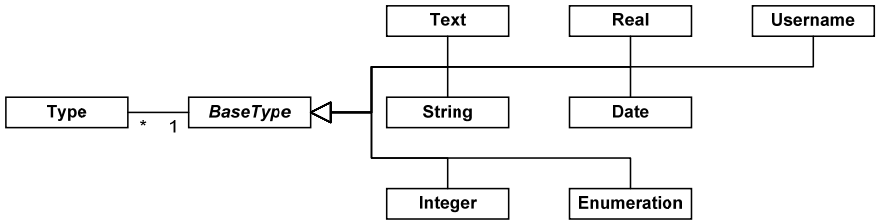

Figure 14: System attribute types in DOORS

These elements are available to build the information model before any real requirements are created. Each type of requirement, scenario; use case; system requirements; etc, can have its own formal module with optional custom attributes. Each type of link between those requirements can have its own link module with also an option for custom attributes. When specific links can be used between two formal modules, the linkset storing these links, can already be specified to make the creation of the actual links later on easier. If there is only specific subset of link types allowed between two formal modules, this can be specified in the source formal module.

It seems like the options to specify an information model in DOORS are quite extended. It is possible to create different formal modules each with their own attributes to store different types of requirements. Link modules can be created to store specific link types, again each with their own set of attributes. To help the users during the requirements creation and maintenance phases, a number of limitations on the creation of, for example, links can be enforced, etc.

Information metamodel

Just as for RequisitePro, Caliber and TopTeam a lot of information about DOORS was gathered through the implementation of the WASP 2.1 case study. Not all this information could be described using the criteria, so the information metamodel for DOORS is depicted in Figure 15. The metamodel is based on the results of our criteria evaluation and an inspection of the data structures used by DOORS.

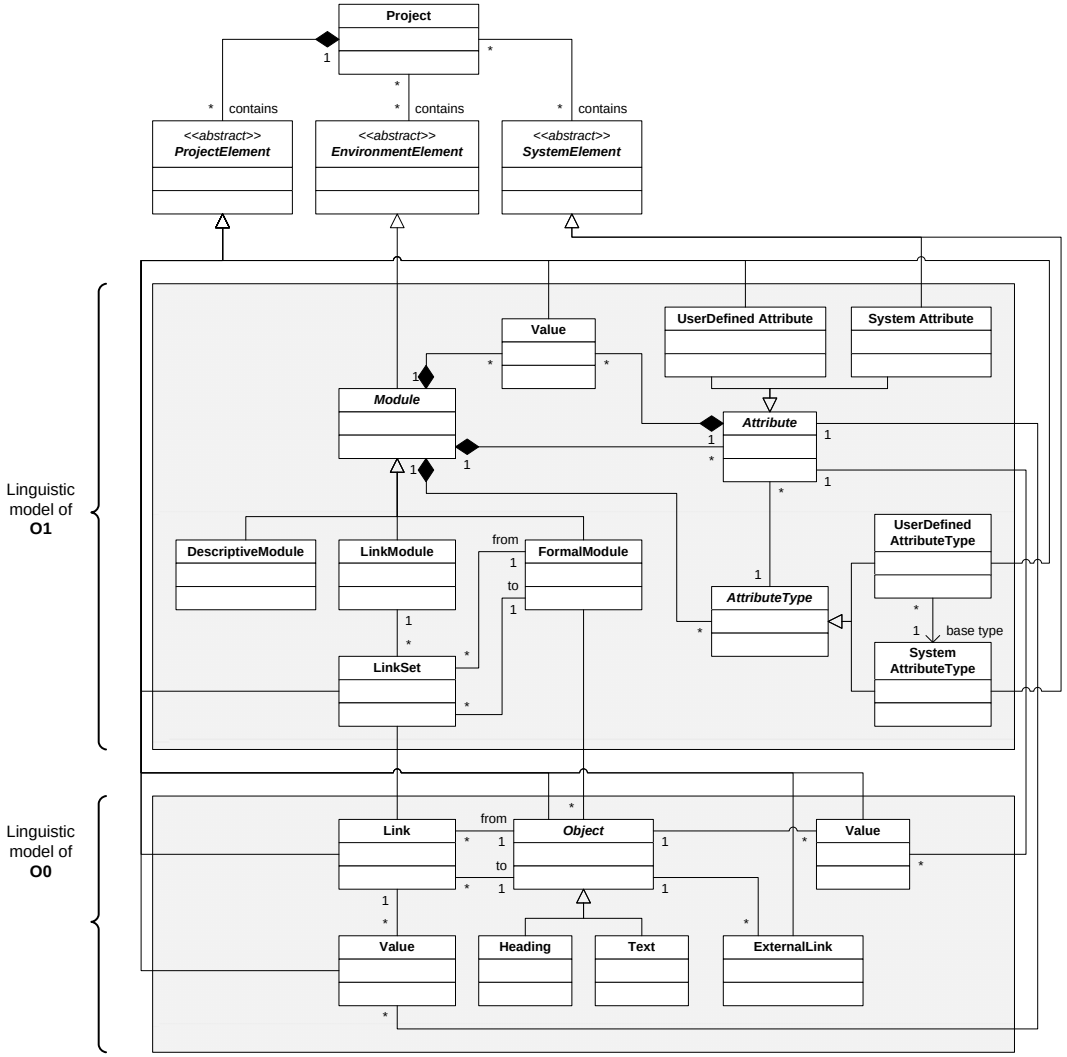

Figure 15: Metamodel for language to model requirements in DOORS

4.4.1.2 Links detached from artifacts

Information about the link should not be stored in the artifact it originates from or points to.

DOORS imports an artifact into a number of objects in one of the formal modules. At this moment no links referring back to this source artifact are created, so all links created in DOORS will always be detached from the source artifact. The links to and from DOORS objects are stored in their own link modules, so the internal links are also detached from the objects.

To link to artifacts not available in DOORS, external links are used. These external links are plain URLs, so as long as an external artifact is a file or originates in an application supporting URLs, it can be linked to. External links linking to artifacts outside DOORS are stored in DOORS and are therefore also detached from the artifacts they link to. These external links can also be used to link to DOORS objects in other databases, since DOORS also supports URLs to links to most elements available. These DOORS URLs can also be used in other documents and applications to link back to elements in DOORS. These links will be stored in the actual artifacts,

as they fall outside the scope of DOORS, and are therefore not detached from the artifacts.

Links are detached from both the source artifact and the representations in DOORS itself and stored in their own link modules. Only the external links are closely integrated with their reference object in DOORS, but this should not be a problem, since these links have only a source in DOORS.

4.4.1.3 Uniquely identifiable reference objects

The tools must be able to uniquely identify a reference object in space and time.

As a result of DOORS importing artifacts into modules and discarding the link with the source artifact, uniquely identifiable reference objects in artifacts are not relevant for normal links. The objects in the DOORS modules have their own unique identifier and the combination of project; module and this identifier make them uniquely identifiable for links in DOORS.

It is possible to create links to artifacts outside DOORS using external links. These external links use URLs to identify the external artifacts in the file system or tool containing them. This moves the responsibility of uniquely identifiable objects away from DOORS, it allows for URLs that can uniquely identify an object, and moves it to the people creating files or the tools creating artifacts to make sure they are unique.

4.4.1.4 Traceability link types

It must be possible to use different types of traceability links, each with their own meaning.

DOORS supports semantics for two link types: internal and external links. Internal links can be created between any two objects in the same database, while external links can be used to link to external artifacts.

Internal links are stored in linksets in the standard DOORS links module. It is possible to add custom attributes to links, so there is an option to add custom types to links. There is however no native support for specific internal link types in DOORS. Semantics must be added manually by using the custom attributes in filters and scripts during analysis of the links.

Another way to specify different link types is by storing them in different link modules. By giving these modules names like *derived to* and *refined to* it becomes easier to see which links are used where. An additional advantage of this method is the ability to allow only linksets in certain link modules to be used between two formal modules. This feature can be used to enforce specific link types between formal modules. Using link modules and linksets to specify link types, has one disadvantage: it is only possible to have one linkset between two formal modules. This means that only one link type can be used between two formal modules.

Besides the links in the link modules, it is also possible to assign external links to objects. External links are stored with the source object and can contain a link to another DOORS database or an URL. An external link is more than just embedding

an URL in a text attribute. It can also have its own set of custom attributes and it is for example part of the traceability wizard.

Example link to a bookmark in a Word document:

```
file:///H:/Afstuderen/Tools/document_with_bookmarks_to_link_to.
doc#second_section
```

This link will be opened by DOORS in a window of the systems default browser. It depends on the browser how the link will be handled further. In the case of this Word document, Internet Explorer will open the document in the browser window itself and navigates to the optional bookmark present behind the number sign.

Besides linking from DOORS to external artifacts, it is also link from external artifacts back to objects in DOORS using the same URL system. An URL to link back to a DOORS object looks like this:

```
doors://EWI620:36677/?version=1,prodID=0,dbid=473b06d8125c3fa0,
container=00000024,object=182
```

DOORS supports two link types: normal and external links. Normal links can be stored in link modules that can represent different types of links. The semantics of these different link types are however not used by DOORS during link analysis.

4.4.1.5 Assign attributes to links

It must be possible to assign attributes to links.

DOORS supports the assignment of attributes to both the link module and the links themselves. The type of the attribute can be of one of the system types, as shown in Figure 14, or a custom type. It is also possible to indicate if a change in this particular attribute should mark the containing object as changed to help with change management and impact analysis. When an attribute should only be accessible for specific users or a group or users, it is possible to assign access rights to specific attributes.

4.4.1.6 Automatic link detection

The tool should be able to automatically detect relations between artifacts through for example information retrieval techniques, examining active code during test scenarios, monitoring of change history, naming schemas or model transformations.

DOORS supports a number of ways to create links between objects. The first way of creating links between objects is by assigning them manually. The object to link to or from is marked active, after which the second object can be selected and a link can be created in the direction required. It is also possible to create links for multiple objects at once, as long as they are adjacent, and in this case it is only possible to create links from the source objects to the target objects. There is no easy way to change the source or target for an existing link. The only way to change the source or target is by removing it and creating a new one. Besides creating links in the main view, it is also possible to create links in the grid view. By selecting the intersection of two requirements, a new link can be created, edited or removed.

The second way to create links is semiautomatic. The identifiers of the objects to link to must be entered in a text attribute of the object to link from. Using the *link by attribute* option the formal module with the target objects, a link module to store the links in and the direction of the links can be selected. When links to more then one module are needed, each of these modules will need its own attribute. When the identifiers in one or more of the attributes have been changed, the linking procedure has to be repeated manually. When identifiers in one of the attributes are removed and the linking procedure is executed again, the links created by the removed identifiers are not removed. So the *link by attribute* option only creates links and does not remove them. Since this option only allows the use of DOORS' own identifiers, it cannot be used to create links based on the information in the original artifact, since the identifiers generated by DOORS during import are not known yet at that moment.

The only way to automatically create links between objects is when the DOORS change proposal system is used. When a change proposal is created, the system will automatically create links from the change proposal to the objects the proposal refers to. These links to meta-information are however not what we are looking for in a tool. This means there is no automatic way to create links between objects. The options for manually creating and changing links are limited, making the maintenance of links a time consuming job and a potential source for invalid links.

4.4.1.7 Enforce traceability links

The tool must be able to enforce the creation and change of certain traceability links.

In a formal module it is possible to specify a list of target modules to which can be linked and the link module used to do this. This triple of source, target and link module can be made mandatory or overridable. The default is mandatory, which means that the link module as specified in the triple is used. When the triple is overridable, the links will be stored in a link set in the users default link module.

When the triples in this list are the only outgoing link allowed, this could be specified in the source module. In this case, users are not able to create links from the source module to any target module that is not specified in the triple list. The link module used to create these allowed links, still depends on the mandatory or overridable flag of each triple.

The rationale for this criterion was the creation or updating of links after changes were made, to make sure the object is still linked to the right target objects and existing links are still valid. Since DOORS only supports the prevention of certain link types between two formal modules, this criterion is not supported.

4.4.1.8 Requirements documentation support

The tool needs to be able to read and import requirements from requirement specification documents created outside the tool, in for example Microsoft Word. At the same time it needs to support the creation of documents describing the requirements in the tool.

Import
Just as the other tools DOORS is able to import requirements specified in various types of text documents. Support for Microsoft Word, rich text format (RTF) and

plain text documents can be used to import the text version of requirements. DOORS also supports files created by Framemaker, a tool to create technical documentation. The support for comma and tab separated files (CSV and TSV) is mainly targeted the exchange of requirements between tools.

During the import of a formatted document, like Word or RTF, styles are recognized and can be mapped to a text or heading object in the formal module. By default, all heading styles are mapped to a heading object, making importing quite easy since the structure of the original document is maintained. In a text section every piece of text ending with a return character is marked as a separate object. In addition, every cell of a table is marked as an individual object, but the structure of the table is maintained using a DOORS table.

During import, it is only possible to fill information into the main text attribute of an object. There is no option to convert certain parts of the imported document, like priorities; relations with other sections; people responsible; etc, into other attributes. The only exception is the storage of the meta-information containing the original formatting of a section in an attribute, so it can be used again during export.

Export
DOORS is able to export the objects in formal modules to the following file formats: Microsoft Word, Excel, Outlook, PowerPoint, Framemaker, CSV, TSV, RTF, HTML and plain text. Most of these formats will contain a simple plain text representation of the attributes enabled during the export of the module. The documents supporting rich formatting, like Word and RTF, will be combined with the information in the formatting attribute. When a template is used with the same formatting properties as the original document, the exported document will look almost the same. Smart objects like indexes will not be smart anymore after exporting, so these have to be regenerated in the new document.

It is possible to export attributes next to the requirements text, but these attributes do not have any formatting codes, since they were not imported from the original document. To be able to format the attributes too during export, DOORS provides a *paragraph style editor.* This makes it possible to add formatting information also to attributes, making the export of requirements to rich formatted documents much easier.

One other option that can be used for documentation is partitioning of the database. It is possible to export a section of the database into a so-called *away database*. This away database can then be copied anywhere without needing an active link with the home database. Based on the rights given to the away database it is not only possible to view the content of the database, but also to make changes that can be merged with the home database on return. When changes can be made in the away database, all changeable fields are locked in the home database to prevent integration problems later on.

The DOORS documentation options appear to be sufficient. It is possible to import and export various types of text documents, which should be enough to get most types of requirements in and from the tool. Since formatting options are stored with the

objects, exporting a document with close resemblance to the original document is easily achieved.

4.4.1.9 Tool integration

The tool needs to be able to interface with other tools in the product life cycle to make information stored in them visible and linkable.

DOORS is mainly targeted at the maintenance of requirements. The communication with external tools is only possible using the external links. Therefore, it totally depends on the integration options, especially the support for the URLs DOORS uses, of external tools and even then it is only possible to use links originating in DOORS.

Telelogic, the developer of DOORS, also provides a number of other software life cycle tools. One of these tools is Rhapsody, a development tool, which also supports limited traceability support. By using the Telelogic Gateway module, it is also possible to integrate Rhapsody with DOORS, Caliber and RequisitePro.

At the moment DOORS is targeted at the maintenance of various types of requirements. It can import and export requirements documents, maintain traces between the various objects in these requirements and it supports external links to other resources in the software lifecycle. Since these external resources cannot be represented in the tool, the traceability becomes poor after this point. To enlarge the reach of the tool, Telelogic has an extension for DOORS called Analyst, which adds UML 2.0 modeling support to DOORS. Analyst makes it possible to trace requirements to their design models with the same customizable traces that were only possible between requirements before. By interleaving requirements and models, a much larger part of the software lifecycle can be executed in the DOORS environment, allowing for much richer traceability. At the same time, the development becomes more model driven and reusable. The usage of traceability moves from product related to a more process related form. This way the tool moves from simple traceability to a more rich traceability as described by [RJ01].

So it looks like there is no real built-in support for external tools in DOORS. It is however possible to connect DOORS to other tools using TeleLogic's Gateway application.

4.4.1.10 Graphical representation traces

The tool must provide graphical user-friendly representation of traces.

There are a number of ways to show the links between objects in DOORS. The first and most basic graphical representation is the set of two arrows next to an object in a formal module. When the arrow pointing to the left is selected all direct incoming links, ordered by the formal module in which they reside, are shown. Selecting the arrow pointing to the right shows all direct outgoing links. When the links are being investigated it can be useful to shown these links all the time. Enabling one or more traceability columns in the formal module view, shows all direct in- or outgoing links for each object all the time. When more link information is needed, it is even possible to show several levels of indirect links to and from this object.

Grid view

In addition to these basic views in the formal modules, there are two more graphical views available in the link modules. The default view presented in a link module is the grid view as shown in Figure 16. The grid view shows a table with all visible objects from the source module in its rows and the visible objects from the target module in its columns. When there is a link between two objects the intersection of the row containing the source object and the column containing the target of this link, is marked with a contrasting color.

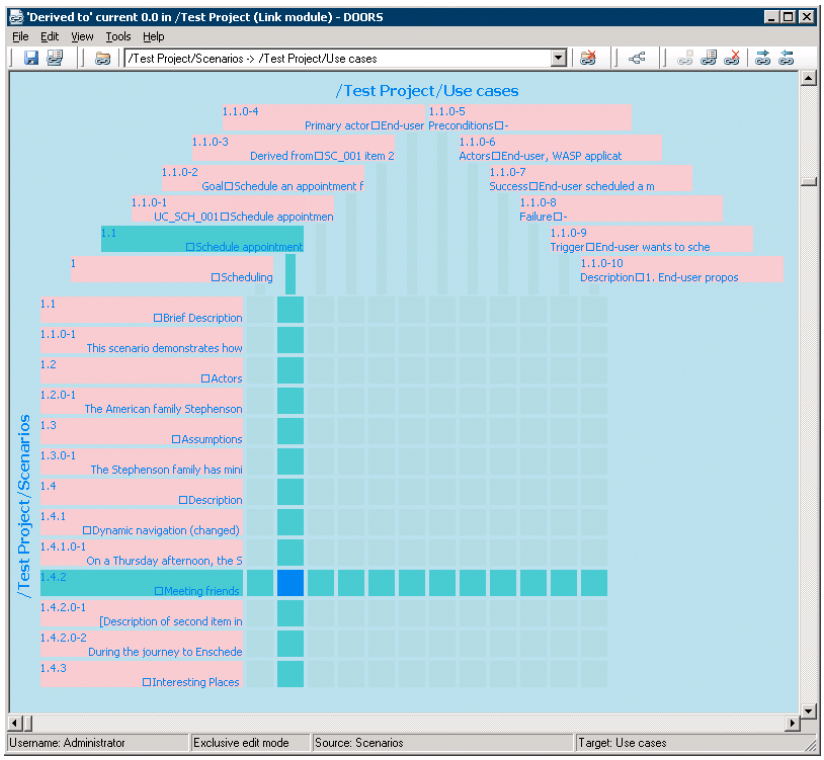

Figure 16: Grid view in DOORS

The objects shown in the rows and columns are a representation of the objects in the formal module views of both types of objects. The filter, level and sort order applied on the objects in the formal module view is also used in the matrix view. These filters make the grid view much more usable, since they allow the user to leave out any objects that are of no interest at this moment. This is a welcome help since the grid view shows only a very limited number of objects at once, making it very difficult to locate relevant links. There is however a potential problem when the links shown in the grid view are between objects in the same formal module. Since the filter will work on both the rows and columns, no links are shown when only objects with outgoing links are filtered, for example. When objects are filtered using the option to show only the first levels of the hierarchy, links from or to objects which are not shown, are not mentioned in any way at the first parent object that is visible. So there is no hierarchical transitivity. The filter options applied to the objects in a formal module can be saved in a so-called *view*. By enabling one of these views, the corresponding set of filters is applied again.

The limited number of objects visible at once is mainly caused by the fact that the number of rows and columns are both limited by the height of the window. Since most computer displays are wider then they are tall, in most cases the screen real

estate will not be used as optimal as possible. A normal window is normally not able to show more then fifteen objects on each side, which leads to a total lack of overview and makes it really hard to locate any (relevant) links as seen in Figure 16.

Graph mode

The second view available in link modules is the graph mode. For each linkset present in a link module, it will show an overview as shown in Figure 17. The left side of the view has a tree-based representation of the objects in the source module and the right side shows the objects and structure in the target module. The filters applied in the formal module are also applied to the objects in the graph view. The links between these two formal modules are represented by the lines in the middle. As with the grid view, the navigation of the graph view is very limited. The vertical scrollbar is used to navigate through the source objects and the horizontal scrollbar is used for the target objects. When there is a link between two objects, the line representing the link becomes black. However, since the objects are as small as in Figure 17, it is impossible to determine which objects are selected. Therefore, the only way to know which objects are linked is by switching back to the grid view.

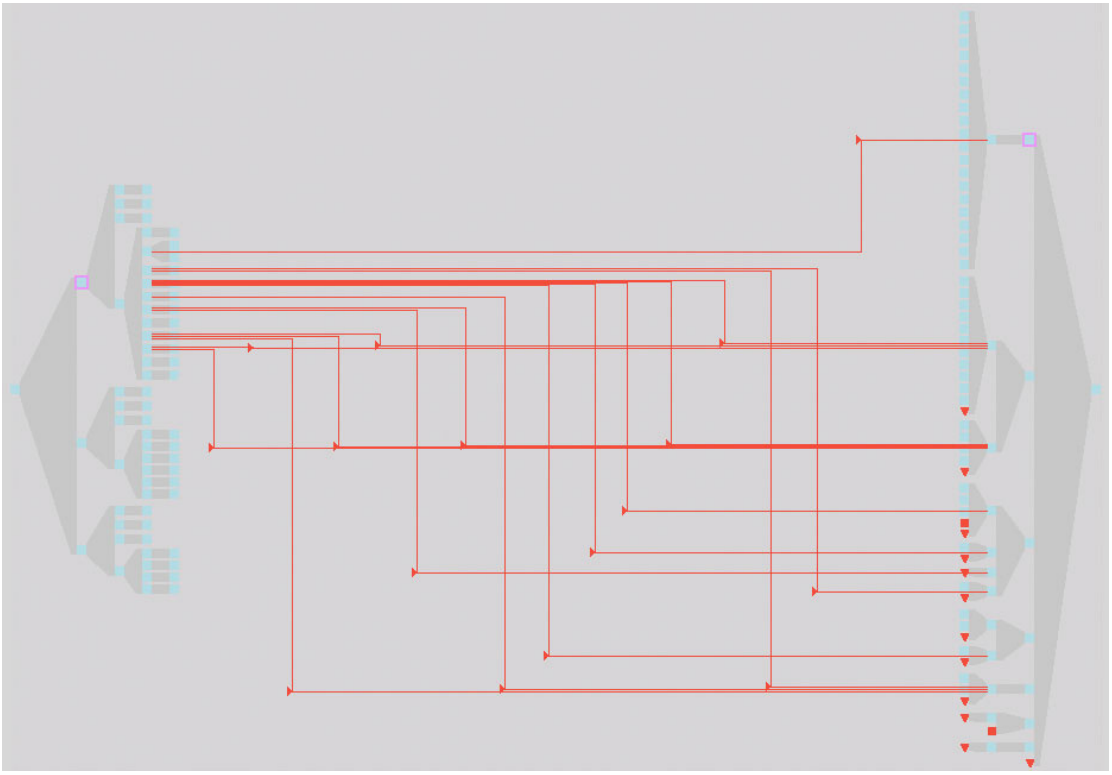

Figure 17: Graph view in DOORS

Next to the graphical representations in DOORS itself, Telelogic also offers an add-on called TraceLine. This add-on allows users to view and change requirements in a browser and this presentation looks more solid and usable then the options available in the core DOORS application. It is possible to specify views for different people in the organization and it is also possible to zoom in and out to show more or less detail in a fast way. It also adds a number of export options to HTML, Visio and Mindjet MindManager. Since this functionality has to be bought as an add-on for the original application, it has not been evaluated.

The various graphical representations of objects and links are mostly targeted at small sets of objects and links. Because of the simple and low-resolution graphics, the information density in the graphical views is really low. Together with the completely non-intuitive navigation in the link modules and a lack of options to configure the graphical presentation, makes it insufficient.

4.4.1.11 Coverage analysis support

The tool must be able to produce an overview of the artifacts that have no traces to previous or subsequent artifacts.

It is possible to add columns showing the in- or out-going links to and from a module. It is also possible to show objects with links to these modules, up to n levels. As with almost all filters, it is possible to show only links to and from objects in open modules or all modules.

To determine which objects do not have ingoing, outgoing or any links at all, a basic filter can be created. This filter can be used for simple direct link coverage analysis.

4.4.1.12 Impact analysis support

The tool must provide a way to perform an impact analysis to determine the effected artifacts.

DOORS does not really provide an automated option to perform a semantic analysis of links after an object has been changed to determine relevant effected artifacts. It does however offer a number of tools and filters to make the manual analysis of the links and objects affected by a change easier.

Suspect links
When an object is changed, its direct in- and outgoing links are marked suspect. This suspect status is only visible from the target or source object and not from the changed object itself. To make the suspect status visible a special column can be created showing various levels of information. At the lowest level only an icon is shown to indicate if there are any in- or outgoing suspect links for this particular object. Like most columns or filters related to links, this column also has the option to show this icon only when suspect links exist to modules which are opened at this moment or to all modules available in the current database. This option helps to filter only relevant links when a lot of links and modules are present.

When more information is required besides an icon, it is also possible to immediately show what objects in which modules cause the suspect status. It is even possible to add an overview of the changes made to this object; including the time and date they were performed, since the last clearance or baseline for this project was created. To make analysis even more useable, a filter can be created for each module showing only objects with in- or outgoing suspect links. This filter can however not be saved in a view.

When changes are made to the objects with in- or outgoing links, the suspect status has to be cleared from these objects manually. When a new baseline is created for the project and one or more of the objects still have a suspect link, these suspect links are maintained. The history describing the changes made to the objects is however

cleared and replaced by a message mentioning that the history from before the baseline can be found in the previous baseline.

Traceability columns
When the objects directly affected by a potential change are identified, their traces to other modules and objects must be inspected. The traceability columns introduced in 4.4.1.10 about the graphical representation can be used for this task. They allow for a recursive analysis with an option to specify the number of levels to show, if links to only open modules or all modules should be shown and which link modules should be used.

Change proposal system
When many people are working on the same project and many changes are made at any given moment, the impact analysis performed for a change, can become obsolete really fast when other changes are being implemented during or right after this particular impact analysis. To prevent that different groups are performing impact analysis or changes at the same time, DOORS includes a Change proposal system (CPS). This system allows a selected group to investigate and allow or deny change proposals. They can keep an overview of proposed changes and can determine the effect of these changes on the existing requirements and each other. Using the CPS in DOORS proposals are stored in a central place, so change proposals do not get lost, and automatic links are created from an to the originating object to make the investigation of proposals more convenient.

When a change is proposed, it will be investigated by this special group working on change proposals. Once they investigated the proposal, they can approve or reject it. The status of a change can be one of the following: *new, in review, approved, on hold, rejected, applied*. The links with the originating objects can be used to locate possible problems, for example when different proposals are related to the same object. The CPS has options to find these proposals, however when there are two proposals related to the same object and one of them is marked *approved* and the other *rejected*, the check still gives a warning. This can make this option less useful when more change proposals have been approved or rejected over time.

The approved changes can be processed by the team. When the changes are made, the status changes from *approved* to *applied*. Since the new values for the attributes mentioned in the proposal are already there, applying the changes is as easy as clicking *Apply* or *Apply all*.

DOORS has no option for semantic analysis based on the type of link and type of change that has to be applied. There are however enough tools to help with the manual analysis of the change impact.

4.4.2 Summary

During the implementation of the WASP 2.1 case study, the criteria created to evaluate the capabilities of the various requirements management tools have been described for DOORS in section 4.4.1. The outcome of these criteria is summarized in Table 6.

Table 6: Summary of criteria evaluation for DOORS

Criteria	Rating	Result
1. Information model support	+	It is possible to specify an information model with specific modules for each requirement type and modules for different link types between these requirements. Each module can have its own attributes and it is possible to put restrictions on which links can be used for which types of requirements.
2. Links detached from artifacts	+	Links are detached from both the original source artifact and the representation of a link in DOORS itself and stored in their own link modules.
3. Uniquely identifiable reference objects	+	The link with the original requirements document is lost after the requirements are imported into DOORS. This means the reference object has no unique identifier. However, the objects in DOORS, representing the original document, are uniquely identifiable, since they have their own unique identifier.
4. Traceability link types	+	DOORS supports normal and external links. Normal links can be stored in link modules that can represent a certain type. The semantics of these types are however not used by DOORS during analysis.
5. Assign attributes to links	+	A number of system attributes are available for links and custom attributes can be added when needed.
6. Automatic link detection	-	There is no option for automatic link detection and the manual ways to create links are cumbersome.
7. Enforce traceability links	□	No, there is no way to enforce the creation or updating of links after a change has been made. It is only possible to restrict the types of links that can be created between two formal modules.
8. Requirements documentation support	+	Yes, it is possible to import and export quite some formats (Microsoft Word, RTF, plain text, etc). During import the paragraph styles are saved and are used again during export, to create documents with close resemblance to the original.
9. Tool integration	□	No, but using another TeleLogic's tool it is possible to exchange data with other applications.
10. Graphical representation traces	□	Yes, but very limited. The graphical representations are only useable for very small sets of requirements, otherwise the lack of overview and good navigation options make it almost unusable.

11. Coverage analysis support	+	Yes, it is possible to filter objects with in- or outgoing links to selectable types of requirements and it is also possible to filter objects without any in- or outgoing links.
12. Impact analysis support	+	There is no way to use the semantics of the various link types to help with impact analysis. There are however e number of tools to analyze suspect links and traces up or down multiple levels. To streamline change management a change proposal system is available, allowing a group of users to investigate and regulate all change proposals and maintain an overview.

4.5 Evaluation of tools

After working with the four selected tools, to determine their performance for all criteria, a good impression of the tools is available. The ratings for all four tools have been combined into Table 7 and besides the necessary differences between the tools; there are also some criteria for which they perform the same.

The coverage analysis support is present in all four tools, for example. It is possible to check only for missing in- or outgoing links for all types, but it is also possible to apply filters to select only specific types of requirements, so coverage analysis of different types is possible. All tools also have their links detached from both original artifacts and the representations in the tool itself. Another thing they have in common is the support for automated link detection; none of them supports it. Some have limited semi-automated support or good tools for manual creation, but none of them supports any automated detection. The integration with other tools is also limited and when it is available, synchronizing links can be hard.

The first differences start to appear at the support of documentation. Importing and exporting documents is an important feature, since it makes it possible to present the requirements, for example, to the customer. RequisitePro works very close with the original document and synchronizes the content of the tool constantly with the content of the word document. Caliber on the other hand imports the original requirements document into the tool and looses the link with the document afterwards. By using a formatting language, the requirements can be exported again in a lot of different ways. TopTeam uses import and export methods comparable to Caliber. Importing is also based on the styles in rich text documents and exporting also uses little scripts to select and order requirements and Microsoft Word templates for formatting. DOORS also imports the original document. It supports most document formats of all tools and stores available formatting information during import. This information is used again during export to create documents with close resemblance to the original document.

The graphical representation of links also differs a bit between the tools. RequisitePro and Caliber both offer a grid-based and graph-based view, providing a good overview of the existing links. TopTeam offers an equal matrix view, but a more limited graph view to maintain a better overview when the number of requirements increases. DOORS supports the same two views, but implemented in a very limited way and therefore less usable then the other three.

The differences between the tools become bigger when the information model and the criteria related to links are compared. All tools are able to create different requirements types, each with their own attributes. TopTeam and DOORS are however the only tools supporting different link types and limitations on what links can be created between two requirement types. DOORS is however the only one supporting the creation of custom link types and assigning custom attributes to them.

In all the tools the impact analysis of a potential change has to be done manually with help of the graphical representations. There is support for indirect links in all tools except TopTeam Analyst, and in RequisitePro and Caliber only links in the same direction are used, so a manual check for effected artifacts is still needed. The big difference here is the DOORS Change Proposal System, allowing a dedicated group to perform the analysis and maintaining the overview.

In the end, there is not really one tool that is without doubt the best tool to use for impact analysis during a project. The biggest difference between the tools, and also the cause of the relatively high score, is the support for link types and attributes in DOORS. It does not really use the link support during analysis yet, but using the script language, it can be made more useful. The change proposal system is also a unique feature that can result in added value for larger projects.

Although it is definitely not the ideal traceability tool, after getting used to the interface and the way it works, DOORS supports most criteria and has the best basis for a trace management tool.

If impact analysis of less importance, the other tools also have their advantages. If the requirements are maintained in the tool itself, but also in a separate Word document, and both are updated simultaneously, RequisitePro's extended integration with Word can be a reason to select this tool. When requirements are often exported using complex export templates, CaliberRM has an advantage over the other tools. And when support for use cases in the tool itself is needed, TopTeam Analyst is highly recommended.

Table 7: Summary of evaluation for all three tools

Criteria	**Requisite**	**Caliber**	**TopTeam**	**DOORS**	**Weight**
1. Information model support	□	□	□	+	+
2. Links detached from artifacts	+	+	+	+	□
3. Uniquely identifiable reference objects	+	+	+	+	–
4. Traceability link types	□	–	□	+	+
5. Assign attributes to links	–	–	□	+	+
6. Automatic link detection	–	–	–	–	□
7. Enforce traceability links	–	–	□	□	–
8. Requirements documentation support	□	+	+	+	–
9. Tool integration	□	□	–	□	–
10. Graphical representation traces	+	+	+	□	+

11. Coverage analysis support	+	+	+	+	+
12. Impact analysis support	□	□	–	+	+

Part B: Change impact analysis performance

In Part B the performance of the impact analysis methods used by the requirements management tools is quantified.

To determine the performance, a method is needed to quantify the result of the impact analysis and a set of metrics are needed add value to these numbers. Chapter 5 describes a number of existing methods, but these seem to focus more on the quality of the change itself then the performance of the impact analysis method. To cope with this problem, a quantification method based on binary classifiers is proposed. Comparable methods are already used to determine the performance of medical tests, production lines and information retrieval methods and we show they can be used for impact analysis too.

In Chapter 6 the various methods discussed in Chapter 5 are applied to two of the four tools. Since the only big difference in impact analysis techniques between all four tools is the usage of only the direct links to mark artifacts suspect or the usage of transitive closure so artifacts related through indirect links are also marked suspect. TopTeam Analyst and DOORS used the first method, while RequisitePro and CaliberRM use the second method. As a result, only the performance of the impact analysis methods in DOORS and RequisitePro are tested.

Since the WASP case, used in Part A, contains no change scenarios, another case is used to determine the impact analysis performance of the tools: the Course Management System. This case contains three change scenarios that can be used to test the tools and performance methods. After implementing the changes in both tools and gathering the numbers for the various metrics, the results show no surprises. Change scenarios affecting several levels of requirements are better predicted by the tools with transitive closure based impact analysis methods, while change scenarios effecting only requirements on the same level are better predicted by the impact analysis methods that follow only direct links.

The methods to determine the performance show a bigger difference. The original methods were indeed targeted at the performance of the change itself and not the impact analysis method used to inspect them. The method based on binary classifiers does show the performance of the actual impact analysis method. A result that has been confirmed by other recent research.

5 Metrics for impact analysis performance quantification

``Change does not necessarily assure progress,
but progress implacably requires change.''

Henry S. Commager, American historian, 1902-1998

In this chapter an overview of current quantification methods is presented and the problems with these methods are discussed. Based on these problems an existing test evaluation technique is introduced for tests with a binary output like impact analysis methods. These metrics based on binary classifiers will be compared with the existing impact analysis methods to show they work without restricting preconditions and at the same time tell more about the performance of an impact analysis method. The **performance** of an impact analysis method will be defined as the ability to identify artifacts that have to be changed as a result of an initial set of changes, with as little false alarms and misses as possible.

5.1 Related work

In the last 20 years several methods to quantify the performance of impact analysis have been proposed. Most of these methods are based on sets with artifacts that have or have not been changed and artifacts that have or have not been identified by the tools. To understand the operations performed on these sets, Section 5.1.1 contains an elementary introduction into set theory. Section 5.1.2 describes a number of methods that have been used in the past to quantify the performance of impact analysis methods. There are however some problems with these methods, disqualifying them for determining impact analysis performance. An alternative method, based on binary classifiers, that has proven itself already in other fields of work is introduced in Section 5.1.3. Section 5.1.4 discusses the metrics that can be used based on binary classifiers. The final section, Section 5.1.5, shows some of the latest methods to determine an impact analysis methods performance, based on binary classifiers and their metrics.

5.1.1 Introduction into set theory

Set theory describes the mathematics that can be applied to collections of elements. These elements can be abstract numbers, real objects like products made in a factory, people, etc. The order of elements in a set is not relevant.

A set, called A, with n elements, called a_1, a_2, …, a_{n-1}, a_n, is denoted as $A = \{a_1, a_2, \ldots, a_{n-1}, a_n\}$. The number of elements in a set is called its cardinality, so the cardinality $|A|$ of set A is n. If an element is part of a set this is symbolized by $\in$. Element a_2 is part of set A: $a_2 \in A$. Two sets are equal when they contain the same elements: A=B. A set with no elements and a cardinality of zero is called an empty set and symbolized by $\varnothing$.

When multiple sets are available, they can be compared and combined into new sets. If all elements of set B are also part of set A and A and B are not equal, then B is a

subset of A: $B \subset A$. The case where B is a subset of A or both sets are equal is symbolized by $B \subseteq A$.

There are three ways to combine two sets:

- When all elements in set A and B are combined in a new set, this is called a union. The union of A and B is denoted as $A \cup B$.
- Selecting only the elements present in both set A and set B, is called an intersection. The intersection of A and B is denoted as $A \cap B$.
- Selecting all elements in set B that are not also part of set A, is called the relative complement of A in B. The relative complement of A in B is denoted as B-A.

5.1.2 Current methods to determine impact analysis performance

Over time a number of methods to determine the performance of impact analysis techniques have been proposed. This section will describe four of them. Three methods are based on sets originally defined by Arnold and Bohner [AB93]. The last method is based on sets defined by Cleland-Huang [CHC03].

5.1.2.1 Arnold and Bohner sets

Arnold and Bohner introduced their proposal to determine the performance of impact analysis for the first time in their framework for the comparison of impact analysis methods in 1993 [AB93]. In 2002 an update was provided with additional metrics. Fasolino and Visaggio propose their own system to measure the performance of impact analysis methods, but these are based on the initial sets introduced by Arnold and Bohner [FV99].

Bohner and Arnold 1993

In 1993 Bohner and Arnold present their first work on a framework for the comparison of impact analysis methods [AB93]. The framework consists of three sections. The first section describes the impact analysis application and looks at how is the impact analysis method is used to accomplish impact analysis and examine the features offered at the methods interface. It can be compared to our work on the criteria used to evaluate requirements management tools in section 3.2.1. The second section describes the impact analysis parts and examines the nature of the internal parts and methods used to perform the impact analysis, like the description of the information model options in section 3.2.1. The third and last section described the effectiveness of the impact analysis method. It examines the properties of the search results of the IA approach, to see if it accomplishes the goals of IA.

For the evaluation of the IA approach the framework compares the following sets:

- *Universe*: all possible artifacts;
- *System*: a bounding set of objects in which the IA will take place (all artifacts present in the system);
- *Start Impact Set* (SIS): all impacted artifacts identified at the start of the change;
- *Estimated Impact Set* (EIS): all additional impacted artifacts found by the IA approach (including the SIS);
- *Actual Impact Set* (AIS): the artifacts actually changed because of the change.

Using these five sets, a property called *Effectiveness* could be determined for an IA method. It contained four elements:

- *SIS and EIS*: What trend is observed in the relative size of the SIS and EIS when the approach is applied to typical problems? They would like the EIS to be as close as possible to the SIS: |SIS| / |EIS| = 1, or nearly 1.
- *EIS and System*: What trend is observed in the relative size of the EIS and System when the approach is applied to typical problems? They would like the EIS to be much smaller than the System: |EIS| / |System| < N, where N is some small tolerance level.
- *EIS and AIS*: What trend is observed in the relative size of the EIS and AIS when the approach is applied to typical problems? They would like the EIS to contain the AIS and the AIS to equal to or smaller than the EIS: |EIS| / |AIS| = l, or nearly 1.
- *Granularity*: What is the relative granularity of the artifact object model vs. the interface object model? They would like the granularities to match, if possible.

In the end, five IA approaches are compared using the framework. Since only the approaches are investigated and not a specific case study, the effectiveness based on the sets cannot be used. Therefore, only the granularity is evaluated.

Bohner 2002
In 2002 Bohner presented an addition to his previous work [Boh02]. He uses a number of the impact sets presented earlier and one of them is renamed:

- Starting Impact Set (SIS): initial set of objects thought to be affected by a change. Normally determined when examining the change specification.
- Candidate Impact Set (CIS): set of objects estimated to be affected. Produced while conducting IA. This was called the estimated impact set (EIS) before.
- Actual Impact Set (AIS): set of objects actually modified. Not necessarily unique, as a change can be implemented in several ways.

Two new impact sets are introduced to make two groups of impacted artifacts more explicit:

- Discovered Impact Set (DIS): set representing an under-estimate of impacts; artifacts identified as part of the implementation of the change, so not present in the CIS but present in the AIS
- False Positive Impact Set (FPIS): set representing the over-estimate of impacts in the analysis; all artifacts identified by the CIS, but not actually changed in the AIS.

These two sets are used to obtain the accuracy .This *Accuracy* is expressed using the *Error* = (|DIS| + |FPIS|) / |CIS|. The target of impact analysis is to get a CIS as close to the AIS as possible, so the size of both DIS and FPIS should be as close to zero as possible. This leads to an error close to zero. The error can be anything in the range [0, ∞ >.

Fasolino and Visaggio
Fasolino and Visaggio propose their own system to measure the performance of impact analysis methods [FV99]. Their approach is based on two dimensions. Adequacy is the capability of the approach to identify potentially affected artifacts

that contain at least all artifacts that will be changed in the end. Effectiveness is the quality of the impact analysis, the ability to produce results that are useful for the maintainer.

Their approach is also based on sets containing artifacts. These sets are used for the analysis:

- System: all artifacts present in the system;
- Primary Impact Set (PIS): the set of affected artifacts determined by the maintainer;
- Secondary Impact Set (SIS): the set of artifacts that is returned by the impact analysis with the PIS as a start set;
- Estimated Impact Set (EIS): PIS and SIS combined;
- Actual Impact Set (AIS): the set of artifact changed because of actually implementing the change.

Adequacy is measured using a metric called inclusiveness, a binary value being one when the EIS contains the AIS and zero when this is not the case. According to Fasolino any results from an impact approach are useless when the inclusiveness is zero, since the maintainer is not receiving all artifacts that need investigation. This means all artifacts have to be investigated anyway, making the impact analysis pointless.

Once the adequacy has been confirmed, the quality of the set of artifacts returned by the impact analysis approach can be investigated. Fasolino uses three metrics to determine the quality: Ripple-sensitivity, Sharpness and Adherence.

Ripple-sensitivity is the characteristic describing the relation between the PIS and the SIS. When the SIS is much larger then the PIS, the approach is sensitive for the ripple effect. This sensitivity is expressed using the amplification ratio metric: |SIS| / |PIS|.

The *sharpness* of the approach is its ability to not include all artifacts in the system, but only the artifacts that are probably affected by the change. It is expressed using the change rate: |EIS| / |System|. Since the EIS is always included by the system, this value is always between 0 and 1. The smaller the better (since the inclusiveness is one to start with, meaning EIS contains the AIS).

The *adherence* of the approach is a more detailed version of the inclusiveness. It describes the relative size difference between the EIS and AIS. When these two sets are close, fewer artifacts not affected by the change have to be investigated by the maintainer, making analysis more efficient. The adherence is expressed using the S-ratio, showing the relation between the EIS and AIS: |AIS| / |EIS|. The result is best when this ratio is one.

5.1.2.2 Cleland-Huang sets

Cleland-Huang also presents a number of scenarios for the implementation of changes based on sets. The sets introduced are the following [CHC03]:

- *Impacted artifacts*: The (theoretical) set of impacted artifacts as a result of proposed change C on artifact a.

- *Identified artifacts*: The set of artifacts actually identified as impacted by the proposed change C upon artifact a. Unless a developer manually identifies artifacts as impacted outside the traceability schema, Identified(C,a) should be a subset of Linked(a).
- *Updated artifacts*: The set of artifacts updated as a result of implementing change C upon artifact a, after moment t in time after which all artifacts and links related to change C are updated: Updated(C,a,t) <= Identified(C,a)

Impacted(C,a) is a theoretical set of all impacted artifacts as a result of change C upon artifact a. Identified(C,a) is the actual set of artifacts that will be impacted by change C upon artifact a (based on the traceability schema and the input from the developers). Updated(C,a) is the set of actually updated artifacts.

The four scenarios presented by Cleland are the following:

- *No errors*: the impacted, identified and updated sets are the same.
- *Identification errors*: artifacts that are impacted are not identified. This could be the result of broken or missing traces or an incorrect interpretation of the source or target artifact: Impacted(C,a) > Identified(C,a). –> not found
- *Update errors*: occur when impacted artifacts and their links are not updated to reflect the implemented change. Impacted(C,a) > Updated(C,a,t). <– not implemented
- *Inclusion errors*: artifacts are identified as being impacted while they are not. So the intersection of Identified(C,a) and not Impacted(C,a) is not an empty set.

So compared to the framework presented by Bohner and Arnold, SIS is a set of change C and artifact a pairs, EIS is a concatenation of all Identified(C,a) sets for each pair in SIS and AIS is a concatenation of all Updated(C,a,t) sets for each pair in SIS.

Although according to the framework EIS is only based on SIS, while Identified(C,a) can also contain manually added artifacts. In the framework it is also possible that |EIS| < |AIS|, but the definition of Updated(C,a,t) states Updated is a subset of Identified. This is comparable to the inclusiveness as presented by Fasolino.

5.1.3 Binary classification

A binary classifier is a test returning a binary result: positive or negative, true or false, etc. A lot of tests can be classified as binary classifiers, as these examples show:

- Medical tests: when a test for a specific disease is applied, it should return positive when someone has this disease and negative when this is not the case.
- Quality control: the test should decide if a product is good enough to be sold.
- Information retrieval: the test should determine if a page should be returned or not as result of a query.

The performance of these tests is most of the time not perfect; meaning the set of elements retrieved by the test is not the same as the set of relevant elements. Since the test is binary, any element inspected by the test can be placed in one of four categories, as shown in Table 8. An element that is retrieved by the test and is relevant, is called a true positive (TP) or hit. For example an actually ill person that is identified by a medical test as being ill. An element that is not retrieved by the test

and is not relevant according to the test, is called a true negative (TN) or correct rejection. For example a person that is not actually ill and also identified by the test as not being ill.

For a perfect test all elements would belong to one of these two sets. Most tests are however not perfect, leading to results that do not fit in one of these sets. An element that is retrieved by the test, but is not actually relevant, is called a false positive (FP), Type I error or false alarm. For example a person that is not actually ill, is being identified as ill by the test. In the other scenario an element that is relevant is not retrieved by the test, called a false negative (FN), Type II error or miss. For example an actually ill person who is not identified as being ill by the test.

Figure 18 shows these four sets in a Venn diagram. As seen in the diagram all four sets are disjoint, so $TP \cap FP \cap FN \cap TN = \varnothing$, $TP \cup FP \cup FN \cup TN = \{\text{all elements}\}$, $TP \cup FP = \text{Retrieved}$ and $TP \cup FN = \text{Relevant}$.

Table 8: Confusion matrix showing four results of binary classifier

		Condition or relevance (as determined by "Gold standard")	
		Relevant	***Nonrelevant***
Test outcome	***Retrieved***	**True Positive (TP)** Hit	**False Positive (FP)** False alarm
	Not retrieved	**False Negative (FN)** Miss	**True Negative (TN)** Correct rejection

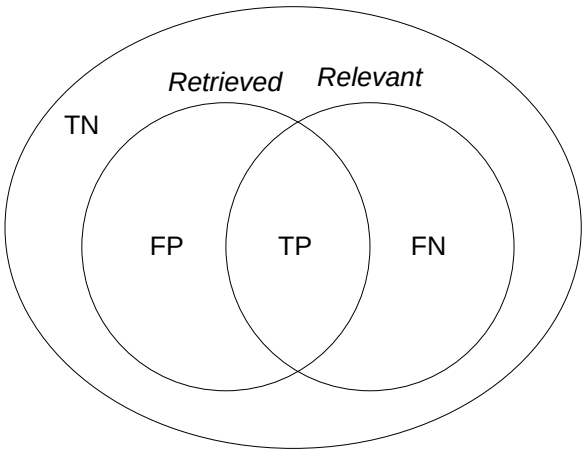

Figure 18: Venn diagram showing four result sets of a binary classifier

5.1.4 Metrics based on binary classification

As said before, most nontrivial tests do not have empty error sets and are therefore not perfect. To determine the performance of a binary classifier, a number of expressions or metrics based on the four results sets have been created. Fawcett gathered a number of these expressions presented over time to analyze the performance of binary classifiers [Faw04]:

- Recall, also known as true positive rate (TPR), hit rate or sensitivity
  Recall = TP / (TP + FN)

- Precision, also known as positive predictive value (PPV)
  Precision = TP / (TP + FP)
- Fall-out, also known as false positive rate (FPR) or false alarm rate
  Fall-out = FP / (FP + TN)
- Specificity (SPC) or True Negative Rate
  SPC = TN / (FP + TN) = 1 – FPR
- Accuracy (ACC)
  ACC = (TP + TN) / (TP + FP + FN + TN)
- F-score, also known as F1-score or F-measure
  F = 2 · (Precision · Recall) / (Precision + Recall)

5.1.4.1 Definitions expressions

The expressions used most are described in some more detail.

Recall shows the ratio between the number of true positives and the total amount of relevant results, containing both the true positives and false negatives. It shows the proportion of relevant elements that was also retrieved by the test. A result of one means all relevant elements were retrieved by the test, while a score of zero means not one of the relevant elements was retrieved.

The result of this metric gives a good indication of a tests performance when it is important to identify all relevant elements. It is however easy to obtain a score of one, by just returning all elements. This way all relevant elements are always included, but all nonrelevant elements are included too. So it also shows Recall should always be presented together with another metric, like Precision or Fall-out.

In case of a highly contagious disease that can be easily cured, it is very important all ill people are identified and treated. In this case the amount of persons not actually being ill, but being identified and cured anyway, is less of a problem then not identifying ill people. Therefore, this is an example where only the Recall describes the performance of the test quite good.

Precision shows the ratio between the number of true positives and the total amount of retrieved results, containing both the true positives and false positives. It shows the proportion of relevant elements in the set of elements retrieved by the test. A result of one means all elements in the retrieved set are relevant, while a score of zero means not one of the elements in the retrieved set is relevant.

Where the result of Recall gives a good indication of the proportion of relevant items retrieved, the result of Precision gives a good indication of a tests performance when it is important to retrieve as little as possible nonrelevant elements. The opposite of Precision is the **False Discovery Rate** (FDR) describing the proportion of nonrelevant elements in the set of elements retrieved by the test. Just as with Recall there are some ways to easily obtain high scores. When one and only one relevant result is returned, the Precision will be one, suggesting a high precision test. So Precision should also be accompanied by the Recall results of a test, to get a complete image of its performance.

In information retrieval, and web searching in particular, retrieving all relevant elements is of little importance, but retrieving as little nonrelevant elements as

possible is important. When people look for information, they do not need thousands of relevant results with even more nonrelevant results, but only a limited number or relevant results with as little as possible nonrelevant results. So this is an example where only the Precision describes the performance of the test quite good.

Fall-out shows the ratio between the number of false positives and the total amount of nonrelevant results, containing false positives and true negatives. It shows the proportion of nonrelevant elements that was incorrectly retrieved by the test, also called false alarms. A result of one means all nonrelevant results were retrieved by the test, meaning the tests performance is not good, while a score of zero means not one of the nonrelevant elements was retrieved by the test, meaning good test performance. Its opposite is **Specificity**, showing the proportion of nonrelevant elements that was not retrieved by the test and thus correctly rejected.

In quality control for critical devices it is of utter importance these devices perform as specified. So the production process is optimized to produce as little defects as possible and of all broken devices, none is allowed to be shipped. The proportion of shipped devices that is correct (Precision), contains too many correct devices compared to defects. The proportion of correct devices that is being shipped (Recall), says nothing about shipped broken devices at all. So the proportion of broken devices that is being shipped is what is needed, and this value should be as small as possible. Fall-out describes just this ratio and describes the performance of this test much better then Precision or Recall.

Accuracy shows the ratio between all correctly identified elements, both true positives and true negatives, and the total number of elements, the total of all four result sets. A result of one means the test identified all elements correctly, while a result of zero means not one of the elements was identified correctly.

A result of zero is only possible when all relevant elements are retrieved as nonrelevant and all nonrelevant elements are retrieved as relevant. This makes the test a perfect qualifier, only the labels are inverted, relevant should be nonrelevant and visa versa. When the labels are corrected the score will be one. So the worst result for Accuracy is not one or zero, but when the classifiers returns random results at 0.5. There is also a problem with the Accuracy, resulting in better Accuracy scores while the other metrics show worse scores. This is the so-called Accuracy paradox, described in Section 5.1.4.3.

F-score as described by Fawcett is the harmonic mean of Precision and Recall. It is a weighted average of these two metrics where a result of zero indicates a poor test result, while a result of one means the result of the test was perfect. This basic F-score weights both Precision and Recall the same. When one of these metrics should have more effect, the $F_ß$-score can be used to shift the weight towards one of the metrics [Rijs79]. The value of the Recall is β times as much important as Precision.

$$F_ß = (1 + ß^2) \cdot (\text{Precision} \cdot \text{Recall}) / (ß^2 \cdot \text{Precision} + \text{Recall})$$

The basic F-score returns the same result as the F_1-score and is therefore also referred to as the F1-score.

5.1.4.2 Visualize and compare metrics

Most of the metrics defined in the previous section do not give a clear view of an impact analysis methods performance on their own. The values for Precision and Fall-out alone can not be used to determine a methods performance. They both need Recall for the total picture. Since a combination of two values is needed, it is possible to plot these combinations in a two-dimensional plot for a quick overview of a tools performance characteristics.

ROC plot

The ROC (Receiver Operating Characteristic) plot is one of these plots to analyze the performance of different binary classifiers or different configurations of one binary classifier. An example of a ROC plot is given in Figure 19. A ROC plot shows how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples. Therefore, a result in the top left corner, means there were no false alarms and all relevant elements were identified by the test, so this is the best result possible. Results on the ascending diagonal have the same amount of true results as false alarms, meaning this method has the same performance as a random guess. So the closer a result is to (0,1) and the further away from the diagonal, the more reliable the predictive capabilities of a method or configuration are.

Results below the diagonal can also have good predictive capabilities. A result close to the bottom right corner means the number of true positives and true negatives is close to zero. So almost all true positives have been identified as negatives and all true negatives have been identified as positive. Inverting the labels of the test results, turns it into a classifier with an almost perfect performance. So inverting the labels of the test results, causes the result points to be mirrored in the in the ascending diagonal, moving them from the area bellow the pure chance line to the area above it.

Instead of a ROC plot with only one data point per classifier, it is also possible to create a ROC curve. When a binary classifier is based on a threshold to determine if an element is retrieved or not, it is possible to perform the test with different thresholds. Each threshold results in a data point and by connecting these data points the ROC curve is created. A curve always starts in the lower left corner and ends in the top right corner. The curve of a good classifier gets, just as the ROC plot, close to the top left corner, while the curve of a random classifier stays close to the ascending diagonal. The area under this curve, or AUC, therefore determines the performance of a classifier.

Since the Fall-out is determined using only negative elements and the Recall is determined using only positive elements, the ratio of positive elements to negative elements has no effect on the ROC plot or curve. This makes it possible to compare the results of data sets with different ratios.

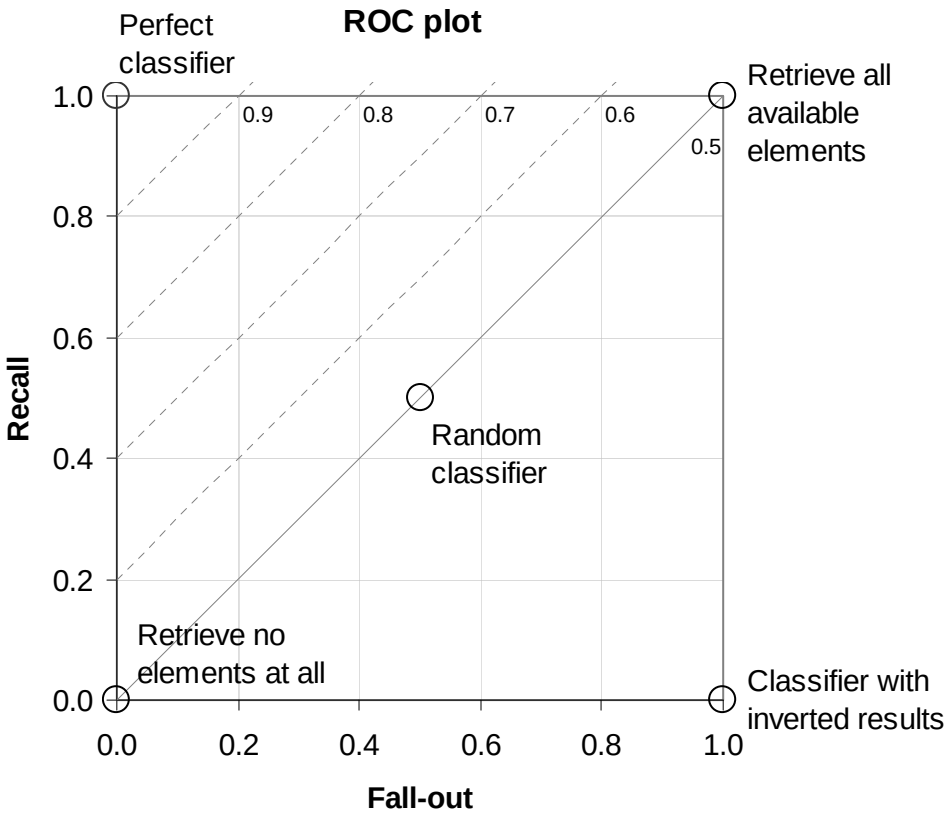

Figure 19: ROC plot template

PR plot

Precision-recall plots are another way to visualize the information in a confusion matrix to determine the performance of the classifier. An example of a Precision-Recall plot, or PR plot, is given in Figure 20. An ideal result would have a Recall and Precision of one, meaning there are no false alarms or misses and the test result was perfect. So the closer a result is to the upper right corner, the better the binary classifier. When a classifier retrieves all available elements, its Recall will be high, since all relevant artifacts have been identified, but the Precision will be low, since there will be a lot of false alarms. Such a result will be located close to the lower right corner. When a method returns only one relevant artifact, its Precision will be high, since there are no false alarms, but the Recall will be low, since there are probably a number of misses. A result like this will be located close to the top left corner of the plot.

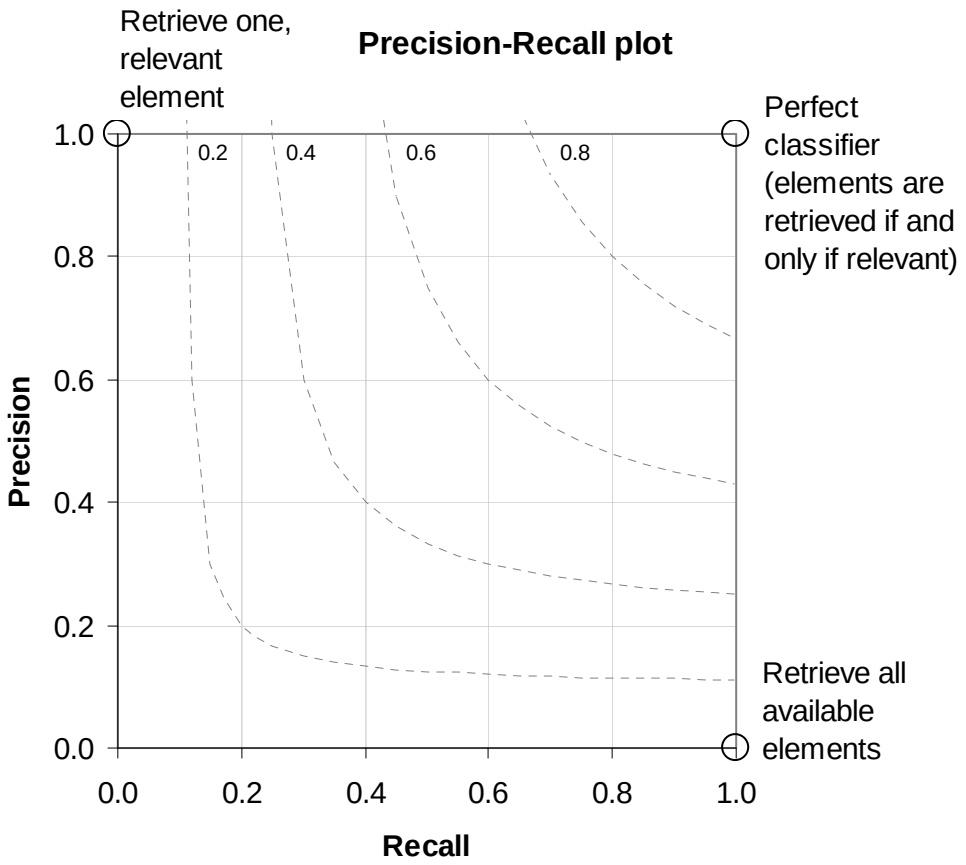

Figure 20: Precision-Recall plot template

Isometrics

Both plots have an ideal spot and the rest of the plot is reserved for non-perfect results. To be able to determine which of two results is best, some sort of isometric is needed for each plot to show which results have equal performance. For both plots a number of options are available [Fla04].

The ROC plot can use Precision, F-score, Accuracy or AUC (area under curve). The first three isometrics are based on additional metrics besides the two key metrics for the ROC plot: Recall and Fall-out. This means the isometrics depend on the ratio of positive and negative elements. This means it is not possible to plot the results of tests using different ratios in the same plot and use the metrics to compare them. Since the AUC is only based on Recall and Fall-out, it does not have this problem.

The PR plot can use the F-score, Accuracy or the AUC. When Accuracy is expressed using Precision and Recall a variable representing the ratio between relevant and irrelevant elements is introduced [Alv02]. This means it is not possible to plot results with different ratios in one plot. Both the F-score and AUC can be used as isometrics when the results of sets with different ratios of positive and negative elements have to be compared.

The ROC plot in this section already shows what the isometrics look like when AUC is used. The numbers along the top indicate the actual area under the curve. The PR plot is provided with the F-score as isometric. The values along the top show the values for the F-score.

Differences between ROC and PR plots
Although the data points in ROC and PR plots are based on the same confusion matrix, there are a number of differences between them [DG06]. First of all the PR plot gives a more objective image when the ratio of relevant and irrelevant artifacts is highly skewed. In addition, a data point or curve dominates in ROC space if and only if it also dominates in PR space. In ROC space it is correct to interpolate between data points using a linear line, in PR space this is not allowed. Finally ROC space can be used to determine if the output of classifier is random or not, by performing multiple tests with a classifier and measure their distance to the ascending diagonal.

5.1.4.3 Accuracy paradox

The Accuracy for binary classifiers can be misleading in some cases. The Accuracy compares the correctly identified elements with the total number of elements:

Accuracy = (TP + TN) / (TP + FP + FN + TN)

When the amount of true positives is smaller then the amount of false positives for a particular test, the Accuracy increases when this test is discarded and replaced by a test where the test outcome for all elements is negative. The same holds when the amount of true negatives is smaller then the number of false negatives and the test outcome for all elements becomes positive.

Table 9 shows the confusion matrix for the impact analysis of the modification of Req9 in DOORS as described in Section 6.2. The impact analysis returned six elements of which one was actually relevant. The Accuracy for this impact analysis and change is therefore 11/16.

Table 10 shows what happens when all elements are just marked as negative; meaning none of the artifacts should be changed. In this case only one artifact is missed and the Accuracy for this ‘impact analysis’ is 15/16.

So by using no test at all, the Accuracy is increased. In this case this is logical, since the remaining classifier participating in the Accuracy (true positive or true negative) is increased with a larger number (false negative or false positive) then the true negative or true positive was originally.

This shows Accuracy is not a reliable metric to determine a tests performance. The combination of Precision and Recall is much more reliable as the result of both these metrics would be zero when the actual test is replaced by a test returning always positive or negative as a result.

Table 9: Accuracy paradox, result normal test

		Condition	
		Relevant	**Not Relevant**
Test outcome	**Positive**	1	5
	Negative	0	10

Table 10: Accuracy paradox, result negative test

		Condition	
		Relevant	Not Relevant
Test outcome	Positive	0	0
	Negative	1	15

5.1.5 Current methods based on binary classifiers

When looking for literature on impacts analysis using binary classifiers and the metrics derived from them, a number of publications have been found.

Lindvall and Sandahl investigated a manual impact analysis method used during a real project taking four years [LS98]. They link requirements to classes and after a change had been performed, not only the affected artifacts are recorded, but also the traces used to get to these artifacts, called back tracing. In the end all of the thirty predicted classes were actually changed based on their relation with one or more requirements. When inspecting the individual changes, some predicted classes were not actually changed as a result of the prediction of the particular change, which can be described as a false positive. At the same time there were also changes where classes were changed that had not been predicted, which can be described as false negatives. During the analysis of the results they introduce a metric called Correctness with the same behavior as Precision. So they have identified all possible options of a binary classifier, but have not identified it as such, missing out on most of the metrics based on them.

A year later Cimitile, Fasolino and Visaggio refer to this same paper to describe the difference between the set of elements used to start an impact analysis (SIS) and the result set of the impact analysis (EIS) [CFV99]. They stick however with their own limited metrics describing more of the change itself then the performance of the impact analysis method they are after.

Ibrahim, Idris, Deraman and Selangor are also looking for a way to evaluate impact analysis methods as part of their software traceability validation for change impact analysis of object oriented software [IID06]. They refer to the work by Lindvall, but also Bohner and Fasolino. In the end they choose the more limited method presented by Fasolino.

Hattori Guerrero Figueiredo Brunet Damasio 2008 [HGF08]
Name of paper: On the Precision and Accuracy of Impact Analysis Techniques
Hattori, Guerrero, Figueiredo, Brunet and Damasio measure both Precision and Recall to cover both false positives and false negatives. Something not done by any of the previous references. They use three simple code-slicing applications to analyze a number of changes from a CVS system and their conclusion is that limiting the number of trace levels increases Precision, but decreases Recall even more. In the title of their publication they mention accuracy, but since they are using Precision and Recall, this is clearly not the Accuracy based on binary classifiers.

5.2 Binary classifiers for impact analysis performance

Impact analysis can be seen as a binary classifier. For each artifact in the system it determines if it is suspect or not. This section will show how the sets presented in

Section 5.1.2 can be mapped to a binary classifier so they can be used in the binary classifier based metrics. Once the mapping is complete, the various methods are compared and a final set of definitions for quantifying the performance of impact analysis methods is presented.

5.2.1 Evaluation of Impact analysis methods

An impact analysis method can also be seen as a binary classifier, determining for each artifact in a system if it has to be changed or not as result of a proposed change. To compare this impact analysis method with other methods, the four qualifications and the associated analysis expressions can be used.

Using the quantification sets as presented by Bohner, impact analysis can be classified as shown in Figure 21. In this case the system is a set containing all artifacts, EIS is the set containing the result of the impact analysis binary classifier: all artifacts marked suspect and AIS contains the artifacts actually changed. Table 11 shows the classifiers when applied to impact analysis methods.

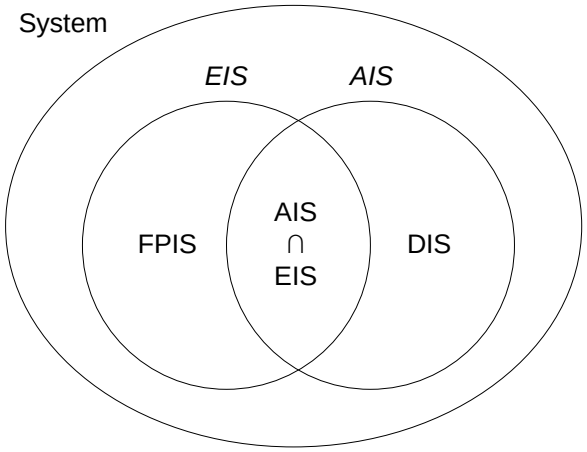

Figure 21: Venn diagram with subsets as described by Bohner

Table 11: Binary classifier results for impact analysis methods

		Result after implementation	
		Actually changed	***Not actually changed***
Result impact analysis method	***Marked suspect***	AIS ∩ EIS	FPIS
	Not marked suspect	DIS	System – AIS – EIS

Now the impact analysis sets have been applied to the confusion matrix, it is also possible to apply them to the metrics based on binary classifiers:
Precision: TP / TP+FP = |AIS ∩ EIS| / |AIS ∩ EIS| + |FPIS| = **|AIS ∩ EIS| / |EIS|**
Recall: TP / TP+FN = |AIS ∩ EIS| / |AIS ∩ EIS| + |DIS| = **|AIS ∩ EIS| / |AIS|**
Fall-out: FP / FP+TN = |FPIS| / |FPIS| + |System – AIS – EIS| = **|FPIS| / |System – AIS|**

In the next section these binary classifier-based metrics will be compared with the old metrics based on the original sets provided by Arnold and Bohner.

5.2.2 Mapping original definitions to binary classifiers

Table 12 shows our first attempt to create a mapping from the original definitions to the results of a binary classifier. This mapping can be used to see if the original impact analysis sets can be mapped to the four sets of the confusion matrix. The rows show the original definitions, the columns show the various binary classifiers and the cells show the relation between the definition and the various results of the binary classifier. When multiple cells in a row show a relation, it means the result sets of each relation are combined, like a union. The gray rows and columns show expressions without a mapping.

For example the System set contains all elements in the system. Using the binary classifier result sets, this means all four sets: System = TP ∪ FP ∪ FN ∪ TN. The Estimated Impact Set contains all results of the impact analysis method, so all elements identified as positive by the test: EIS = TP ∪ FP**.** The Starting Impact Set (SIS) contains a subset of the EIS, so it can contain both elements from TP and FP: SIS ⊆ TP ∪ FP.

Table 12: Mapping current definitions on binary classifier results

Element sets		**Binary classifier sets**			
Code	**Name**	TP	FP	FN	TN
A2	System	=	=	=	=
A3	SIS	⊆	⊆		
A4	EIS	=	=		
A5	AIS	=		=	
B1	System	=	=	=	=
B2	PIS	⊆	⊆		
B3	SIS	⊆	⊆		
B4	EIS	=	=		
B5	AIS	=		=	
C1	System	=	=	=	=
C2	SIS	⊆	⊆		
C3	CIS	=	=		
C4	AIS	=		=	
C5	FPIS		=		
C6	DIS			=	
D1	Impacted				
D2	Identified	=	=		
D3	Updated	=		=	

The result shows most sets used by the original definitions can be expressed using the results of a binary classifier. Only the starting sets and secondary impact set have no direct mapping, but this is no problem since these sets are only used for metrics describing the change itself instead of the performance of the impact analysis method used. The Impacted set cannot be mapped, since its definition is not clear.

Table 13 shows the first attempt to map the original expressions to expressions based on the binary classifier. The rows show all original metrics and the columns the metrics based on the binary classifier. The first four columns in Table 13 are the same as the columns in Table 12, since some analysis expressions only return one of the four results of the binary classifier. Recall, Precision, Fall-out and Accuracy are the four metrics based on the results of the binary classifier as presented in other research. The gray rows and columns show again metrics without a mapping.

As in Table 12 not all original metrics have a direct mapping to the metrics based on the results of the binary classifier. Therefore two different relation types are used:

= the result for the original metric, when acknowledging its preconditions, is the same as for the BC metric;

≈ the result for the original metric is not the same as for the BC metric, but both metrics focus on the same characteristic.

For example the intersection of EIS and AIS (E3.3) is the same as TP, since it contains all artifacts that have to be changed and are returned by the IA method.

The ratio between the number of elements in AIS and EIS (E3.1 and F2.3) has the precondition that AIS must be a subset of EIS. When this is the case, the result of this metric is the same as the result of Precision. So when the precondition is acknowledged, the result for E3.1 is the same as the result for Precision (since Precision has no preconditions, its results are meaningful for any input, making it more generic, but this is discussed when describing Table 14).

The Error as described by original metric G1 compares the errors made by the IA method to the returned result, while Accuracy compares the correctly identified elements with all elements in the system. So they both say something about the accuracy, but one based on the errors and the other based on the correctly identified elements.

Table 13: Mapping original metrics to BC-based metrics

Original metrics		**Binary classifier sets and metrics**							
Code	**Metric**	TP	FP	FN	TN	Recall	Precision	Fall-out	Accuracy
E1	|SIS| / |EIS|								
E2	|EIS| / |System|								
E3.1	|AIS| / |EIS|						=		
E3.2	|EIS| / |AIS|					=			
E3.3	|EIS ∩ AIS|	=							
E4	Granularity								
F1	AIS ⊆ EIS					=			
F2.1	|SIS| / |PIS|								
F2.2	|EIS| / |System|								
F2.3	|AIS| / |EIS|						=		

G1	(|DIS|+|FPIS|) / |CIS|								≈
H1	No errors		=	=					
H2	Identification errors			=					
H3	Update errors								
H4	Inclusion errors		=						

The mappings show which original metrics are not covered by metrics based on the binary classifiers. Granularity and Update errors are not covered since their definitions are not clear enough for exact quantification. The other original metrics not covered by the BC-based metrics, appear to be evaluation the change itself, instead of the impact analysis method. At the end of this section these metrics are evaluated to see if they describe any additional details about the performance of the impact analysis method.

The opposite mapping from BC-based metrics to the original metrics is presented in Table 14. Only the original metrics with any relation to BC-based metrics in Table 13 are presented in the columns to save space. The first four rows show the four results of the binary classifier and to which original definitions they can be mapped, just like the first four columns of Table 13. The other rows contain metrics using the four basic results to get a better view of the performance of an impact analysis method. Rows with a grey background do not have any mappings to original expressions. Some of the BC-based metrics are just the opposite of others, for example the False discovery rate and Specificity. The result range for the BC-based metrics is [0,1], the same range as for probabilities in probability theory. To invert a probability, it is decorated with the prime symbol: $P(A') = 1 - P(A)$. So when the result of a mapping is inverted, the mapping is also decorated with the prime symbol.

As in Table 12 not all BC-based metrics have a direct mapping to the original metrics. Therefore four different relation types are used:

= the result for the BC-based metric is the same as for the original metric;

⊇ the result for the BC-based metric is the same as for the original metric as long as the input is in the range of the original metrics preconditions. Nothing can be said about the result of the original metric when its preconditions are not met, while the result of the BC-based metric is meaningful for any input. So the input options for the original metric are a subset of or equal to the input options of the BC-based metric;

≈ the result for the BC-based metric is not the same as for the original metric, but both metrics focus on the same characteristic;

′ the result for the BC-based metric is the inverse of the result for the original metric.

When ⊇ and ′ are combined, this does not mean the superset relation is inverted. The superset symbol describes the relation between the allowed input sets for both metrics, while the prime symbol inverts the result or output value. Since the first relates to the input and the second to the output, they cannot be combined, for example into the ⊂ relation.

Table 13 showed that the intersection of EIS and AIS was the same as TP and this table shows TP is also the same as the intersection of EIS and AIS. Therefore the relation is marked with the = symbol.

Table 13 also showed that the ratio between the number of elements in AIS and EIS (E3.1 and F2.3) returned the same result as Precision if and only if AIS was a subset of EIS. However, when looking at Precision, it has no preconditions and it returns meaningful results for any combination of input sets, not only when AIS is a subset of EIS. Therefore the relation is marked with the $\supseteq$ symbol.

The False discovery rate has the opposite result of Precision, but it accepts the same input sets, so the relation is also marked with the $\supseteq$ symbol and accompanied by the prime symbol to indicate the result is inverted.

The relation between the Accuracy and the Error is the same as in Table 13, both give an indication of the number of errors or correctly identified elements.

Table 14: Mapping BC-based metrics to original metrics

Original expressions **Code**			E3.1	E3.2	E3.3	F1	F2.3	G1	H2	H4
		Name				Inclusiveness	Adherence	Error	Identification errors	Inclusion errors
BC expressions Name	**Synonyms**	**Expression Expression**	|AIS| / |EIS|	|EIS| / |AIS|	|EIS ∩ AIS|	AIS ⊆ EIS	|AIS| / |EIS|	(|DIS|+|FPIS|) / |CIS|		
True positive	Hit	TP			=					
False positive	False alarm, Type I error	FP								=
False negative	Miss, Type II error	FN							=	
True negative	Correct rejection	TN								
Recall	True positive rate (TPR), hit rate, sensitivity	TP / (TP + FN)		⊇		⊇				
Precision	Positive predictive value (PPV)	TP / (TP + FP)	⊇				⊇			
False discovery rate (FDR)	-	*FP / (TP + FP) = 1 – Precision*	⊇′				⊇′			
Accuracy (ACC)	-	(TP + TN) / (TP + FP + FN + TN)						≈		
Fall-out	False predictive rate (FPR), false alarm rate	FP / (FP + TN)								
Specificity	*True negative rate (TNR)*	*TN / (FP + TN) = 1 – Fall-out*								
Negative predictive value (NPV)	-	*TN / (TN + FN)*								

The mappings show that all original metrics covering impact analysis methods instead of the change itself are included in the BC-based metrics. So for the quantification of impact analysis performance the BC-based metrics describe at least all IA related original metrics. They describe even more, since no limiting preconditions are needed, making the BC-based metrics more generic and some metrics, like Fall-out are not covered at all by the original metrics.

The following original metrics do not have direct mappings, but do these metrics add any new information about the performance of the impact analysis method?

Ripple-sensitivity
According to its original definition ripple-sensitivity gives an indication of the work that has to be done to check the artifacts identified by the impact analysis method. It gives the ratio of artifacts in the starting set to all artifacts identified by the impact analysis method: |SIS| / |EIS| for Bohner or |PIS| / |SIS| for Fasolino. Since the AIS is not part of the equation, it is hard to say anything about the performance of the method based on this ratio. A big change has probably a bigger ripple-effect then a small change, so without the end result (AIS) nothing can be said about the IA method. At the same time the SIS is part of EIS, so it is already part of the Precision and Recall metrics.

Impact analysis is used to get an idea of the number of artifacts that have to be changed as a result of a proposed change. So the smaller the number of errors (not identified or not changed artifacts), the better the impact analysis method used by the tool. So the ripple-sensitivity of the change itself is |SIS| / |AIS|, it is however hard to measure the distance between this ratio and the ratio predicted by the tool using |SIS| / |EIS|. When the ripple-sensitivity is higher, this could mean the tool identified too many false positives, meaning the predicted impact is too big. This is also described by Precision and without the additional comparison to the actual ripple-sensitivity of the change itself. The same holds for a smaller ripple-sensitivity, this could mean the tool identified too little artifacts, meaning the predicted impact is too small. Something that is also described by Recall.

Sharpness
Sharpness shows the ratio between EIS and System and according to its original definition not all artifacts in the system should be affected, unless that is indeed the case. So the definition already indicates it does not say much about the performance of the impact analysis method used.

When comparing sharpness for two IA methods on the same change, this number says nothing about which of the two methods is better, since it is not related to AIS. Precision and recall show how many elements in EIS were correctly marked suspect and how many of the elements that needed change, were actually detected. This information can be used to identify which method performs better, instead of only showing which one returned most elements.

The only used for these two definitions is to give the reader a better view of the change being used to determine the performance of the impact analysis method.

Update errors
These errors are introduced when the changes made to the artifacts when performing the change, are not updated in the traceability information in the tool. Since this is outside the scope of impact analysis methods, there is not mapping.

5.2.3 Definitions

Since the metrics and definitions proposed by earlier work, are not always sufficient of the quantification of an impact analysis methods performance, problems with current definitions are described and the final set of definitions is presented.

5.2.3.1 Problems with definitions from literature

The three papers used before by Arnold, Bohner and Fasolino have some problems regarding the difference between measuring the performance of a change itself and the impact analysis method.

Bohner 1993
The problems with especially the *effectiveness* measurements is that the effectiveness of the change scenario itself is tested. Why is an impact analysis method predicting an EIS close to the System bad, when the AIS is also close to System and EIS? This means the impact analysis method performs well, but the change proposal itself is poor, since almost all elements have to be changed. The same holds for the comparison of EIS and SIS. A method returning an EIS much bigger then the SIS, but almost equal to the AIS, is again a good result. The change itself is maybe not, but that is not what *effectiveness* should test for. So in the end only the comparisons between the elements in EIS and AIS say something about the IA method itself.

When EIS and AIS are compared the metric used depends on a set of preconditions. It would be more robust if there were one or two metrics without any preconditions that would work in any case.

Granularity is not easily quantified. The evaluation of granularity offers three options, the artifact model (actual artifacts) has a finer granularity, the interface model (impact analysis tool) has a finer granularity, or both are about the same. There is however not really a way to measure this granularity for an objective comparison. It can be good criterion, since big difference in granularity between the artifacts and the tool can limit the usefulness of the impact analysis results. On the other hand, the key impact analysis itself is not really affected. The problems occur at the transitions across the interfaces. This is where (most of the time) a fine granularity is made coarser to get it into the impact analysis process. Since the impact analysis process only works on a set of objects, unaware of any granularity or context, the granularity does not affect the key impact analysis.

Fasolino 1999
Fasolino demands Inclusiveness, which means the Adequacy (Recall) is always one (AIS is a subset of EIS). The Effectiveness definitions are only valid when the investigated

method is Inclusive, limiting the use of this evaluation system, since none of the current systems is able to produce an Adequacy of one (without returning almost all elements in the system). Adjusting the definitions so they take in account AIS ∩ EIS != AIS can make this evaluation method usable in more cases.

Dropping Inclusiveness effects primarily Adherence = |AIS| / |EIS|, one of the ways to measure Effectiveness. It will get the same problem as the definitions by Bohner: preconditions to describe the sizes of both the AIS and EIS sets are needed to use the expressions. The Ripple-effect only inspects EIS and Sharpness compares EIS to the total System, so these two are not really affected when the Inclusiveness rule is dropped.

Fasolino is however also testing the change itself instead of the impact analysis method. Ripple-sensitivity and sharpness have nothing to do with the impact analysis method, leaving only inclusiveness and adherence. It is also a bit aggressive to exclude any method not retrieving all elements in the AIS, since traceability information helps to find missed artifacts.

Bohner 2002
Accuracy has been added to the earlier sets, describing the ratio between both the Type I and II errors and the total number of results of the impact analysis approach: Error = (|FPIS| + |DIS|) / |CIS|. Once the correct numbers are obtained during quantification, the Error is a clear measurement. This version of Accuracy is not exactly the same as Accuracy described in section 5.1.4.1, but is also affected by the Accuracy paradox. The effects for this specific version are described in section 5.2.3.2.

When all elements are identified correctly, the ratio is zero and when none of the elements is identified correctly, the ratio is |System|/|EIS|, which can be almost any value. So the result is always in the [0,∞> range, instead of the normalized [0,1] range, making comparisons harder. For a ratio in the [0,1] range the amount of errors should be divided by the largest possible amount, which is |System|. So a better definition would be: Error = (|FPIS| + |DIS|) / |System|.

Cleland 2003
Impacted elements (C,a) is a theoretical set of all elements impacted as result of change C on artifact a. It is not the same as the set of Identified artifacts (determined by the maintainer or the IA method) or the Updated artifacts (actually implemented). So it is almost impossible to obtain this set in a operational way.

Since the scenarios presented by Cleland to evaluate the IA method, are all using the hard to determine Impacted elements set, it is hard to use these scenarios to evaluate an impact analysis method in a real world case.

5.2.3.2 Accuracy paradox

The Accuracy as presented by Bohner is a little different as it is called Error and compares both errors with the positive test outcome:

Error = |FPIS + DIS| / |EIS| = (FP + FN) / (TP + FP)

In this case it is however also possible to create a higher Accuracy by mapping all elements to one test outcome for certain situations, as in Section 5.1.4.3. When the number of false negatives is equal or greater then the number of true negatives, the Error is reduced when the test outcome for all elements is positive[1].

Table 15 shows an example where fourteen artifacts were returned, from which four were false alarms and there was one miss. The Error for this test is 5/14. When instead of using a test, the test outcome of all artifacts just becomes positive, as shown in Table 16. The Error in this case becomes 5/16.

So again by not using any test at all, the Accuracy is increased by lowering the Error. This is logical, since the true negatives are not part of the Error equation. When the method marks all artifacts as positive, the number of true positives becomes the number of true positives and false negatives, and at the same time the number of false positives becomes the number of false positives and true negatives. So when the number of false negatives is equal or greater then the number of true negatives, the EIS becomes relatively larger then the addition of both errors, leading to a lower Error.

This shows Error is also not a reliable metric to determine a tests performance. The combination of Precision and Recall is much more reliable as the result of both these metrics would be zero when the actual test is replaced by a test returning always positive or negative as a result.

Table 15: Accuracy paradox, result normal test

		Condition	
		Relevant	**Not Relevant**
Test outcome	**Positive**	10	4
	Negative	1	1

Table 16: Accuracy paradox, result positive test

		Condition	
		Relevant	**Not Relevant**
Test outcome	**Positive**	11	5
	Negative	0	0

5.2.3.3 Final definitions

The sets are used to compare different impact analysis methods on the same set of changes, not to analyze the result of a single impact analysis for a single change. But the more it can be used on its own the better, since less other methods are needed for comparison.

[1] To be precise: when $FN^2 + (2FP+TP+TN)FN - TP \bullet TN > 0$, the Error is smaller when the test outcome for all elements is positive, demonstrating the accuracy paradox.

To evaluate the performance of impact analysis methods a number of sets is needed:

- System: set of all artifacts in the tool.
- EIS: set of all artifacts that have to be changed according to the impact analysis method.
- AIS: set of all artifacts actually changed during correct implementation of the change.
- FPIS: set of all false positives or false alarms identified by the impact analysis method, can be derived from EIS and AIS.
- DIS: set of all false negatives or misses identified by the impact analysis method, can also be derived from EIS and AIS.

These sets can then be used to obtain values for the following metrics:

- Recall gives the ratio of artifacts identified by the impact analysis method that have been actually changed to all artifacts that have been changed: $|EIS \cap AIS| / |AIS|$. The result range is [0,1] and closer to one means a smaller underestimation of the change size.
- Precision gives the ratio of artifacts identified by the impact analysis method that have been actually changed to all artifacts that have been identified: $|EIS \cap AIS| / |EIS|$. The result range is [0,1] and closer to one means a smaller overestimation of the change size.
- Fall-out gives the ratio of artifacts identified by the impact analysis method that have not been actually changed to all artifacts that have not been changed: $|FPIS| / (|System|–|AIS|)$. Like precision it gives and indication of the overestimation of the method. The result range is [0,1] and closer to one means a bigger overestimation of the change size. This alternative is used in the ROC plot.
- F-score provides a weighted average of Precision and Recall where a result of zero indicates a poor test result, while a result of one means the result of the test was perfect: $2 \cdot (Precision \cdot Recall) / (Precision + Recall)$.

Accuracy is left out since it does not really add information compared to Precision and Recall. No distinction is made between the two types of correct identifications and the two types of incorrect identification, nor the relations between them. Sharpness and ripple-sensitivity have also been left out, since they only described the change and not the performance of the impact analysis method.

5.2.4 Visualize and compare impact analysis metrics

When the results of an impact analysis method are presented in a ROC plot, the meaning of the important points in the plot are as follows. A result in the top left corner, means there were no false alarms and all relevant artifacts were found, so this is the best result possible. Results on the ascending diagonal have the same amount of true results as false alarms, meaning this method has the same performance as a random guess. So the closer a result is to (0,1) and the further away from the diagonal, the more reliable the predictive capabilities of a method or configuration are. An example is presented in Figure 22.

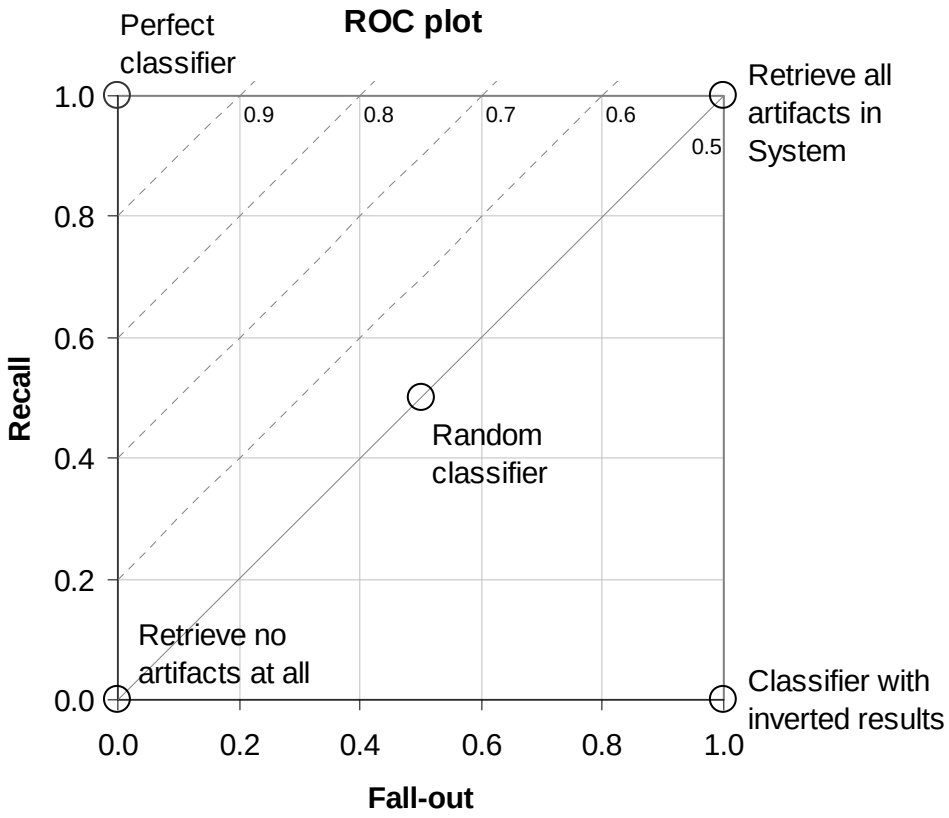

Figure 22: ROC plot for use with impact analysis

When plotting the results of the impact analysis method in a PR plot, the following data points are important. An ideal result would have a Recall and Precision of one, meaning there are no false alarms or misses and the prediction was perfect. So the closer a result is to the upper right corner, the better the impact analysis performance of a method. When a method retrieves all available elements, its Recall will be high, all relevant artifacts have been identified, but the Precision will be low, since there will be a lot of false alarms. When a method returns only one relevant artifact, its Precision will be high, there are no false alarms, but the Recall will be low, since there are probably a number of misses. Figure 23 shows all points in an actual plot.

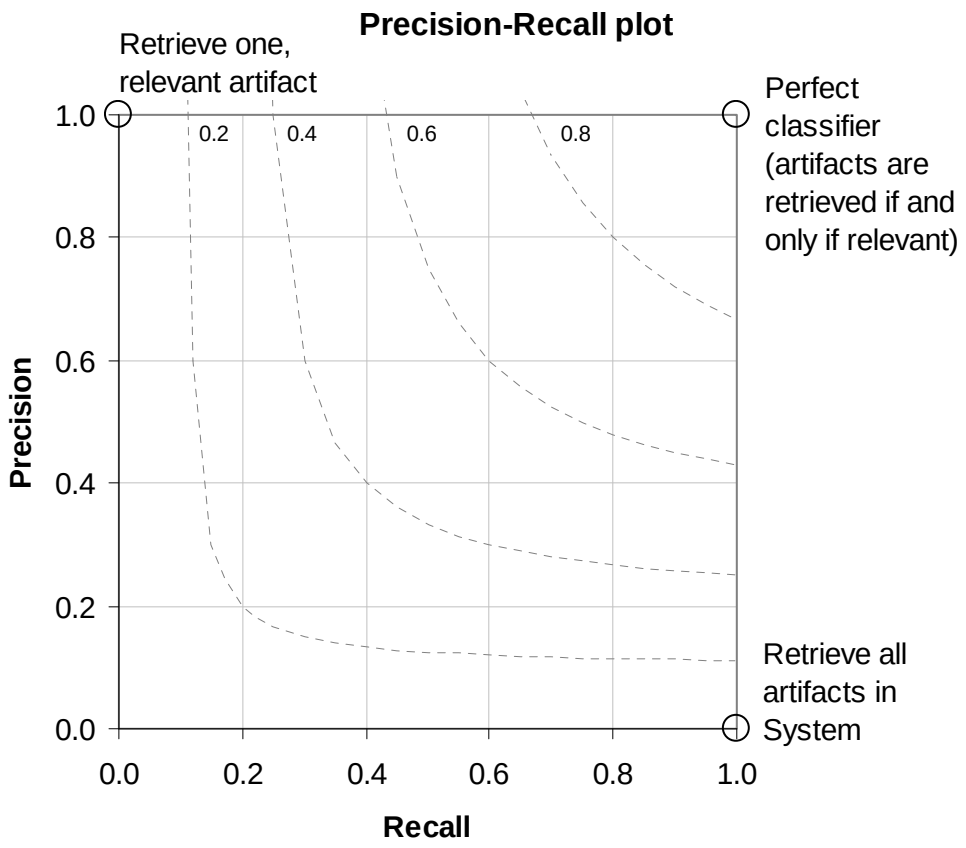

Figure 23: PR plot for use with impact analysis

Based on the differences between the ROC and PR plot described in Section 5.1.4.2, the PR plot shows the most reliable image of a methods performance. Most changes in software systems will not change half of the available artifacts each time. So the ratio of relevant to irrelevant artifacts is in most cases highly skewed, making a PR plot the better option to visualize the results.

Since the ROC plot uses Fall-out, which compares false positives to true negatives, in the case of a set with a lot of irrelevant elements, the Fall-out changes only a little when some false positives are added. When the number of true positives is also small (which it is as a result of the relevant to irrelevant ratio) the Precision changes more aggressively, since it looks at the ratio between true positives and false positives. Therefore the PR plot gives a more interesting view of the data when it is skewed.

In Section 5.1.4.2 a number of isometrics were presented for both plots to be able to compare two data points. The problem with most of these isometrics was their sensitivity to different relevant to irrelevant ratios. Since the results for multiple test scenarios will be plotted in the same figure for each impact analysis method, it is important the isometric used does not depend on this ratio.

For the ROC plot this means only the AUC can be used. For the PR plot both the AUC and F-score can be used. Since the F-score is a weighted average of Precision and Recall, much used in statistics, and AUC is a generic metric, the F-score is used as isometric for PR plots.

5.3 Case study CMS

In Part A the WASP case was used to test the features present in the requirements management tools. The WASP case does provide no change scenarios, so it cannot be used to determine the performance of the current requirements management tools. Therefore the Course Management System (CMS) case (see Figure 24) used by Goknil is being implemented in DOORS and RequisitePro [GKB08]. The requirements for this course management system describe basic functionality like enrolling for courses, uploading rosters and course material, grading students and sending e-mails to students. The CMS contains sixteen requirements as artifacts, seventeen relations to connect them and three change scenarios to determine the performance of the impact analysis methods. Each of the requirements is described in Appendix A.

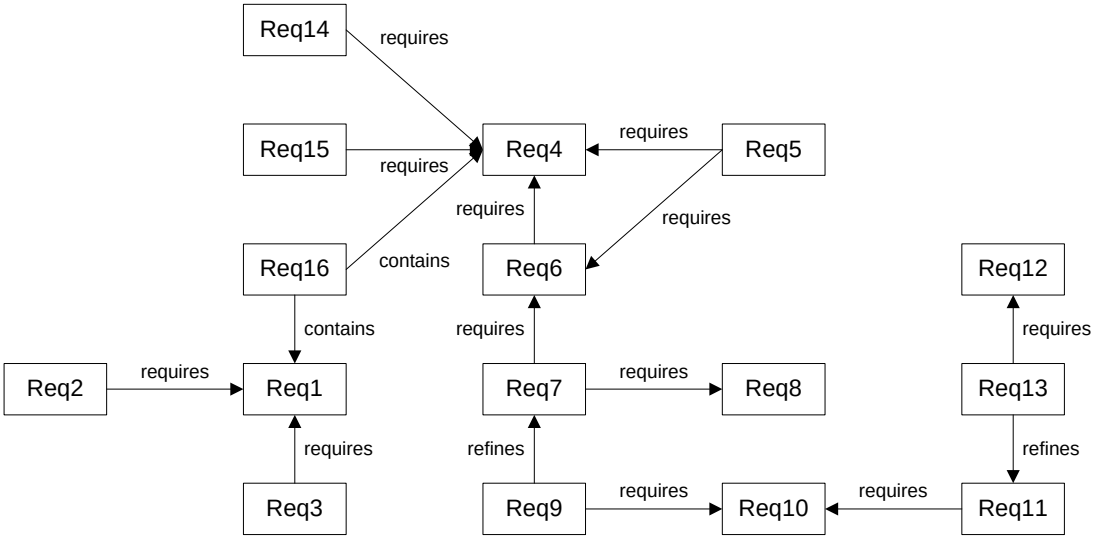

Figure 24: Subsection of the Course Management System

5.3.1 Changes

The following three changes were made to the CMS:

- Delete Req6: The system shall maintain a list of events the students can be notified about.
- Modify Req9: The system shall notify students about the events of the lectures they are enrolled for.
- Modify Req11: The system shall allow lecturers to send e-mail to students enrolled for the lecture given by that lecturer.

When implementing these changes in the tools, the support for impact analysis for the deletion of artifacts seemed to be missing. To get a better view of what would be possible

when the impact analysis method used for modifications was also used for deletion, Req6 was also modified before deletion. This change scenario is called ‘Modify Req6’.

These four changes will be used in Chapter 6 to determine the impact on the other artifacts. The input for the impact analysis is Starting Impact Set (SIS) or Primary Impact Set (PIS). For each of the changes this set only contains the changed or deleted requirement, so the cardinality for each change is one. The impact of these changes as determined by Goknil, using a combination of an automated IA method and manual confirmation, is used as the Actual Impacted Set (AIS), since these are the artifacts actually changed when the changes would be performed.

5.4 Conclusion

This chapter has shown that earlier techniques to quantify the performance of impact analysis have serious limitations. They can be used in a limited number of cases and often test the quality of the change itself instead of the performance of the impact analysis method.

The metrics based on binary classifiers cover all impact analysis performance aspects also covered by the quantitative metrics in the earlier techniques. In addition they do not use restrictive preconditions, but can be used for all cases. They also focus on the performance of the impact analysis method instead of the change that is used to test the method. Therefore, the metrics based on binary classifiers appear to be a good way to quantify the impact analysis performance of our requirements management tools.

6 Change impact performance quantification

``To change is difficult. Not to change is fatal.''

Ed Allen, contributor to the cable television industry, 1924-2000

The various impact analysis performance quantification methods introduced in Chapter 1 is applied to two of the requirements management tools investigated before. The results of all methods are compared to see if the method based on binary classifiers is indeed a reliable way to quantify the performance of an impact analysis method. Finally the two tools will be compared to see which one has the best impact analysis performance.

6.1 Application of metrics on impact analysis

In Section 1 a lot of sets and metrics have been introduced. To test what these sets and metrics can tell about the impact analysis method used by a tool, two tools will be used to implement a test case and a number of changes and the results of this implementation will be analyzed using the sets and metrics.

All sets and metrics with clear operational definitions have been gathered in the following list:

Sets

A. Arnold/Bohner 1993:
   - A1. Universe
   - A2. System
   - A3. SIS
   - A4. EIS
   - A5. AIS

B. Fasolino 1999:
   - B1. System
   - B2. PIS (primary)
   - B3. SIS (secondary)
   - B4. EIS (=PIS+SIS)
   - B5. AIS

C. Bohner 2002:
   - C1. System
   - C2. SIS
   - C3. CIS
   - C4. AIS
   - C5. FPIS
   - C6. DIS

D. Cleland 2003:
   - D1. Identified artifacts
   - D2. Updated artifacts

Metrics

E. Arnold/Bohner 1993:
   E1. SIS and EIS: |SIS| / |EIS|
   E2. EIS and System (sharpness): |EIS| / |System|
   E3. EIS and AIS:
      E3.1. |AIS| / |EIS|
      E3.2. |EIS| / |AIS|
      E3.3. |EIS ∩ AIS|
F. Fasolino 1999:
   F1. Inclusiveness: AIS is a subset of or equal to EIS
   F2. Effectiveness
      F2.1. Ripple-effect: Amplification = |SIS| / |PIS| -> 1
      F2.2. Sharpness: ChangeRate = |EIS| / |System|
      F2.3. Adherence: S-Ratio = |AIS| / |EIS|
G. Bohner 2002:
   G1. Accuracy: Error = (|DIS| + |FPIS|) / |CIS|
H. Binary Classification:
   H1. Precision: |AIS ∩ EIS| / |EIS|
   H2. Recall: |AIS ∩ EIS| / |AIS|
   H3. I3 Fall-out: |FPIS| / (|System|–|AIS|)
   H4. F-score
   H5. Area under curve (AUC)

6.2 Implementing the CMS

The four changes for the CMS example have been implemented in DOORS and RequisitePro. DOORS is chosen because it seems to be the tool with the best options and RequisitePro has transitivity support for links. TopTeam and CaliberRM have no special impact analysis features compared to DOORS and RequisitePro and therefore they are not used for the example case. The sizes of the sets related to these changes have been gathered in Table 17. The complete sets can be found in 7.4Appendix B.

The rows containing sets A1 to D3 are the sizes of the different sets needed to say something useful about the various analysis methods. The cardinalities of these sets can only differ for both tools, when the sets contain the result of the impact analysis method or subsets of the result:

- EIS, CIS, FPIS and DIS for Bohner;
- SIS (Secondary Impact Set) and EIS for Fasolino;
- Identified artifacts for Cleland.

Estimated Impact Set (EIS), CIS, Identified artifacts and SIS (A4, B3, B4, C3, D2)
For the deletion of Req6 and the modification of Req11 both tools return the same EIS. In the case of Req6 the EIS only contains Req6 itself as a result of the lack of IA support for the deletion of artifacts in both tools. Req11 only has two direct relations with artifacts that have no further relations in the same direction. So the result for DOORS, without transitive closure, and RequisitePro, with transitive closure, is the same. When Req6 is

modified instead of removed, RequisitePro marks one more artifact suspect as a result of its transitive closure support. The same holds for the modification of Req9.

False Positive Impact Set (FPIS) (C5)
The amount of false positives in the IA result set, so the artifacts present in EIS, but not present in AIS: |EIS – AIS|. For the deletion of Req6 only Req6 itself was suspect, so there are no false positives. When modifying Req6 both tools include Req4, as it has a direct relation with Req6, but in the end it was not altered. Modifying Req9 only modifies Req9 in the end, so all other direct and indirect related artifacts identified by both tools are false positives. Since RequisitePro applied transitive closure, the amount of false positives was higher. When modifying Req11 only one of the two detected suspects was actually implemented, leading to one false positive for both tools.

Discovered Impact Set (DIS) (C6)
The amount of false negatives not in the result set, so the artifacts present in AIS, but not present in EIS: |AIS – EIS|. The deletion of Req6 resulted in adjusting three other requirements, since both tools missed all these artifacts, both have three false negatives. When Req6 is modified DOORS misses one artifact that should be changed, while RequisitePro detects all of them. For the two remaining changes none of the tools miss any artifacts that have to be changed.

Rows E1 to G1 contain the metrics used to obtain numbers describing the performance of the impact analysis methods of the tools for each change.

Only the differences between to results returned by the tools are discussed and since the result sets for both the deletion of Req6 and the modification of Req11 were the same, these will not be discussed. In the end the differences between both tools are caused mostly by the transitive closure used in RequisitePro, resulting in more identified artifacts, and the 1-level transitivity in DOORS, resulting in less identified artifacts.

Sharpness, |EIS| / |System| (E2, F2.2)
Gives more of an indication of the change then of the impact analysis method. The size of the EIS compared to the system does not say anything about the impact analysis method on its own, only when it is compared to the result of other tools. More artifacts marked suspect leads to a larger Sharpness and since RequisitePro uses transitive closure, leading to more suspects, its Sharpness is equal or larger then DOORS'. It is however hard to say something about the quality of this IA method based on this information alone. It really depends on the change if more or less Sharpness is good in this case.

Ripple-effect, |SIS| / |EIS|, |SIS| / |PIS| (E1, F2.1)
The Ripple-effect shows the ratio between the number of artifacts in the starting set and the number of results of the impact analysis method. Bohner describes this ratio as the size of the starting set compared to the total result set, while Fasolino describes it as the number of elements purely identified by the IA method compared to the starting set, also called the amplification. As a result of the transitive closure in RequisitePro, the number of results returned is larger then for DOORS, leading to more Ripple-effect for the same

change. Again without knowledge about the change and other tools to compare it with, it is hard to say anything about the quality of an IA method using this information.

Adherence, |AIS| / |EIS|, Inclusiveness, AIS ⊆ EIS (E3.2, F2.3, F1)
For the expression |AIS| / |EIS|, Bohner added the pre-condition that AIS has to be a subset of EIS (FN=0), so the result is always in the range [0,1]. It indicates that the situation is safe, all elements were detected by the IA method. All Fasolino's expressions have the pre-condition that AIS should be a subset of EIS, it is called Inclusiveness. Otherwise all artifacts have to be inspected, resulting in a useless IA method, according to Fasolino.
For the deletion of Req6 both tools do not include AIS in EIS and for the modification of Req6 only DOORS does not include AIS in EIS. This means the precondition of Bohner is not met in these cases, so the expression cannot be applied and the rest of Fasolino's expressions have no use, since the IA method failed the first test.
The other two changes and the modification of Req6 by RequisitePro include AIS in the EIS. Since RequisitePro included AIS in EIS for the modification of Req6 and DOORS did not, the first performed better. For the modification of Req9 DOORS performed better then RequisitePro, since it did not return as many artifacts as a result of the lack of transitive closure.
When the Recall=1, Bohner describes the Precision, so it really tells something about the IA method.

|EIS| / |AIS|, (E3.1)
For the expression |EIS| / |AIS|, Bohner added the pre-condition that EIS was a subset of AIS (FP=0), so the resulting value was always in the range [0,1]. When an IA method is used with clear and explicit relations between its artifacts, a small difference between EIS and AIS means less work to detect the artifacts missing in the result set and more work when the difference between EIS and AIS becomes bigger.
When relations between artifacts are less clear, it becomes harder to find affected artifacts based on relations. So when not all artifacts are detected by the IA method, this means all artifacts in the system have to be investigated manually. In such a case this metric says less about an IA method.
When the Precision=1, Bohner describes the Recall, so it really says something about the IA method.

|EIS ∩ AIS| (E3.3)
The pre-conditions added by Bohner are |EIS ∩ AIS| ≥ 0 (always true, very hard to produce negative cardinalities this way) and AIS ≠ EIS. Since there is no case in our example in which EIS is equal to AIS, this expression would apply to all change scenarios and tools. What Bohner wanted to test were situation in which the two sets were disjoint. So the precondition should be: |AIS – EIS| > 0 and |EIS – AIS| > 0 (= FN > 0 and FP > 0). When modifying Req6 DOORS missed one artifact that should be changed and RequisitePro did not, so it this case the latter performed better. When modifying Req9 only one artifact was changed in the end and it was part of both result sets, so they performed the same. On its own this expression does not say much about the

performance of a tool, but when combined with some others it can (basis for Precision and Recall).

Error, (|DIS| + |FPIS|) / |CIS| (G1)
The Error as described by Bohner is the number of errors per retrieved artifact. When Req6 is being modified, RequisitePro returns more results and one false alarm. DOORS returns less artifacts, also one false alarm and it also misses one artifact. So RequisitePro has a smaller Error and performs better. When Req9 is modified, only one artifact was actually changed. Both tools returned more artifacts, but as a result of the transitive closure in RequisitePro, it returned more artifacts and therefore the Error was bigger and DOORS performs better.
Metric works with the error related to the result, so it says something about the IA method, but is susceptible to the Accuracy paradox.

Based on these individual expressions it is hard to say something about the performance of an IA method. It says more about the change itself. Therefore these expressions based on binary classification can help:

Recall, |EIS ∩ AIS| / |AIS| (H1)
Recall describes the ratio between the actually changed artifacts in the result set and in the total system. For IA it is important to have a Recall of one, since misses can make the change look smaller then it actually is. Identifying missed artifacts can mean inspecting a large portion of artifacts in the system, taking a lot of time and money. So it is a more general version of Bohner's |EIS| / |AIS| and Fasolino's AIS ⊆ EIS (meaning Recall=1). When modifying Req6, DOORS missed one artifact that had to be changed, while RequisitePro found all because of its transitive closure. This makes RequisitePro the better performer here. When modifying Req9 both identify the one artifact being changed, making their performance equal.

Precision, |EIS ∩ AIS| / |EIS| (H2)
Precision describes the ratio between actually changed artifacts in the result set and all identified artifacts in the result set. For IA it is desirable to have a Precision of one, since false alarms make a change look bigger then it actually is. All artifacts in the result set are inspected to see what must be changed, and false alarms need no changes at all, saving time and money. It is a more general version of Bohner and Fasolino's |AIS| / |EIS| expression. When Req6 is modified both tools return on false alarm, but since RequisitePro returned more artifacts in the result set, its Precision is a little higher. When Req9 is modified only one artifact is actually changed, but both tools return more artifacts, causing a number of false alarms. As a result of RequisitePro's transitive closure, it returns more false alarms then DOORS in this case, leading to a lower Precision.

Fall-out, |FPIS| / (|System|–|AIS|) (H3)
The Fall-out is in a way the inverse of the Recall. It describes the ratio between the false alarms in the result set and the total number of artifacts in the system not affected because of the change. Since it looks at the false alarms, just as Precision, it gives an idea of how

much the IA method overestimates the number of results. The Fall-out value will be big when a there are a lot of false alarms and a lot of artifacts are actually changed. Therefore, a high value means the change is big, but not as big as predicted by the IA method. Small Fall-out values mean there are only a small amount of false alarms and little artifacts are changed. So the prediction of the change is small, but not much bigger then predicted.
When Req6 is modified, both tools identify only one false alarm, resulting in the same score. When Req9 is modified, RequisitePro returns more false alarms as a result of the transitive closure, leading to a higher Fall-out then DOORS.

F-score (H4)
The F-score is the harmonic average of Precision and Recall. It makes it possible to combine these two metrics into one number to determine the absolute order of results. Low results for Precision or Recall have a serious effect on the F-score. Tools focusing on only one of these metrics and ignoring the other will therefore obtain really low values on this test.
This behavior is shown really well when Req9 is modified. Both tools achieve a Recall of one, but Precision is low. As a result both F-scores are also low, much lower then the normal average of Precision and Recall. When Req6 is modified both Precision and Recall are close to one, resulting in a F-score more close to the normal average of both metrics.

Area under curve (AUC) (H5)
The AUC is based on the ROC plot and combines the results of Recall and Fall-out into one number as the F-score does for Precision and Recall. Non-random results will have a AUC between 1 and a 0.5 and the closer to one the better the result. The AUC behaves like a normal average, so a low Recall or Fall-out has no stronger effect on the AUC then a high result.
When Req6 is modified RequisitePro's Recall is higher then DOORS', leading to a higher AUC. The higher number of indirectly effected artifacts when modifying Req9 leads to the opposite result: DOORS has a lower Fall-out, leading to a higher AUC.

Table 17: Impact analysis results for change scenarios in both tools

		Delete Req6		**Modify Req6**		**Modify Req9**		**Modify Req11**	
Code	**Set or metric**	RequisitePro	DOORS	RequisitePro	DOORS	RequisitePro	DOORS	RequisitePro	DOORS
A1	|Universe|	∞	∞	∞	∞	∞	∞	∞	∞
A2	|System|	16	16	16	16	16	16	16	16
A3	|SIS|	1	1	1	1	1	1	1	1
A4	|EIS|	1	1	5	4	6	3	3	3
A5	|AIS|	4	4	4	4	1	1	2	2
B1	|System|	16	16	16	16	16	16	16	16

B2	|PIS| (primary)	1	1	1	1	1	1	1	1
B3	|SIS| (secondary)	0	0	4	3	5	2	2	2
B4	|EIS| (|PIS| + |SIS|)	1	1	5	4	6	3	3	3
B5	|AIS|	4	4	4	4	1	1	2	2
C1	|System|	16	16	16	16	16	16	16	16
C2	|SIS|	1	1	1	1	1	1	1	1
C3	|CIS|	1	1	5	4	6	3	3	3
C4	|AIS|	4	4	4	4	1	1	2	2
C5	|FPIS|	0	0	1	1	5	2	1	1
C6	|DIS|	3	3	0	1	0	0	0	0
D1	|Identified|	1	1	5	4	6	3	3	3
D2	|Updated|	4	4	4	4	1	1	2	2
E1	|SIS| / |EIS|	1	1	1/5	1/4	1/6	1/3	1/3	1/3
E2	|EIS| / |System|	1/16	1/16	5/16	4/16	6/16	3/16	3/16	3/16
E3.1	|AIS| / |EIS|	*4*	*4*	4/5	*1*	1/6	1/3	2/3	2/3
E3.2	|EIS| / |AIS|	1/4	1/4	*1/4*	*1*	*6*	*3*	*3/2*	*3/2*
E3.3	|EIS ∩ AIS|	1	1	4	3	1	1	2	2
F1	AIS ⊆ EIS	False	False	False	False	True	True	True	True
F2.1	|SIS| / |PIS|	*0*	*0*	*4*	*3*	5	2	2	2
F2.2	|EIS| / |System|	*1/16*	*1/16*	*5/16*	*4/16*	6/16	3/16	3/16	3/16
F2.3	|AIS| / |EIS|	*4*	*4*	*4/5*	*1*	1/6	1/3	2/3	2/3
G1	(|DIS|+|FPIS|) / |CIS|	3	3	1/5	2/4	5/6	2/3	1/3	1/3
H1	Precision: |EIS ∩ AIS| / |EIS|	1	1	4/5	3/4	1/6	1/3	2/3	2/3
H2	Recall: |EIS ∩ AIS| / |AIS|	1/4	1/4	1	3/4	1	1	1	1
H3	Fall-out: |FPIS| / (|System|–|AIS|)	0	0	1/12	1/12	5/15	2/15	1/14	1/14
H4	F-score	2/5	2/5	8/9	6/8	2/7	2/4	4/5	4/5
H5	AUC	0.63	0.63	0.96	0.83	0.83	0.93	0.96	0.96

Table 18: Impact analysis results in decimals for final selection of metrics

H1	Precision: |EIS ∩ AIS| / |EIS|	1.00	1.00	0.80	0.75	0.17	0.33	0.67	0.67
H2	Recall: |EIS ∩ AIS| / |AIS|	0.25	0.25	1.00	0.75	1.00	1.00	1.00	1.00
H3	Fall-out: |FPIS| / (|System|–|AIS|)	0.00	0.00	0.08	0.08	0.33	0.13	0.07	0.07
H4	F-score	0.40	0.40	0.89	0.75	0.29	0.50	0.80	0.80
H5	AUC	0.63	0.63	0.96	0.83	0.83	0.93	0.96	0.96

Results in italic are not relevant according to the analysis method based on preconditions, but have been calculated anyway.

6.3 Presentation of metrics and visualization

A summary of Table 17 can be found in Table 19. It contains only the final definitions as presented in Section 5.2.3.3.

Table 19: Performance of RequisitePro and DOORS

		Delete Req6		Modify Req6		Modify Req9		Modify Req11	
Code	Set or metric	RequisitePro	DOORS	RequisitePro	DOORS	RequisitePro	DOORS	RequisitePro	DOORS
A2	|System|	16	16	16	16	16	16	16	16
A4	|EIS|	1	1	5	4	6	3	3	3
A5	|AIS|	4	4	4	4	1	1	2	2
C5	|FPIS|	0	0	1	1	5	2	1	1
C6	|DIS|	3	3	0	1	0	0	0	0
H1	Precision |EIS ∩ AIS| / |EIS|	1.00	1.00	0.80	0.75	0.17	0.33	0.67	0.67
H2	Recall |EIS ∩ AIS| / |AIS|	0.25	0.25	1.00	0.75	1.00	1.00	1.00	1.00
H3	Fall-out: |FPIS| / (|System|–|AIS|)	0.00	0.00	0.08	0.08	0.33	0.13	0.07	0.07
H4	F-score	0.40	0.40	0.89	0.75	0.29	0.50	0.80	0.80
H5	AUC	0.63	0.63	0.96	0.83	0.83	0.93	0.96	0.96

The results of our case study in Table 19 show the performance for none of the changes is perfect. Both tools perform rather well when predicting the impact of the modification of Req6 en Req11. The performance for the deletion of Req6 is not that good, since both tools missed a lot of artifacts that had to be changed. Predicting the impact of modifying Req9 was hard too, since both tools identified too many artifacts that were not changed in the end. Based on these results it is hard to pick a clear winner. RequisitePro performed better when modifying Req6, while DOORS performed better when Modifying Req9 and in the other two cases their performance was the same.

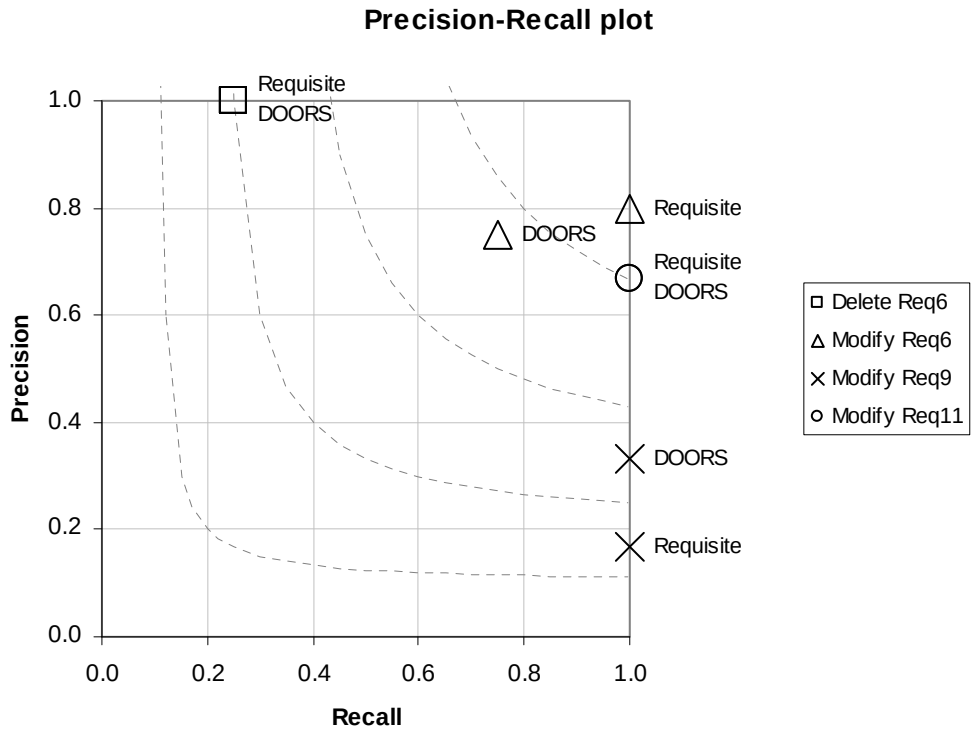

Figure 25: Precision-Recall plot with change scenario results

The results have also been plotted in a ROC plot, shown in Figure 26. Most results are close to the upper left corner, meaning they are reliable. The deletion of Req6 is closed to the ascending diagonal, meaning it is less reliable. Since it is also close to the lower left corner, it means both tools behave as if they just retrieve no artifacts at all. This is correct, since both tools do not perform any analysis when artifacts are deleted. For the modification of Req9 both tools showed low performance in the PR plot. In the ROC plot both results are relatively close to the ideal point, meaning the low performance for this change is reliable.

The plot shows the characteristics of both tools quite good. The transitivity used by RequisitePro caused too many false alarms in some cases, moving results towards the top right corner (returning all artifacts and therefore catching all true results, but also all false results). The lack of transitivity in the method used by DOORS causes results closer to the lower left corner (returning no results at all and therefore missing all false alarms, but also all true results). In addition, the lack of proper support for the deletion of artifacts in both tools becomes clear, since this result is really close to the lower left corner.

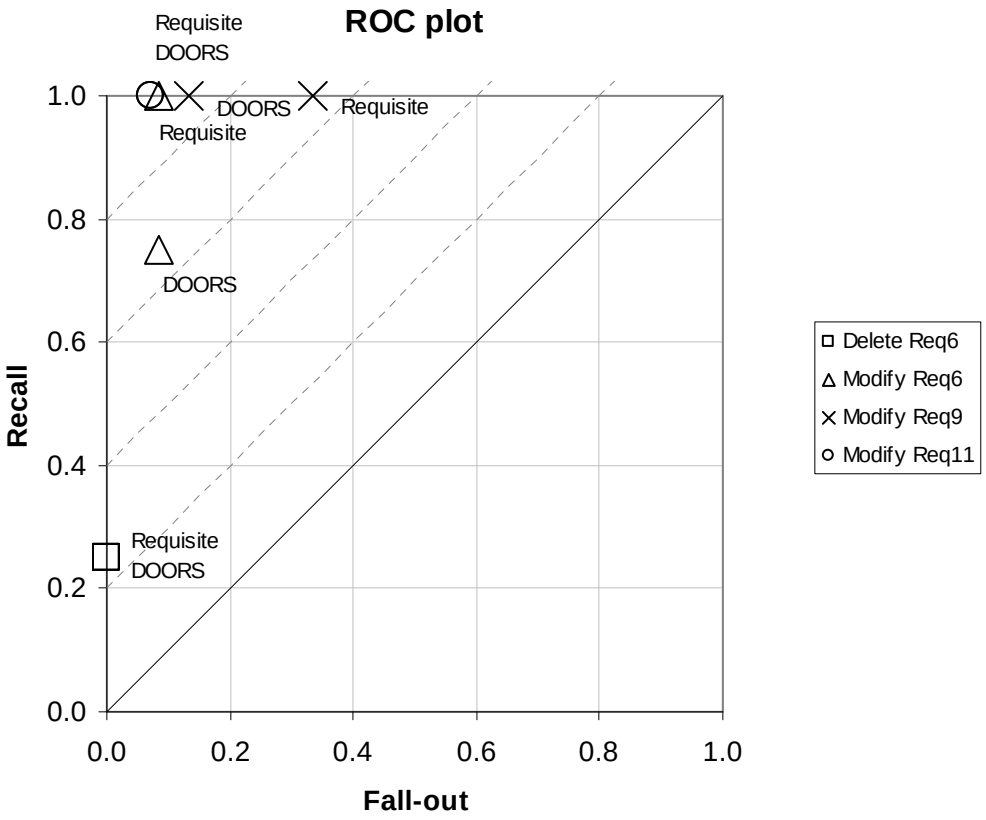

Figure 26: ROC plot with change scenario results

6.4 Conclusion

First of all the lack of a real life case makes it hard to measure the real performance of the tools or the quantification methods. But based on the results of the sample case it looks like the metrics based on binary classifiers give a better view of a methods performance. It is clear it looks only at the impact analysis method and not the quality of the change itself, as some of the existing methods do.

There is no clear winner in our set of requirements management tools when it comes to impact analysis. Both tools have their strengths and weaknesses and both are clearly shown in the results, but in the end they level out to the same level.

7 Conclusion

``Change starts when someone sees the next step.''

William Drayton, American politician, 1776-1846

The work done will be summarized in Section 7.1. The research questions defined in the introduction will be answered in Section 7.2. In Section 7.3 we will reflect on our work and discuss on the methods used to evaluate the requirements management tools and the results of this evaluation. Section 7.4 describes some of the open issues that still need investigation.

7.1 Summary

Since every company, market and economy are subject to change, the software systems to support them are too. The requirements are the foundation of each system and the basis for all of the following artifact types. As a result good management of requirements is essential when changes to a software system are initiated. A lot of tools are available to support the change management process and a number of these tools have been evaluated in this work.

The current requirements management tools will be represented by a selection of four tools. Three of them are used by partners of the QuadREAD project: IBM RequisitePro, Borland CaliberRM and TopTeam Analyst. The last tool, TeleLogic's DOORS, is mentioned often in the literature for its extended set of options for requirements management and therefore also part of the selection.

7.1.1 Criteria for tool comparison

Based on previous research on change management a number of criteria have been gathered to test if and to what extend tools have features to support change impact analysis. Some of the properties tested for are information model support, options to import and export artifacts, integration with other tools used in the software life cycle, graphical representation of the artifacts and their relations, coverage analysis and impact analysis. The final list of criteria is presented in Section 3.2.

The criteria have been applied to the four selected tools to determine their support for impact analysis related features. The WASP case has provided us with an example information model, containing different types of requirements, and a concrete set of requirements.

Based on these criteria none of the tools turned out to be a clear winner; all tools had poor impact analysis features, meaning most work still had to be done by hand. The differences between the tools appeared in their supporting features for the actual impact analysis process. DOORS offered the best basis for impact analysis through its extended information model support, providing custom link types and custom attributes for links. Since the actual impact analysis in all tools is very limited, DOORS offers at least the

options to model different relation types so the traceability information can be exported and used in a tool specialized in impact analysis.

7.1.2 Measure performance of impact analysis methods

Besides determining the features supporting impact analysis in current requirements management tools, quantifying their actual performance shows the quality of the impact analysis method provided by the tools. Literature describes a number of methods to quantify the performance of impact analysis methods. Measuring performance using these metrics does not really express the performance of the impact analysis method, but more the characteristics of the change itself. These metrics also come with a number of preconditions, limiting their use.

A series of new definitions have been proposed to provide better metrics to determine the impact analysis performance of tools, based on the concepts of binary classifiers.

To evaluate the performance of impact analysis methods, a number of sets is needed:

- **System**: set of all artifacts in the tool.
- **Estimated Impact Set** (EIS): set of all artifacts that have to be changed according to the impact analysis method.
- **Actual Impact Set** (AIS): set of all artifacts actually changed during a correct implementation of the change.
- **False Positive Impact Set** (FPIS): set of all artifacts identified by the impact analysis method, but not actually changed during the implementation.
- **Discovered Impact Set** (DIS): set of all artifacts not identified by the impact analysis method, but changed during the implementation.

These sets can then be used to obtain values for the following metrics:

- **Recall** gives the ratio of artifacts identified by the impact analysis method that have been actually changed to all artifacts that have been changed: $|EIS \cap AIS| / |AIS|$. Also known as hit rate, sensitivity or true positive rate (TPR).
- **Precision** gives the ratio of artifacts identified by the impact analysis method that have been actually changed to all artifacts that have been identified: $|EIS \cap AIS| / |EIS|$. Also known as positive predictive value (PPV).
- **Fall-out** gives the ratio of artifacts identified by the impact analysis method that have not been actually changed to all artifacts that have not been changed: $|FPIS| / (|System| - |AIS|)$. Also known as false alarm rate or false positive rate (FPR).

For both Recall and Precision a result of one means the best result possible for the aspect they cover and zero means the worst result possible. For Fall-out it is the other way around; one means worst possible result, while zero represents the best possible result for this aspect.

7.1.3 Visualization

Literature presents two types of plots to visualize these metrics: the Precision-Recall plot and ROC plot. For each change scenario performed in a tool the previous metrics can be calculated, resulting in a single point in each of the plots. Plotting the results of multiple

tools for a particular change scenario can be used to determine what tool predicts this type of change best. Plotting the results of multiple change scenarios for a single tool to discover what type of changes are handled best by this tool.

The Precision-Recall plot assigns the Recall value of a result to the horizontal axis and the Precision value to the vertical axis. An ideal result would have a Recall and Precision of both one, meaning there are no false alarms or misses and the prediction was perfect. Results closer to the lower left corner are caused by low Precision and Recall numbers, meaning relative large amounts of false alarms and misses and therefore a bad performing impact analysis method. The upper left corner is home to results with low Recall but high Precision, probably caused by a method returning only a very limited amount of correct results, but as a result also a lot of misses. Results in the lower right corner are the result of the opposite strategy. Impact analysis methods returning almost all artifacts in the system will get a high Recall ratio, but a low Precision ratio. So the closer a result is to the upper right corner, the better the impact analysis performance of a method.

The ROC plot assigns the Fall-out value of a result to the horizontal axis and the Recall value to the vertical axis. As a result the ROC plot shows how the number of changed artifacts marked suspect by the method varies with the number of non-changed artifacts incorrectly marked suspect by the method. This means a result in the top left corner shows there were no false alarms and all relevant artifacts were identified by the method. This is the best result possible. Results on the ascending have the same ratio of changed artifacts marked suspect as non-changed artifacts marked suspect. This means the ratio of hits to false alarms in the set of artifacts marked suspect by the impact analysis method is the same as the ratio of changed to non-changed artifacts. The result of such a method is equal to pure chance and the worst possible impact analysis method.

Results below the ascending diagonal can also have good predictive capabilities. A result close to the lower right corner means the number of changed artifacts marked suspect and non-changed artifacts not marked suspect is close to zero. So almost all changed artifacts have been marked non-suspect and all non-changed artifacts have been marked suspect. So almost all changed artifacts have been identified as non-changed artifacts and all non-changed artifacts have been identified as changed artifacts. Inverting the labels of the test results leads to an almost perfect performance for this impact analysis method. So inverting the labels of the test results, causes the location of the result in the plot to be mirrored in the in the ascending diagonal, moving them from the area bellow the pure chance line to the area above it.

7.1.4 Change impact performance quantification example

The metrics and visualization options to quantify a tools impact analysis performance have been applied to two of the tools representing current requirements management tools. Both RequisitePro and CaliberRM use a method heavily relying on transitive closure of the traceability information, while TopTeam and DOORS do not offer this functionality or only with a lot of manual work. Therefore RequisitePro will represent the transitive closure methods and DOORS will represent the tools lacking this feature.

After performing the four change scenarios based on the CMS in the two tools, neither of the tools was able to predict exactly which artifacts had to be changed for any of the change scenarios. Three of the changes actually changed an artifact and the last change deleted an artifact. For each change both tools marked at least one non-changed artifact as suspect or a changed artifact as not suspect. It appears as if both tools do not support impact analysis for the deletion of artifacts, since none of the other artifacts were marked suspect. In the end there was no clear winner, since in two change scenarios both tools performed the same, while in the third case RequisitePro performed better, because of its support for transitive closure, and in the fourth case DOORS performed better, because of its lack of transitive closure.

The results of this evaluation of requirements management tools with support for traceability-based change impact analysis shows the basics for impact analysis are present in current tools through the support for information models, different link types, custom link types, link attributes, graphical representations, etc. A lot of this information is however not used during the impact analysis itself, leading to poor results. When the performance of the impact analysis methods is quantified, it shows none of the tools is able to predict the impact of any of the sample changes perfectly. Neither is there a clear winner, since the differences in their approach can be an advantage in one scenario and a disadvantage in the next. Therefore, the impact analysis in current requirements management tools is very limited and more effective methods are needed.

7.2 Results of the research questions

The research questions in the introduction of this report have been investigated throughout this report and the answers to these questions will be presented in this section.

What features in traceability based requirements management tools are needed to perform change impact analysis?
A list of criteria has been composed to determine what features are present in current tools to perform impact analysis. Not only the impact analysis feature itself is covered, but also the features needed to make impact analysis possible. Several studies have investigated what features are needed by requirements management tools and based on this literature a number of features important to impact analysis have been selected.

The first feature being tested is the support for an information model to describe the artifacts and their relations at the start of a project. To what extend is it possible to customize artifact types, relation types and the possible combinations of these two. Once the model is ready, the project data has to be loaded into the model. The test this feature the import and export options available in the tools are examined. Besides working with data from external documents it should also be possible to work with data from other tools in the software life cycle, therefore the features for tool integration are tested. To examine the relations between artifacts, used to inspect the results of impact analysis, options to depict the relations should be available, so features for graphical representations are tested. Before any reliable impact analysis can be performed, all artifacts should have links to previous and next stages in the software lifecycle. Coverage

analysis features are tested to see if this is supported. Finally the features available for impact analysis itself will be tested. The final set of criteria is presented in Table 2.

How can the performance of change impact analysis support in current tools be quantified?
Existing methods to quantify the performance of impact analysis methods do not support all types of results returned by a method or test the change itself instead of the impact analysis method. Binary classifiers and metrics based on them are used to quantify the performance of medical tests, production lines and information retrieval methods. Since the output of an impact analysis method is binary too, an artifact has to be changed or not, metrics based on binary classifiers can be used to measure an impact analysis methods performance too. Based on their relevance to impact analysis a selection of metrics has been made that should be able to get a clear image of the performance of any impact analysis method.

What is the status of impact analysis in current tools, based on these questions?
Four tools have been selected as a representation of current requirements management tools. The structure and requirements from the WASP case have been used to test if and how tools support the criteria. Modeling different requirement types, performing coverage analysis, detach links from their source document, the lack of support for automated link detection and the limited integration with other tools were the results all four tools had in common.

The other results differed for at least one of the tools. During import and export Requisite was the only tool maintaining a close link with the source document. The other tools lost this link the moment the artifacts were imported into the tool. All four tools support some form of matrix and graph views to inspect the different relations. The version in DOORS is however barely usable, while the other tools offer different levels of customizability to show the information needed. DOORS is the only tool supporting custom relations and attributes for relations. TopTeam offers three different types of relations, but their effect is the same and the remaining tools support only one generic relation.

Impact analysis is really limited since the analysis methods make no use of the different link types, when available. Two of the tools mark artifacts with direct links to the changed artifact suspect and have an option to also mark artifacts multiple links away. DOORS and TopTeam only mark artifacts with a direct link suspect and TopTeam has no option to do this automatically once an artifact is changed. Based on these results there is no clear winner. All of them have their problems, but the extended information model support of DOORS provides a better basis for impact analysis then the simple models offered by the other tools.

Since the most important difference in impact analysis support was the availability of transitive closure, the quantitative performance of only two of tools has been determined. RequisitePro was used to show the effect of transitive closure while DOORS showed the effect of a lack of transitive closure.

The CMS and the four changes being performed, as presented in [GKB08], have been used as the change scenarios to quantify the performance of both tools. Both tools performed well when working on a change targeted at their way of performing impact analysis. The change affecting a lot of indirect artifacts was predicted very well by RequisitePro, while DOORS missed some artifacts. The changes affecting only artifacts with a direct link to the changed artifact are predicted correctly by DOORS, while RequisitePro returned a lot of false alarms.

7.3 Discussion

The tools representing the current requirements management tools were only evaluated for their requirements and relations support. Support for multiple users, versioning, usability of the interface, etc. has not been investigated. Since these parts have not been evaluated, this is no complete requirements management tools evaluation covering all aspects of requirements management and should therefore not be used as such.

The CMS used quantify the performance of the tools, is artificial and designed for the analysis of traceability support in requirements management tools. The change scenarios were really simple, and cannot be compared to the scenarios presented in. The real life cases that were available to us, were not complete enough to be used with our metrics. None of the cases described the situation before and after a specific change, making it impossible to perform the impact analysis and compare it with the actually impacted requirements.

The current gold standard, used to determine which artifacts have or have not been changed, is predetermined. Since the current impact analysis methods used by the requirements management tools are very poor, this is no problem. In the future it is possible impact analysis methods come up with different solutions, making it necessary to investigate each impact analysis individually and not work with one predetermined solution. Since it will be possible the impact analysis proposed by a newer method is better in some way then the predetermined solution used for the evaluation. Evaluating each result individually is already used for the evaluation of medical tests.

The quantification of the performance using binary classifiers can be used for any impact analysis technique, not only the ones based on explicit traceability of requirements. As long as there is set with all available artifacts, a set with all artifacts changed in the final implementation and a set with all artifacts marked suspect by the impact analysis method the BC metrics can be applied. This makes it also usable for impact analysis based on probabilities or impact analysis for design or code elements.

7.4 Open issues

After the evaluation of the features and performance of requirements management tools, there are still a number of issues that still need attention. One of these issues was already mentioned in the discussion: how will the quantification methods work for a real world case. What is the effect of a case's artifact granularity, is the performance of the impact analysis method indeed measured or is the effect of the change itself again dominant? Once the quantification method has been investigated, what is the actual performance of

the tools using these real world cases. Are the results acceptable, as with our sample CMS, or are the Precision and Recall values so low, the impact analysis methods present in current tools are even worse then concluded after our evaluation.

Once a case study based on a real project has been used, it can also be used as the foundation for a balanced set of test scenarios to test the impact analysis methods used by tools supporting impact analysis. Some of the aspects that have to be covered by such a set of change scenarios:

- **Deep traces**, a change that affects a limited number of artifacts, but the artifacts affected reside on as much different levels of the information model as possible. This information can be used to test the transitive closure capabilities of a method.
- **Wide traces**, a change affecting a lot of artifacts directly, but does not go to deep. This information can be used to determine if a method is able to stop its transitive closure in time, but still return all effected artifacts.
- **Large relevant artifact sets**, most changes do not affect more then half of the artifacts, since they are probably very expensive. It is however interesting to see how impact analysis methods cope with such big changes and if they are not too conservative by default since this type of changes is unlikely.
- **Small relevant artifact sets**, since current have trouble identifying small changes were only one or two artifacts are affected, it is interesting to see how other methods cope with these small estimated impact sets.

At the moment the starting impact set is still part of the estimated impact set. This affects the performance of an impact analysis method. Since the artifacts in the starting set have been manually selected and the rest of the analysis is based on them, these artifacts must definitely be changed. This artificially increases the Recall and especially the Precision of an impact analysis method. An option to solve this problem is to leave the starting set out of the estimated set. This will however lead to a varying number of artifacts in the system set. It also causes problems when no additional artifacts were changed besides the ones in the starting set. This means the set with changed artifacts that are also marked suspect will be empty and both Precision and Recall will be zero as soon as at least one non-changed artifact is marked suspect or one changed artifact is not marked suspect. Precision and Recall will be undefined (infinite) as soon as the result is perfect, since there will be a division by zero.

References

[AB93] Arnold, R.S. and Bohner, S.A., Impact analysis - Towards a framework for comparison. in *Proceedings of Conference on Software Maintenance, 1993. CSM-93*, (1993), 292-301.

[AK03] Atkinson, C. and Kuhne, T. Model-driven development: a metamodeling foundation. *Software, IEEE*, *20* (5). 36-41.

[Alv02] Alvarez, S.A. An exact analytical relation among recall, precision, and classification accuracy in information retrieval. *Boston College, Boston, Technical Report BCCS-02-01*.

[ARN06] Aizenbud-Reshef, N., Nolan, B.T., Rubin, J. and Shaham-Gafni, Y. Model traceability. *IBM Syst. J.*, *45* (3). 515-526.

[BA96] Bohner, S.A. and Arnold, R.S. *Software change impact analysis*. IEEE Computer Society Press, Los Alamitos, Calif., 1996.

[Boh02] Bohner, S.A., Software change impacts-an evolving perspective. in, (2002), 263-272.

[Cam09] Cambridge University Press. Cambridge Dictionaries Online - http://dictionary.cambridge.org/, 2007.

[CFV99] Cimitile, A., Fasolino, A.R. and Visaggio, G., A software model for impact analysis: a validation experiment. in *Reverse Engineering, 1999. Proceedings. Sixth Working Conference on*, (1999), 212-222.

[CHC03] Cleland-Huang, J., Chang, C.K. and Christensen, M. Event-based traceability for managing evolutionary change. *IEEE Transactions on Software Engineering*, *29* (9). 796-810.

[CMS06] Cant, T., McCarthy, J. and Stanley, R. Tools for Requirements Management: a Comparison of Telelogic DOORS and the HIVE, Australian Government Department of Defense, Defence Science and Technology Organisation, 2006.

[DG06] Davis, J. and Goadrich, M., The relationship between precision-recall and ROC curves. in *23rd International Conference on Machine Learning*, (Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2006), ACM New York, NY, USA, 233-240.

[Dic02] Dick, J. Rich Traceability. *Proceedings Automated Software Engineering*.

[Dic07] Dictionary.com LLC. Dictionary.com - http://dictionary.reference.com/, 2007.

[DP98] Domges, R. and Pohl, K. Adapting traceability environments to project-specific needs. *Commun. ACM*, *41* (12). 54-62.

[EDF96] Ecklund Jr, E.F., Delcambre, L.M.L. and Freiling, M.J., Change cases: use cases that identify future requirements. in, (1996), ACM New York, NY, USA, 342-358.

[EKS02] Ebben, P., Koolwaaij, J.W., Setten, M.v. and Wibbels, M. Requirements for the WASP Application Platform, Telematica Instituut, Enschede, Netherlands, 2002.

[Faw04] Fawcett, T. ROC graphs: Notes and practical considerations for researchers. *Machine Learning*, *31*.

[Fla04] Flach, P.A., The Many Faces of ROC Analysis in Machine Learning. in *The Twenty-First International Conference on Machine Learning*, (Banff, Alberta, Canada, 2004).

[FV99] Fasolino, A.R. and Visaggio, G., Improving Software Comprehension through an Automated Dependency Tracer. in, (1999), 58Ð65.

[GF94] Gotel, O. and Finkelstein, A., An Analysis of the Requirements Traceability Problem. in *Proceedings of the First International Conference on Requirements Engineering*, (Colorado Springs, CO, USA, 1994), 94-101.

[GKB08] Goknil, A., Kurtev, I. and van den Berg, K., Change Impact Analysis based on Formalization of Trace Relations for Requirements. in *ECMDA Traceability Workshop (ECMDA-TW)*, (Berlin, Germany, 2008), 59-75.

[GSC04] Greenfield, J., Short, K., Cook, S. and Kent, S. *Software factories*. Wiley, 2004.

[HGF08] Hattori, L., Guerrero, D., Figueiredo, J., Brunet, J. and Damasio, J., On the Precision and Accuracy of Impact Analysis Techniques. in *7th IEEE/ACIS International Conference on Computer and Information Science*, (Portland, Oregon, USA, 2008), IEEE Computer Society.

[HKW04] Hoffmann, M., Kuhn, N., Weber, M. and Bittner, M. Requirements for requirements management tools. *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*. 301-308.

[How07] Howe, D. Free On-line Dictionary of Computing - http://foldoc.org/, 2007.

[IBM06] Corporation, I. Rational RequisitePro 7 Online Help, 2006.

[IID06] Ibrahim, S., Idris, N.B., Deraman, A. and Selangor, M., A software traceability validation for change impact analysis of object oriented software. in *International Conference on Software Engineering Research and Practice*, (Las Vegas, Nevada, USA, 2006), 453-459.

[Kne02] von Knethen, A. and Paech, B. A Survey on Tracing Approaches in Practice and Research *IESE-Report No. 095.01/E*, Fraunhofer IESE, 2002, 41.

[Kur05] Kurtev, I. Adaptability of model transformations. *Faculty of Electrical Engineering, Mathematics & Computer Science, UT*, *8*.

[LS98] Lindvall, M. and Sandahl, K. Traceability aspects of impact analysis in object-oriented systems. *Journal of Software Maintenance: Research and Practice*, *10* (1).

[LV00] Lavazza, L. and Valetto, G., Enhancing requirements and change management through processmodelling and measurement. in *Proceedings 4th International Conference on Requirements Engineering*, (Schaumburg, IL, USA, 2000), 106-115.

[Mor96] Moreton, R. A Process Model for Software Maintenance Software. *Change Impact Analysis, IEEE Computer Society*. 29-33.

[MW09] Merriam-Webster, I. Merriam-Webster Online Dictionary - http://www.merriam-webster.com/dictionary/, 2009.

[OC01] O'Neal, J. and Carver, D., Analyzing the Impact of Changing Requirements. in, (2001), 190-195.

[OR23] Ogden, C.K. and Richards, I.A. *The meaning of meaning: a study of the influence of language upon thought and of the science of symbolism*. Harcourt, Brace & World, Inc., New York, New York, USA, 1923.

[Pfl98] Pfleeger, S.L. *Software engineering: theory and practice*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1998.

[PG96] Pinheiro, F.A.C. and Goguen, J.A. An object-oriented tool for tracing requirements. *IEEE Software*, *13* (2). 52-64.

[Pin03] Pinheiro, F.A.C. Requirements Traceability. in Prado Leite, J.C.S.d. and Doorn, J.H. eds. *Perspectives on Software Requirements*, Kluwer Academic Publishers, Assinippi Park, MA, USA, 2003, 91-113.

[Rijs79] Van Rijsbergen, C.J. Information Retrieval. Dept. of Computer Science, University of Glasgow, Butterworth, London, 1979.

[RJ01] Ramesh, B. and Jarke, M. Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering*, *27* (1). 58-93.

[RLL98] Rowe, D., Leaney, J. and Lowe, D. Defining systems evolvability – a taxonomy of change. *Proceeding of the IEEE Conference on Computer Based Systems*. 45-52.

[RR97] Robertson, J. and Robertson, S. Volere requirements specification template. *Atlantic Systems Guild*.

[Som89] Sommerville, I. *Software Engineering*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[SS09] Sparx Systems Reviewers Guide to Enterprise Architect 7.5.

[Wie03] Wieringa, R. *Design Methods for Reactive Systems: Yourdan, Statemate, and the UML*. Morgan Kaufmann Publishers, 2003.

[Wie95] Wieringa, R.J. *An Introduction to Requirements Traceability*. Vrije Universiteit, Faculty of Mathematics and Computer Science, Amsterdam, 1995.

[WW02] Weber, M. and Weisbrod, J. Requirements engineering in automotive development-experiences and challenges. *Proceedings. IEEE Joint International Conference on Requirements Engineering, 2002.* 331-340.

Appendix A Requirements Course Management System

The CMS used to determine the performance of the requirements management tools and described in Section 5.3 contains the following sixteen requirements:

Req1: The system shall allow end-users to provide profile and context information for registration.

Req2: The system shall provide functionality to search for other people registered in the system.

Req3: The system shall provide functionality to allow end-users to log in the system with their password.

Req4: The system shall support three types of end-users (administrator, lecturer and student).

Req5: The system shall allow lecturers to set an alert on an event.

Req6: The system shall maintain a list of events the students can be notified about.

Req7: The system shall notify the students about the occurrence of an event as soon as the event occurs.

Req8: The system shall actively monitor all events.

Req9: The system shall notify students about the events of the lectures they are enrolled for.

Req10: The system shall allow students to enroll for lecturers.

Req11: The system shall allow lecturers to send e-mail to students enrolled for the lecture given by that lecturer.

Req12: The system shall allow assigning students to teams for each lecture.

Req13: The system shall allow lecturers to send e-mail to students in the same group.

Req14: The system shall allow lecturers to modify the content of the lectures.

Req15: The system shall give different access rights to different types of end-users.

Req16: The system shall support two types of end-users (lecturer and student) and it shall provide functionality to allow end-users to log in the system with their password.

Appendix B Concrete sets used for impact analysis

The CMS example is implemented in DOORS and RequisitePro. DOORS is chosen because it seems to be the tool with the best options and RequisitePro has transitivity support for links. TopTeam and CaliberRM have no special impact analysis features compared to DOORS and RequisitePro and therefore they are not used for the example case.

B.1 DOORS sets

This section enlists the sets of effected requirements for each change scenario when implemented in DOORS.

B.1.1 Delete Req6

The deletion of Req6 is the first change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993

This table contains the sets described in the original paper by Arnold and Bohner.

Set	**Effected requirements**	#
Universe	∞	∞
System	(all)	16
SIS	Req6	1
EIS	Req6	1
AIS	Req6,Req5,Req7,Req9	4

Fasolino 1999

This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	**Effected requirements**	#
System	(all)	16
PIS (primary)	Req6	1
SIS (secondary)	(none)	0
EIS (=PIS+SIS)	Req6	1
AIS	Req6,Req5,Req7,Req9	4

Bohner 2002

This table contains the updated set of sets as specified by Bohner.

Set	**Effected requirements**	#
System	(all)	16
SIS	Req6	1
CIS	Req6	1
AIS	Req6,Req5,Req7,Req9	4
FPIS	(none)	0
DIS	Req5,Req7,Req9	3

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	**Effected requirements**	#
Identified elements	Req6	1
Impacted elements	undeterminable	undeterminable
Updated elements	Req6,Req5,Req7,Req9	4

B.1.2 Modify Req6

Since removing an artifact from most tools, does not trigger the IA process, the to be deleted artifact is first changed to trigger the IA process. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the sets described in the original paper by Arnold and Bohner.

Set	**Effected requirements**	#
Universe:	∞	∞
System:	(all)	16
SIS:	Req6	1
EIS:	Req6,Req4,Req5,Req7	4
AIS:	Req6,Req5,Req7,Req9	4

Fasolino 1999
This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	**Effected requirements**	#
System	(all)	16
PIS (primary)	Req6	1
SIS (secondary)	Req4,Req5,Req7	3
EIS (=PIS+SIS)	Req6,Req4,Req5,Req7	4
AIS	Req6,Req5,Req7,Req9	4

Bohner 2002
This table contains the updated set of sets as specified by Bohner.

Set	**Effected requirements**	#
System	(all)	16
SIS	Req6	1
CIS	Req6,Req4,Req5,Req7	4
AIS	Req6,Req5,Req7,Req9	4
FPIS	Req4	1
DIS	Req9	1

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	Effected requirements	#
Identified elements:	Req6,Req4,Req5,Req7	4
Impacted elements:	undeterminable	undeterminable
Updated elements:	Req6,Req5,Req7,Req9	4

B.1.3 Modify Req9

The modification of Req9 is the third change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the sets described in the original paper by Arnold and Bohner.

Set	Effected requirements	#
Universe:	∞	∞
System:	(all)	16
SIS:	Req9	1
EIS:	Req9,Req7,Req10	3
AIS:	Req9	1

Fasolino 1999
This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	Effected requirements	#
System	(all)	16
PIS (primary)	Req9	1
SIS (secondary)	Req7,Req10	2
EIS (=PIS+SIS)	Req9,Req7,Req10	3
AIS	Req9	1

Bohner 2002
This table contains the updated set of sets as specified by Bohner.

Set	Effected requirements	#
System	(all)	16
SIS	Req9	1
CIS	Req9,Req7,Req10	3
AIS	Req9	1
FPIS	Req7,Req10	2
DIS	(none)	0

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	Effected requirements	#
Identified elements:	Req9,Req7,Req10	3

Impacted elements:	undeterminable	undeterminable
Updated elements:	Req9	1

B.1.4 Modify Req11

The modification of Req11 is the fourth and last change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993

This table contains the sets described in the original paper by Arnold and Bohner.

Set	**Effected requirements**	#
Universe:	∞	∞
System:	(all)	16
SIS:	Req11	1
EIS:	Req11,Req10,Req13	3
AIS:	Req11,Req13	2

Fasolino 1999

This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	**Effected requirements**	#
System	(all)	16
PIS (primary)	Req11	1
SIS (secondary)	Req10,Req13	2
EIS (=PIS+SIS)	Req11,Req10,Req13	3
AIS	Req11,Req13	2

Bohner 2002

This table contains the updated set of sets as specified by Bohner.

Set	**Effected requirements**	#
System	(all)	16
SIS	Req11	1
CIS	Req11,Req10,Req13	3
AIS	Req11,Req13	2
FPIS	Req10	1
DIS	(none)	0

Cleland-Huang 2003

This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	**Effected requirements**	#
Identified elements:	Req11,Req10,Req13	3
Impacted elements:	undeterminable	undeterminable
Updated elements:	Req11,Req13	2

B.2 DOORS metrics

This section enlists the results of the application of the metrics based on binary classifiers using the sets specified in the previous section for DOORS.

B.2.1 Delete Req6

The deletion of Req6 is the first change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993

This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	**Metric**	#
SIS and EIS	|SIS| / |EIS|: 1/1 =	1
EIS and System (sharpness)	|EIS| / |System|: 1/16 =	1/16
EIS and AIS	|AIS| / |EIS|: 4/1 =	4
EIS and AIS	|EIS| / |AIS|: 1/4 =	1/4
EIS and AIS	|EIS ∩ AIS|: |Req6| =	1
Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999

This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	**Metric**	#
Adequacy	Inclusiveness = AIS ⊆ EIS: false	0
Ripple-effect	Amplification = |SIS| / |PIS|: 0/1	0
Sharpness	ChangeRate = |EIS| / |System|: 1/16	1/16
Adherence	S-Ratio = |AIS| / |EIS|: 4/1	4

Bohner 2002

This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	**Metric**	#
Accuracy	Error = (|DIS| + |FPIS|) / |CIS|: (3+0) / 1	3

Cleland-Huang 2003

This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	**Metric**	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification
This table contains the results of the metrics based on binary classifiers.

Effect	Metric	#
Precision	|AIS ∩ EIS| / |EIS| = |Req6| / |Req6|	1
Recall	|AIS ∩ EIS| / |AIS| = |Req6| / |Req6,Req5,Req7,Req9|	1/4
Fall-out	|FPIS| / (|System| – |AIS|) = 0/(16-4)	0
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-3/16	13/16

B.2.2 Modify Req6

Since removing an artifact from most tools, does not trigger the IA process, the to be deleted artifact is first changed to trigger the IA process. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	Metric	#
SIS and EIS	|SIS| / |EIS|: 1/4 =	1/4
EIS and System (sharpness)	|EIS| / |System|: 4/16 =	1/4
EIS and AIS	|AIS| / |EIS|: 4/4 =	1
EIS and AIS	|EIS| / |AIS|: 4/4 =	1
EIS and AIS	|EIS ∩ AIS|: |Req6,Req5,Req7| =	3
Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999
This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	Metric	#
Adequacy	Inclusiveness = AIS ⊆ EIS: false	0
Ripple-effect	Amplification = SIS# / PIS# -> 1: 3/1 =	3
Sharpness	ChangeRate = EIS# / System#: 4/16 =	1/4
Adherence	S-Ratio = AIS# / EIS#: 4/4 =	1

Bohner 2002
This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	Metric	#
Accuracy:	Error = (|DIS| + |FPIS|) / |CIS|: 1+1/4 =	1/2

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	Metric	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification
This table contains the results of the metrics based on binary classifiers.

Effect	Metric	#
Precision	|AIS ∩ EIS| / |EIS| = |Req6,Req5,Req7| / |EIS|	3/4
Recall	|AIS ∩ EIS| / |AIS| = |Req6,Req5,Req7| / |AIS|	3/4 (9/12)
Fall-out	|FPIS| / (|System| – |AIS|) = 1/(16-4)	1/12
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-2/16	14/16

B.2.3 Modify Req9

The modification of Req9 is the third change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	Metric	#
SIS and EIS	|SIS| / |EIS|: 1/3 =	1/3
EIS and System (sharpness)	|EIS| / |System|: 3/16 =	3/16
EIS and AIS	|AIS| / |EIS|: 1/3 =	1/3
EIS and AIS	|EIS| / |AIS|: 3/1 =	3
EIS and AIS	|EIS ∩ AIS|: |Req9| =	1
Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999
This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	Metric	#
Adequacy	Inclusiveness = AIS ⊆ EIS: true	1
Ripple-effect	Amplification = SIS# / PIS# -> 1: 2/1 =	2
Sharpness	ChangeRate = EIS# / System#: 3/16 =	3/16
Adherence	S-Ratio = AIS# / EIS#: 1/3 =	1/3

Bohner 2002
This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	Metric	#
Accuracy	Error = (|DIS| + |FPIS|) / |CIS|: 0+2/3 =	2/3

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	Metric	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification
This table contains the results of the metrics based on binary classifiers.

Effect	Metric	#
Precision	|AIS ∩ EIS| / |EIS| = |Req9| / |Req9,Req7,Req10|	1/3
Recall	|AIS ∩ EIS| / |AIS| = |Req9| / |Req9|	1
Fall-out	|FPIS| / (|System| – |AIS|) = 2/(16-1)	2/15
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-2/16	7/8

B.2.4 Modify Req11

The modification of Req11 is the fourth and last change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	Metric	#
SIS and EIS	|SIS| / |EIS|: 1/3 =	1/3
EIS and System (sharpness)	|EIS| / |System|: 3/16 =	3/16
EIS and AIS	|AIS| / |EIS|: 2/3 =	2/3
EIS and AIS	|EIS| / |AIS|: 3/2 =	3/2
EIS and AIS	|EIS ∩ AIS|: |Req11, Req13| =	2
Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999
This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	Metric	#
Adequacy	Inclusiveness = AIS ⊆ EIS: true	1
Ripple-effect	Amplification = SIS# / PIS# -> 1: 2/1 =	2
Sharpness	ChangeRate = EIS# / System#: 3/16 =	3/16
Adherence	S-Ratio = AIS# / EIS#: 2/3 =	2/3

Bohner 2002
This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	Metric	#
Accuracy	Error = (|DIS| + |FPIS|) / |CIS|: 0+1/3 =	1/3

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	Metric	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification
This table contains the results of the metrics based on binary classifiers.

Effect	Metric	#
Precision	|AIS ∩ EIS| / |EIS| = |Req11,Req13| / |Req11,Req10,Req13|	2/3
Recall	|AIS ∩ EIS| / |AIS| = |Req11,Req13| / | Req11,Req13|	1
Fall-out	|FPIS| / (|System| – |AIS|) = 1/(16-2)	1/14
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-1/16	15/16

B.3 RequisitePro sets

This section enlists the sets of effected requirements for each change scenario when implemented in DOORS. The option to show indirect traces is enabled when available and all artifacts identified with these settings are counted.

B.3.1 Delete Req6

The deletion of Req6 is the first change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the sets described in the original paper by Arnold and Bohner.

Set	Effected requirements	#
Universe:	∞	∞
System:	(all)	16
SIS:	Req6	1
EIS:	Req6	1
AIS:	Req6,Req5,Req7,Req9	4

Fasolino 1999
This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	Effected requirements	#
System	(all)	16
PIS (primary)	Req6	1
SIS (secondary)	(none)	0
EIS (=PIS+SIS)	Req6	1
AIS	Req6,Req5,Req7,Req9	4

Bohner 2002
This table contains the updated set of sets as specified by Bohner.

Set	Effected requirements	#
System	(all)	16
SIS	Req6	1
CIS	Req6	1
AIS	Req6,Req5,Req7,Req9	4
FPIS	(none)	0
DIS	Req5,Req7,Req9	3

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	Effected requirements	#
Identified elements:	Req6	1
Impacted elements:	undeterminable	undeterminable
Updated elements:	Req6,Req5,Req7,Req9	4

B.3.2 Modify Req6

Since removing an artifact from most tools, does not trigger the IA process, the to be deleted artifact is first changed to trigger the IA process. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the sets described in the original paper by Arnold and Bohner.

Set	Effected requirements	#
Universe:	∞	∞

System:	(all)	16
SIS:	Req6	1
EIS:	Req6,Req4,Req5,Req7,Req9	5
AIS:	Req6,Req5,Req7,Req9	4

Fasolino 1999
This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	**Effected requirements**	#
System	(all)	16
PIS (primary)	Req6	1
SIS (secondary)	Req4,Req5,Req7,Req9	4
EIS (=PIS+SIS)	Req6,Req4,Req5,Req7,Req9	5
AIS	Req6,Req5,Req7,Req9	4

Bohner 2002
This table contains the updated set of sets as specified by Bohner.

Set	**Effected requirements**	#
System	(all)	16
SIS	Req6	1
CIS	Req6,Req4,Req5,Req7,Req9	5
AIS	Req6,Req5,Req7,Req9	4
FPIS	Req4	1
DIS	(none)	0

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	**Effected requirements**	#
Identified elements:	Req6,Req4,Req5,Req7,Req9	5
Impacted elements:	undeterminable	undeterminable
Updated elements:	Req6,Req5,Req7,Req9	4

B.3.3 Modify Req9

The modification of Req9 is the third change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the sets described in the original paper by Arnold and Bohner.

Set	**Effected requirements**	#
Universe:	∞	∞
System:	(all)	16
SIS:	Req9	1
EIS:	Req9,Req7,Req10,Req6,Req8,Req4	6
AIS:	Req9	1

Fasolino 1999
This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	Effected requirements	#
System	(all)	16
PIS (primary)	Req9	1
SIS (secondary):	Req7,Req10,Req6,Req8,Req4	5
EIS:	Req9,Req7,Req10,Req6,Req8,Req4	6
AIS	Req9	1

Bohner 2002
This table contains the updated set of sets as specified by Bohner.

Set	Effected requirements	#
System	(all)	16
SIS	Req9	1
CIS:	Req9,Req7,Req10,Req6,Req8,Req4	6
AIS	Req9	1
FPIS:	Req7,Req10,Req6,Req8,Req4	5
DIS	(none)	0

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	Effected requirements	#
Identified elements:	Req9,Req7,Req10,Req6,Req8,Req4	6
Impacted elements:	undeterminable	undeterminable
Updated elements:	Req9	1

B.3.4 Modify Req11

The modification of Req11 is the fourth and last change scenario. The following sets were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the sets described in the original paper by Arnold and Bohner.

Set	Effected requirements	#
Universe:	∞	∞
System:	(all)	16
SIS:	Req11	1
EIS:	Req11, Req10,Req13	3
AIS:	Req11,Req13	2

Fasolino 1999
This table contains the sets described by Fasolino based on sets defined by Bohner.

Set	Effected requirements	#
System	(all)	16
PIS (primary)	Req11	1

SIS (secondary)	Req10,Req13	2
EIS (=PIS+SIS)	Req11, Req10,Req13	3
AIS	Req11,Req13	2

Bohner 2002
This table contains the updated set of sets as specified by Bohner.

Set	**Effected requirements**	#
System	(all)	16
SIS	Req11	1
CIS	Req11, Req10,Req13	3
AIS	Req11,Req13	2
FPIS	Req10	1
DIS	(none)	0

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. Since the Impacted elements set was not defined clearly, this set could not be determined.

Set	**Effected requirements**	#
Identified elements:	Req11, Req10,Req13	3
Impacted elements:	undeterminable	undeterminable
Updated elements:	Req11,Req13	2

B.4 RequisitePro metrics

This section enlists the results of the application of the metrics based on binary classifiers using the sets specified in the previous section for RequisitePro.

B.4.1 Delete Req6

The deletion of Req6 is the first change scenario. The following sets and numbers were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	**Metric**	#
SIS and EIS	|SIS| / |EIS|: 1/1 =	1
EIS and System (sharpness)	|EIS| / |System|: 1/16 =	1/16
EIS and AIS	|AIS| / |EIS|: 4/1 =	4
EIS and AIS	|EIS| / |AIS|: 1/4 =	1/4
EIS and AIS	|EIS ∩ AIS|: |Req6| =	1
Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999
This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	Metric	#
Adequacy	Inclusiveness = AIS ⊆ EIS: false	0
Ripple-effect	Amplification = SIS# / PIS# -> 1: 0/1 =	0
Sharpness	ChangeRate = EIS# / System#: 1/16 =	1/16
Adherence	S-Ratio = AIS# / EIS#: 4/1 =	4

Bohner 2002
This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	Metric	#
Accuracy	Error = (|DIS| + |FPIS|) / |CIS|: 3+0/1 =	3

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	Metric	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification
This table contains the results of the metrics based on binary classifiers.

Effect	Metric	#
Precision	|AIS ∩ EIS| / |EIS| = |Req6| / |Req6|	1
Recall	|AIS ∩ EIS| / |AIS| = |Req6| / |Req6,Req5,Req7,Req9|	1/4
Fall-out	|FPIS| / (|System| – |AIS|) = 0/(16-4)	0
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-3/16	13/16

B.4.2 Modify Req6

Since removing an artifact from most tools, does not trigger the IA process, the to be deleted artifact is first changed to trigger the IA process. The following sets and numbers were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	Metric	#
SIS and EIS	|SIS| / |EIS|: 1/5 =	1/5
EIS and System (sharpness)	|EIS| / |System|: 5/16 =	5/16
EIS and AIS	|AIS| / |EIS|: 4/5 =	4/5
EIS and AIS	|EIS| / |AIS|: 5/4 =	5/4
EIS and AIS	|EIS ∩ AIS|: |Req6,Req5,Req7,Req9| =	4

Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999
This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	**Metric**	#
Adequacy	Inclusiveness = AIS ⊆ EIS: false	0
Ripple-effect	Amplification = SIS# / PIS# -> 1: 4/1 =	4
Sharpness	ChangeRate = EIS# / System#: 5/16 =	5/16
Adherence	S-Ratio = AIS# / EIS#: 4/5 =	4/5

Bohner 2002
This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	**Metric**	#
Accuracy	Error = (|DIS| + |FPIS|) / |CIS|: 0+1/5 =	1/5

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	**Metric**	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification

Effect	**Metric**	#
Precision	|AIS ∩ EIS| / |EIS| = |Req6,Req5,Req7,Req9| / |EIS|	4/5
Recall	|AIS ∩ EIS| / |AIS| = | Req6,Req5,Req7,Req9| / |AIS|	1
Fall-out	|FPIS| / (|System| – |AIS|) = 1/(16-4)	1/12
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-1/16	15/16

B.4.3 Modify Req9

The modification of Req9 is the third change scenario. The following sets and numbers were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	**Metric**	#
SIS and EIS	|SIS| / |EIS|: 1/6 =	1/6

EIS and System (sharpness)	|EIS| / |System|: 6/16 =	6/16
EIS and AIS	|AIS| / |EIS|: 1/6 =	1/6
EIS and AIS	|EIS| / |AIS|: 6/1 =	6
EIS and AIS	|EIS ∩ AIS|: |Req9| =	1
Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999
This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	**Metric**	#
Adequacy	Inclusiveness = AIS ⊆ EIS: true	1
Ripple-effect	Amplification = SIS# / PIS# -> 1: 5/1 =	5
Sharpness	ChangeRate = EIS# / System#: 6/16 =	6/16
Adherence	S-Ratio = AIS# / EIS#: 1/6 =	1/6

Bohner 2002
This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	**Metric**	#
Accuracy	Error = (|DIS| + |FPIS|) / |CIS|: 0+5/6 =	5/6

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	**Metric**	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification
This table contains the results of the metrics based on binary classifiers.

Effect	**Metric**	#
Precision	|AIS ∩ EIS| / |EIS| = |Req9| / |EIS|	1/6
Recall	|AIS ∩ EIS| / |AIS| = |Req9| / |Req9|	1
Fall-out	|FPIS| / (|System| – |AIS|) = 5/(16-1)	5/15
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-5/16	11/16

B.4.4 Modify Req11

The modification of Req11 is the fourth and last change scenario. The following sets and numbers were the result of the impact analysis and implementation of the change.

Bohner 1993
This table contains the results for the metrics described in the original paper by Arnold and Bohner.

Effect	Metric	#
SIS and EIS	|SIS| / |EIS|: 1/3 =	1/3
EIS and System (sharpness)	|EIS| / |System|: 3/16 =	3/16
EIS and AIS	|AIS| / |EIS|: 2/3 =	2/3
EIS and AIS	|EIS| / |AIS|: 3/2 =	3/2
EIS and AIS	|EIS ∩ AIS|: |Req11, Req13| =	2
Granularity	Granularity of artifact and interface model	undeterminable

Fasolino 1999
This table contains the results for the metrics described by Fasolino based on sets and metrics defined by Bohner.

Effect	Metric	#
Adequacy	Inclusiveness = AIS ⊆ EIS: true	1
Ripple-effect	Amplification = SIS# / PIS# -> 1: 2/1 =	2
Sharpness	ChangeRate = EIS# / System#: 3/16 =	3/16
Adherence	S-Ratio = AIS# / EIS#: 2/3 =	2/3

Bohner 2002
This table contains the additional metric based on the updated set of sets specified by Bohner.

Effect	Metric	#
Accuracy	Error = (|DIS| + |FPIS|) / |CIS|: 0+1/3 =	1/3

Cleland-Huang 2003
This table contains the sets as defined by Cleland-Huang. But since the way to obtain the Impacted set is not specified and can therefore not be determined, it is also impossible to use the metrics, since all metrics make use of the Impacted set.

Effect	Metric	#
No errors	Impacted = Identified = Updated	undeterminable
Identification errors	Impacted ⊃ Identified	undeterminable
Update errors	Impacted ⊃ Updated	undeterminable
Inclusion errors	Identified ∩ Impacted ≠ ∅	undeterminable

Binary Classification
This table contains the results of the metrics based on binary classifiers.

Effect	Metric	#
Precision	|AIS ∩ EIS| / |EIS| = |Req11,Req13| / |Req11,Req10,Req13|	2/3
Recall	|AIS ∩ EIS| / |AIS| = |Req11,Req13| / | Req11,Req13|	1
Fall-out	|FPIS| / (|System| – |AIS|) = 1/(16-2)	1/14
Accuracy	|System| – |FPIS ∪ DIS| / |System| = 16-1/16	15/16