A Design Theory For Requirements Mining Systems

Inauguraldissertation

zur Erlangung des akademischen Grades eines Doktors der Wirtschaftswissenschaften der Universität Mannheim

vorgelegt

von

Hendrik Meth

im FSS 2013

Dr. Jürgen M. Schneider (Dekan)
Prof. Dr. A. Mädche (Referent)
Prof. Dr. A. Heinzl (Korreferent)
10.06.2013 (Tag der mündlichen Prüfung)

für Marga

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
DF	Design Feature
DG	Decisional Guidance
DP	Design Principle
DR	Design Requirement
DREPT	Design Relevant Explanatory/ Predictive Theory
DSR	Design Science Research
DSS	Decision Support Systems
ERP	Enterprise Resource Planning
GMDSR	General Methodology of Design Science Research
IR	Information Retrieval
IS	Information Systems
ISDT	Information System Design Theory
IT	Information Technology
JDBC	Java Database Connectivity
JSF	Java Server Faces
M	Mean
NFR	Non-Functional Requirement
NLP	Natural Language Processing
NLR	Natural Language Requirement
NLRR	Natural Language Requirements Resource
ODBC	Open Database Connectivity
POS	Part-Of-Speech
RDS	Requirements Discovery Systems
RE	Requirements Engineering
RMANCOVA	Repeated Measures of Analysis of Covariance
RMANOVA	Repeated Measures of Analysis of Variance
RMS	Requirements Mining Systems

SD	Standard Deviation
SPSS	Statistical Package for the Social Sciences
SQL	Structured Query Language
UML	Unified Modeling Language
XML	Extensible Markup Language

1 Introduction

1.1 Motivation

In consequence of the pervasive existence of information technology in modern life, the development of software became increasingly important within the software industry and other industrial sectors. Contemporary software development is confronted with significant challenges including increased innovation, cost and time pressure, soaring complexity and high quality demands (Pohl 2010). Many software development projects cannot cope with these challenges. According to a recent study, issued by the Standish Group, only 32% of all software development projects are finished successfully, while the remaining projects invest more resources than planned, reduce their original functional scope or entirely fail (Standish 2009).

The success of IS[1] development highly depends on the accuracy of the requirements gathered from users and other stakeholders (Appan and Browne 2012; Hickey and Davis 2004). Requirements which have been overlooked, misinterpreted or incompletely specified can cause high costs. Boehm and Basili (2001) estimate that the detection and removal of a software problem after delivery is 100 times more expensive than the correction of a problem during the requirements or design phase. Therefore, the efficient determination of complete and correct software requirements is of utmost importance.

Approximately 80% of software requirements are recorded in natural language (Mich et al. 2004; Neill and Laplante 2003), within informal requirements documents, interview transcripts, discussion forums, or narrative scenarios. Natural language is inherently powerful and expressive and can thus be used to communicate between a broad range of stakeholders and users (Casamayor et al. 2011). Even though it appears to be a well-suited means to articulate and discuss requirements, severe problems emerge when using natural language in specification documents as they might be ambiguous, inconsistent and incomplete (Wilson et al. 1997). Moreover, a direct interpretation of these documents by subsequent development tools is almost impossible. Accordingly,

[1] Information Systems.

natural language requirements are usually transformed from initially informal statements into more consistent and unambiguous representations (Tichy and Koerner 2010). This process is referred to as *requirements discovery* in the context of this doctoral thesis[2].

Especially in large IS development projects, requirements discovery is a challenging task as a huge number of natural language requirements becomes available and needs to be analyzed. In these cases, manual requirements discovery can become time-consuming, error-prone, and monotonous, especially if it has to be repeated multiple times when updates to previously existing documents become available (Ambriola and Gervasi 2006; Huffman Hayes et al. 2005). These problems lead to a low individual performance and more specifically to a low productivity of requirements engineers involved in this process. As a consequence, the question can be raised if and how requirements discovery can be supported by software development systems.

1.2 Research Goals

Many systems have been suggested to support requirements discovery by the means of technology (Ambriola and Gervasi 2006; Casamayor et al. 2010; Cleland-Huang et al. 2007; Gacitua et al. 2011) and ultimately to improve requirements engineers' productivity. Additionally to a first identification of requirements or requirements abstractions, these systems also support different processing steps such as requirements interrelation (Ambriola and Gervasi 2006; Harmain and Gaizauskas 2003; Sampaio et al. 2007) or requirements classification (Casamayor et al. 2010; Cleland-Huang et al. 2007; Vlas and Robinson 2012). The latter class of systems (systems to support requirements identification and classification) is focused in the context of this thesis and referred to as *Requirements Mining Systems* (RMS).

Although former works made major progress in the technical development of RMS, few efforts have been made to systematically capture the prescriptive knowledge gained. An according codification and abstraction of results in a design theory could significantly extend the requirements discovery knowledge base and guide future research in this

[2] In the following, this doctoral thesis will be simply referred to as „thesis".

area. To increase the probability of an effective design, this theory should be grounded on practical experiences in the area of requirements discovery on the one hand and existing kernel theories which are relevant in this context on the other. Furthermore, existing RMS have been mainly evaluated through simulations, comparing the results of the presented system with a previously defined gold standard. Even though these evaluations allow precise measurements of absolute quality criteria, they do not allow a comparison to the as-is situation of manual discovery. More specifically, the question if RMS improve a requirements engineer's individual productivity is hardly answered yet.

As a consequence, this research project aims at 1) deriving a theoretically grounded design theory for RMS 2) implementing an artifact based on this design theory and 3) evaluating if requirements mining supported by this artifact results in increased productivity (in comparison to manual discovery). The leading research question to attain these goals is: *How can a system be designed which aims at improving requirements mining productivity over manual discovery?*

Following a Design Science approach, the theory which shall be derived is structured according to the eight components of a design theory suggested by Gregor and Jones (2007). Design requirements are identified based on general knowledge and kernel theories, design principles are conceptualized and mapped to design features which are then instantiated in an artifact. The artifact is used to measure effects of the identified design principles on requirements mining productivity in two experiments: one in a laboratory and one in a field setting. This thesis contributes to the design theory body of knowledge by providing a design theory for RMS. The design theory is a contribution to the IS literature because RMS represent an important class of design situations that have not been adequately described yet by existing works. From a practical point of view, the study can help commercial providers of requirements engineering software packages in the design of their applications. Applied to commercial software development, the design theory can guide developers by reducing the range of possible system features and development activities to a more manageable set, and thus increase the probability of success.

1.3 Structure of the Work

The remainder of this thesis is organized in the following chapters: Chapter two summarizes the foundations of this research. In this chapter, first requirements discovery as the superordinate process of requirements mining is defined and related to different requirements engineering and software development approaches. Then different types of requirements discovery systems and their technological characteristics are presented.

In the third chapter, an analysis framework for the related work of this thesis is conceptualized. The analysis framework is then applied to research works in the area of RMS which represent the related work of this thesis. This analysis results in the identification of research gaps to be addressed in this thesis.

In the fourth chapter, the overall methodology which is applied in this thesis is presented, including an introduction to the concepts of Design Science Research (DSR), the research paradigm which is followed here.

Chapter five then describes the first main result of this thesis, a design theory for RMS. The description is structured along the eight components of an IS design theory suggested by Gregor and Jones (2007), including a presentation of the designed artifact.

In chapter six, the results of two quantitative evaluations which have been conducted over the course of this thesis project are depicted. The first evaluation was performed during the design of the artifact while the second evaluation was conducted based on the artifact's final version.

In the subsequent chapter seven, results of both evaluations and the overall research project are discussed.

Finally, in chapter eight, the contents of this thesis are summarized, limitations and future research opportunities are outlined and both research and practice contributions are depicted.

2 Foundations

In the following sections, requirements discovery and related terms are defined and characterized. Subsequently, requirements discovery is related to existing software development and requirements determination approaches.

2.1 Defining Requirements Discovery

In general, a requirement is “a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents” (IEEE 1990, p. 62). Requirements can include "specifications of the service the system should provide, the constraints on the system and background information which is necessary to develop the system" (Rayson et al. 2000, p. 1363). Following the suggestion of Ambriola and Gervasi (1997) in the context of this work, the term “requirement” is used for the final product of requirements determination as well as for early incarnations of the same information.

The determination and management of requirements is generally associated with the Requirements Engineering (RE) discipline. Pohl (2010, p.48) characterizes RE as a “cooperative, iterative and incremental process” aiming at 1) gathering and understanding all requirements 2) agreeing on requirements between all stakeholders and 3) documenting requirements complying to defined specification formats and rules.

Requirements can be documented in natural language (e.g., a narrative scenario), in models (e.g., UML[3] models) or even figures (e.g., a drawn user interface mockup) (Pohl 2010). This thesis focuses on natural language requirements (NLR). NLR can be expressed in documents (e.g., informal requirements specifications, interview transcripts, workshop memos, or narrative scenarios) as well as in other resources (e.g., entries in issue tracking or test case management systems, support databases or discussion forums) (Vlas and Robinson 2012). Therefore, in the following the term “natural language requirements resources” (NLRR) is used instead of “natural language documents”.

[3] Unified Modeling Language.

As depicted in the introduction, NLR are usually transformed from initially informal statements into a more consistent and unambiguous representation, often containing additional information about a requirement's category or interrelation to other requirements. In RE research there are different terms describing this process as requirements elicitation (Castro-Herrera et al. 2009), requirements analysis (Ambriola and Gervasi 2006), requirements identification (Casamayor et al. 2010) or requirements classification (Cleland-Huang et al. 2007). In absence of an agreed-upon term and in analogy to the Knowledge Discovery process (Fayyad et al. 1996) which proceeds similarly, this process is referred to as "Requirements Discovery" in the context of this thesis. Within requirements discovery, two main process steps can be differentiated: requirements identification and requirements transformation (Cleland-Huang et al. 2007; Vlas and Robinson 2012). Both the identification as well as the transformation of requirements can be performed with and without system support. These two steps are looked upon in detail in the following.

Within a NLRR, a requirement may be represented by anything from single words (e.g., a data field to be implemented), over an entire sentence (e.g. the description of a function) to a sequence of sentences (e.g. to specify a non-functional requirement). *Requirements identification* mainly serves two purposes: First, it separates text that describes requirements from text which is not relevant from a requirements point of view. Second, it delimits each requirement within the document, resulting in multiple, individual requirements statements (Vlas and Robinson 2012). Depending on the text's degree of structure and preprocessing, the amount of irrelevant content can largely vary. In Open Source Software Development, for example, requirements are often identified from forums containing thousands of lines of social communications, code segments or slang which do not contain any requirements (Cleland-Huang et al. 2007). At the other end of the spectrum, requirements could be identified within already pre-processed, semi-structured use case descriptions which contain requirements in a very condensed form. By ignoring or even eliminating non-relevant passages of a requirements description, the requirements identification also results in a summarization of the source information. In addition to this document-wide summarization, requirements descriptions can also be *abstracted* to derive the main concepts and most significant

terms of the domain under investigation. From the requirement "The user interface should provide information about the flight number, gate and departure time" for example, the abstractions "flight number", "gate" and "departure time" could be extracted to build up domain-specific knowledge for traveling applications. Abstractions can be used to support subsequent identifications and transformations or to provide a value in itself. They can be used for example in early requirements elicitation steps to assist an analyst in gaining an understanding of an unfamiliar domain by providing a collection of the core terminology (Goldin and Berry 1997).

Based on the identification of individual requirements, a subsequent transformation can be conducted. *Requirements transformation* can include multiple, non-exclusive transformation steps which are introduced in the following. A widespread way to enrich requirements with additional semantics is the *classification* into distinct categories (Casamayor et al. 2010; Cleland-Huang et al. 2007; Vlas and Robinson 2012). By using requirements templates (e.g. the Volere requirements template[4]), requirements are classified into categories such as functional or non-functional requirements and sub-categories of these (e.g. performance requirements as a sub-category of non-functional requirements). An according classification can simplify (or even be a prerequisite for) subsequent modeling activities. Classified requirements can be grouped together to derive specific model types (e.g., a data model). In addition, a classification structure which is envisioned in a template can help to avoid omitting certain aspects of software (e.g., usability requirements).

After individual requirements have been identified, they can be interrelated to create models. A requirements specification for a purchasing application for example could describe individual data requirements for a user interface (e.g., "The user interface to enter purchase orders should include a data field to select a purchasing organization. In case a purchasing organization is subdivided, it should also be possible to select a purchasing group"). During requirements *interrelation*, these two individual requirements could be linked in a data model, in which the according relationship between purchasing organizations and purchasing groups is depicted. Requirements

[4] http://www.volere.co.uk/template.htm (5.2.2013).

interrelation is based on abstract terms, and therefore is usually performed after requirements abstraction has been conducted (Kof 2004; Mich and Garigliano 2002).

2.2 Relating Requirements Discovery to IS Development

In traditional IS development approaches, requirements discovery is associated with a formal process and distinct phases summarized as Requirements Engineering (Sommerville 2010). In the following, traditional RE is characterized with a focus on the relation to requirements discovery activities. Even though traditional RE is still a widely-followed approach, various alternative development approaches (e.g., market-driven development) have emerged in recent years, resulting in different settings and challenges for requirements discovery. Therefore, in addition to traditional RE, requirements discovery is also related to alternative development and requirements determination approaches.

2.2.1 Traditional Requirements Discovery

Traditional RE differentiates between two main processes, requirements determination and requirements management (Davis 1982; Pohl 2010). Requirements determination includes the elicitation, analysis, negotiation, specification and validation of requirements (Davis 1982; Pohl 2010). Requirements management includes change, traceability and release management for requirements (Pohl 2010; Sommerville 2010) (Figure 1).

There is no general agreement to which phase requirements discovery should be assigned. While some authors relate it to requirements elicitation (Castro-Herrera et al. 2009; John and Dörr 2003; Kaiya and Saeki 2006; Kiyavitskaya and Zannone 2008; Shibaoka et al. 2007), others assign it to requirements analysis (Cybulski and Reed 1998; Mich and Garigliano 2002; Park et al. 2000; Seresht et al. 2008). While one could argue that it contains aspects of both phases (associating the identification task with elicitation and the transformation task with analysis), this apparent inconsistency could also be caused by the inconsistency in definitions of the phases themselves. For example, Pohl (2010) regards analysis activities to be part of elicitation, without being a phase on its own. Sommerville (2010) similarly sees elicitation and analysis tightly

interwoven and combines them in one phase called “elicitation and analysis”. Hickey and Davis (2004) in contrast see them as two separate phases. Moreover, the term “requirements analysis” is often used as a synonym for “requirements engineering” in the RE literature (Cao and Ramesh 2008).

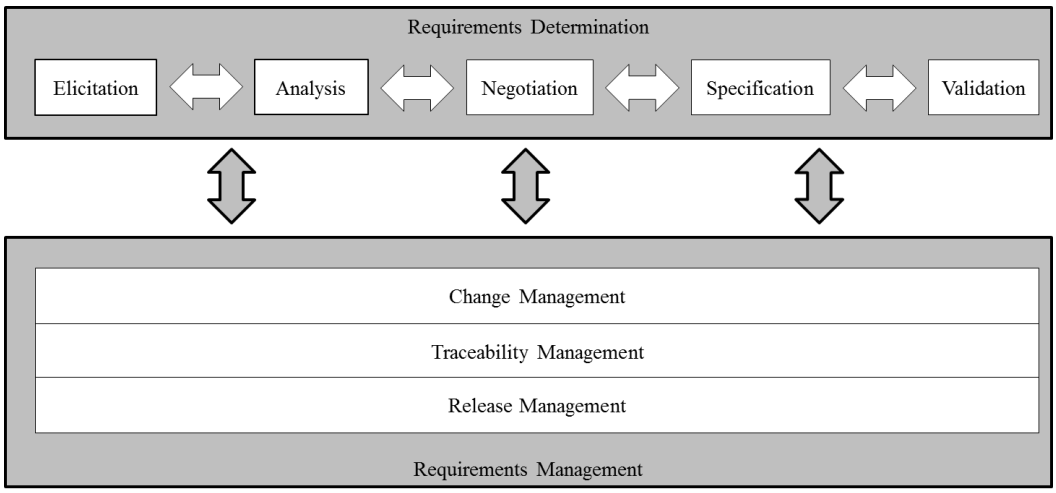

Figure 1: Requirements Engineering Processes

Despite this disagreement in allocation, the discovery of requirements depends on the provision of unstructured or semi-structured requirements descriptions which are usually gained through elicitation methods in the context of traditional RE (Pohl 2010). The majority of these methods involves a direct interaction between requirements owners and requirements producers (Goguen and Linde 1993). Requirements owners are usually stakeholders and users of the software who provide requirements. Requirements producers conduct a first documentation of requirements and are generally part of the product or development team. Ideally, requirements elicitation would ultimately result in a set of complete and correct requirements. However, due to cognitive, motivational and communicative issues in the exchange between requirements owners and producers, this is often not the case (Davis 1982; Valusek and Fryback 1985). For example, when a user is asked concerning his requirements for a new system, he is challenged to verbalize his implicit knowledge. This requires an immediate mental compilation and structuration of previously unordered information

resulting in significant cognitive work. Instead of delivering an optimal solution to this task, users tend to be satisfied with a "good enough" one (Valusek and Fryback 1985).

To respond to these issues, a plethora of methods such as interviewing, focus groups, observations, document analysis or repertory grids have been researched and practiced (Davis et al. 2006; Goguen and Linde 1993; Tuunanen 2003). Even though some authors propose the usage of one single method in any possible situation, an approach fitting every domain, application and requirements context is yet to be found. Instead, Hickey and Davis (2004) suggest an active selection process for elicitation methods, incorporating problem, solution, and project domain characteristics as well as the state of the requirements.

Many methods used during requirements elicitation result in unstructured or semi-structured NLRR. Interview outcomes for example are summarized in interview notes or even transcripts and results of focus groups are documented in meeting protocols or in a simple email. In a subsequent requirements discovery these documents are analyzed to identify single requirements and transform them into a more formal representation. Therefore requirements discovery can be seen as a connecting activity between the requirements elicitation phase and subsequent phases.

The traditional RE approach is characterized by distinct, sequential phases and an upfront and “en bloc” determination of requirements (Sillitti et al. 2005). Each of the phases is self-contained, and the process does not move to the next phase until the previous phase is completed. Furthermore, it is subject to a high degree of formality, enforcing standards at the hand-off between different phases and involving an abundance of documentation (Robey et al. 2001). Although this is still a widely-followed approach (particularly in custom software development), various alternative development approaches have emerged in recent years and became increasingly important (Ramesh et al. 2007; Sharp et al. 2007; Vlas and Robinson 2012; van de Weerd et al. 2006). Caused by different delivery models (such as packaged or open source software) or alternative development paradigms (such as agile or user-centered development), requirements discovery is often performed in a different setting than in the traditional development approaches. In the following, these differences and their consequences are pointed out.

2.2.2 Market-Driven Requirements Discovery

Software is increasingly developed by specialized companies (software vendors) implementing packaged software (Sawyer 2000). Packaged software (also known as commercial-off-the-shelf or commercial software) includes all types of software sold as tradable products (purchased from vendors, distributors or stores) for multiple types of hardware and operating systems (Carmel 1997). In contrast to custom-built software, packaged software is usually licensed, instead of sold (Sawyer 2000).

The development of packaged software (sometimes also called market-driven development) aims at implementing standardized software products for markets consisting of a potentially large number of different customers (Karlsson et al. 2002). In contrast to traditional RE, in this development approach a clear differentiation between requirements owners and producers is often not possible. Users often act as requirements producers: customer wishes (which later evolve into market-driven requirements) are directly articulated and described in natural language through customers using issue tracking systems, emails or similar electronic communication means (Regnell et al. 1998). Similarly, developers frequently act as requirements owners: technology-driven requirements are "invented" by developers or product managers of the software company to differentiate the own product from a competitive market (Karlsson et al. 2002; Regnell et al. 1998). The relative ease of requirements creation in combination with a development model which aims at a large number of customers can easily result in a big and continuous flow of incoming requirements, a situation which is referred to as "requirements overload" (Karlsson et al. 2002). In addition, due to requirements owners from different companies, requirements are not synchronized between different stakeholders resulting in a high probability of requirements duplicates, overlaps and contradictions (van de Weerd et al. 2006). Even for requirements without interdependencies, the initial description quality is often poor (Regnell et al. 1998). Prior to the first inspection through the software vendor, requirements do usually not pass any quality control, do not adhere to specification standards and are often formulated by authors not familiar with requirements specification (Regnell et al. 1998).

Consequently, product owners and other employees responsible for requirements discovery at software vendors are facing two major challenges. First, during requirements identification, the main issue is the sheer amount of different NLRR to be analyzed (Karlsson et al. 2002). Second, during requirements transformation, potentially inconsistent customer wishes need to be processed into consolidated product requirements (Natt och Dag et al. 2004). Consolidation is further impeded by the continuous arrival of new requirements and the changes applied by customers to already processed ones.

2.2.3 Agile Requirements Discovery

Traditional RE approaches face the problem that requirements are often changed, added or dismissed during the course of a development project, a circumstance which cannot be adequately handled in a linear, sequential development model (Rajlich 2006). As a consequence, the resulting software often does not match the users' needs after deployment on the one hand, while on the other, implemented features are sometimes not used (Petersen and Wohlin 2010). Addressing this issue, Agile Software Development became increasingly popular in the last decade. It propagates an iterative and incremental software development approach (Larman and Basili 2003) and the compliance to a set of principles expressed in the Agile Manifesto:

> *"Individuals and interactions over processes and tools*
> *Working software over comprehensive documentation*
> *Customer collaboration over contract negotiation*
> *Responding to change over following a plan"*
> The Agile Manifesto (Beck et al. 2001)

These principles are also applied to requirements determination and manifest in the following differences to traditional RE. First, instead of formal specifications, requirements are mainly specified via face to face communication and narrative user stories (De Lucia and Qusef 2010). The latter represent short, natural language feature descriptions of the system to be built (Cohn 2004). In contrast to use cases, user stories

describe a single requirement to be fulfilled instead of a complete scenario (Leffingwell 2011). User stories are written from the user's perspective, addressing the strong customer focus of the agile principles. A typical way to formulate a user story is the "role-activity-business value" form, in which a stakeholder describes in one sentence, in which role he interacts with the system during an activity to achieve a business value (Cohn 2004). While the choice of lean documentation can increase responsiveness to customers' needs and reduce time efforts for documentation, it becomes problematic when customers are not available or cannot come to consensus (in case of multiple customers) (Cao and Ramesh 2008). Furthermore, when people are leaving the development team (or even the company) their work and knowledge is hardly reproducible from documentation.

Second, instead of an initial upfront elicitation, requirements are determined iteratively (Ramesh et al. 2007). As customers often do not have a complete picture of the set of requirements at the beginning of a project, this approach offers the opportunity to explore requirements incrementally (Leffingwell 2011). While the elicitation quality of functional requirements can benefit from iterative elicitation, there is, however, a strong concern that it neglects certain non-functional requirements, such as scalability, maintainability, portability, safety, or performance (Cao and Ramesh 2008). In traditional RE, these technical requirements are often contributed by developers or architects, also viewing the system from a technical perspective, which can get lost when elicitation strictly focuses on the user perspective.

For requirements discovery, the focus on face to face communication reduces the amount of documented NLR, which are necessary for requirements discovery. Accordingly, the added value of requirements discovery in an agile setting can be questioned. However, as previously described, continuous, extensive and direct customer integration is an ideal which can often not be realized in practice. In cases customers cannot be physically present for face to face communication, requirements are still formulated and discussed using information and communication technology (e.g., through emails, ticket systems or similar means). To complement requirements information from face to face communication, these sources therefore additionally need to be considered and can be adequately analyzed by requirements discovery. An

according strategy to combine personally with electronically communicated requirements becomes even more important when the agile principle of iterative and incremental requirements elicitation is applied and requirements discovery is a continuous activity.

2.2.4 Distributed Requirements Discovery

Distributed development is a major trend in software engineering (Agerfalk et al. 2009; Pries-Heje and Pries-Heje 2011). It is usually conducted by virtual teams which are working together but without being co-located (Casey and Richardson 2006). Virtual teams can collaborate across geographical and organizational boundaries and are usually linked by communication and information technology (Lipnack and Stamp 1997). According to a study by Robinson and Kalakota (2004), over 95% of the Fortune 1,000 firms utilize globally distributed development teams. Multiple advantages are associated with an according approach, including decreased costs through wage differences between countries, a better access to highly qualified employees through global sourcing and reduced implementation times as a result of working "around the clock" in different time zones (Herbsleb and Moitra 2001; Holmström et al. 2006). However, it also creates new challenges, due to increased complexity. Working in a virtual team, the complexity of communication, coordination and collaboration can increase, e.g. due to different cultural backgrounds and differing work practices (Agerfalk et al. 2009; Li and Maedche 2012).

In distributed RE, methods which rely on face to face, synchronous communication are often replaced by electronically mediated, asynchronous communication (Menten et al. 2010). Electronically identified requirements enable the assurance of traceability and rationale management which are of utmost importance for overall distributed development and specifically for distributed RE (Geisser et al. 2007; Hildenbrand et al. 2009). In recent years, using internet technology, multiple types of information and communication support have been established to support distributed requirements elicitation. Using wikis (Geisser et al. 2007), forums, issue tracking systems (Scacchi 2002) or similar technologies, a lean early documentation of requirements, often in natural language, can be achieved. For requirements discovery, these NLRR provide

abundant material for the identification of requirements. In this setting, the consolidation of requirements is a major challenge, as requirements statements can be spread across different sources and media. To support this consolidation, systems which enable an identification and classification of individual requirements have been suggested (Vlas and Robinson 2012).

2.2.5 User-Centered Requirements Discovery

The idea of a "User-centered design" was first propagated by Donald Norman in the 1980s and became popular after the publication of two books (Norman and Draper 1986; Norman 1988) in which the author explains how the usability of products can be improved by putting the user (and not the system) into the center of all design activities. In this approach designers have the primary role of simplifying the user-system interaction and make sure that the actual system usage equals (or at least comes close to) the intended usage. This aspired congruence prerequisites an extensive understanding of the users and their tasks which shall be accomplished by a strong integration of users in all development phases. Additionally to user-centricity, Gould and Lewis (1985) recommend two further principles which have been incorporated in most user-centered procedure models, namely "empirical measurements" and an "iterative design". While the first principle recommends evaluating prototypes of the software in early development stages through actual users, the second suggests to continuously design, test and measure to be able to fix usability problems. To apply user-centered design in practice, different procedure models have been proposed (e.g., the "Star Lifecycle Model" (Hartson and Hix 1989), the "Usability Engineering Lifecycle" (Mayhew 1999), or "Goal Directed Design" (Cooper et al. 2007)). Furthermore a "Human-Centered Design Process" has been normed by ISO standardization (see Figure 2).

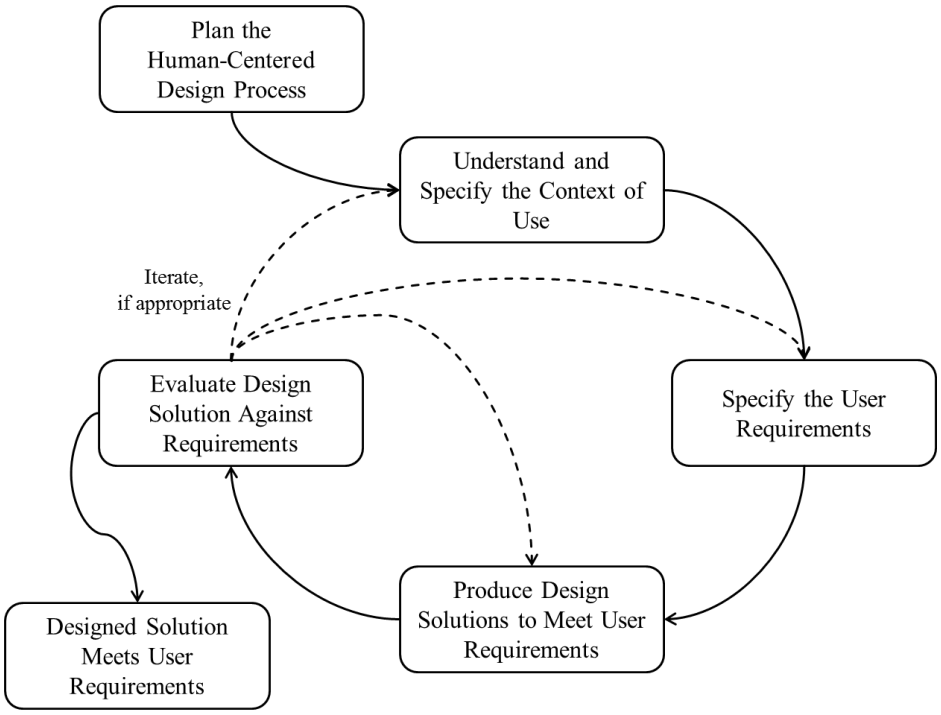

Figure 2: Human-Centered Design Process[5]

One of the distinguishing elements in comparison to other software engineering approaches is the initial activity "Understand and specify the Context of Use" before the specification of user requirements. Revisiting the goal of user-centered design to increase usability, this activity reflects the fact that usability is no generic attribute, but defines "[t]he extent to which a product can be used by *specified users* to achieve *specified goals* with effectiveness, efficiency and satisfaction in a *specified context of use*." (ISO 1998)

An established method to capture the specific context of use is the contextual analysis, proposed by (Beyer and Holtzblatt 1998). The basic principle of contextual analysis is the observation and inquiry of users at their actual workplace and during their daily work activities. Applying this method, requirements or usability engineers learn which aspects of the current work practices (including the utilized IS) are helpful or hindering. Furthermore, it can be clarified which features of an IS are important or less important

[5] According to ISO 9241-210 (ISO 2010).

for a user. To get a comprehensive picture, contextual analyses are usually conducted with multiple users (even in similar working contexts) (Wixon et al. 1990). Requirements Engineers should remain passive during contextual analyses, taking the role of an apprentice who learns the users work context from him (Beyer and Holtzblatt 1998). Learning how and why something is done or not is one of the main goals of this exercise.

During the specification of the context of use, a plethora of unstructured and semi-structured documents and materials is compiled which can be analyzed during requirements discovery. This includes interview transcripts, observation notes or first, narrative scenario descriptions describing a typical work practice (Sharp et al. 2007). Contextual analyses which involve observations may also result in audio or video material containing requirements information. The combined analysis of textual and non-textual information therefore represents an additional challenge in user-centered requirements discovery.

2.3 Summary

In this chapter, topic-specific terms and concepts which are relevant in the context of this thesis were introduced. Starting with general definitions of requirements and requirements engineering, the specific process of requirements discovery was defined and conceptualized. This specific process has then been related to existing approaches to develop software and determine requirements, highlighting the specific impact and context of requirements discovery.

3 Related Work [6]

In this chapter, an analysis framework for related research works on Requirements Discovery Systems (RDS) is presented. First, an overview of the analysis framework is depicted. Then each of the framework's dimensions and characteristics is presented in detail. In the last section of the chapter, the framework is applied to research in the area of Requirements Mining Systems (the focus of this thesis) and the research gap which will be referred to is outlined.

3.1 Analysis Framework

As previously described, unassisted requirements discovery can be time-consuming and error prone. Therefore a plethora of systems have been proposed to support the process (Meth et al. 2013a). These systems are referred to as RDS in the following and are analyzed along a multi-dimensional analysis framework, which is depicted in Figure 3. The framework consists of multiple dimensions (e.g., purpose), characteristics which are assigned to a dimension (e.g. "evaluation approach" is assigned to "evaluation") and values for characteristics (e.g. the characteristic "evaluation approach" can have the value "controlled experiment"). The first two dimensions (purpose and design) are used to analyze RDS from a technological point of view. First, analyzing the *purpose* of the systems, a differentiation concerning the output of the systems is made. Second, investigating the *design* of the systems, characteristics of the employed technology are distinguished. The third and fourth dimension (evaluation and knowledge exchange) complement the framework to enable a holistic assessment of RDS research works. This includes an analysis of the chosen *evaluation* approaches and constructs as well as a classification of the type of *knowledge exchange* applied in the research work. Each of the dimensions, their related characteristics and the different values of these characteristics will be explained in detail in the following.

[6] Parts of this chapter of the thesis are based on Meth et al. (2013a).

<table>
<tr><td colspan="3" rowspan="2">Purpose</td><td colspan="2">Identification of Abstractions</td><td colspan="2">Identification of Requirements</td></tr>
<tr><td colspan="2">Classification of Requirements</td><td colspan="2">Interrelation of Requirements</td></tr>
<tr><td rowspan="5">Design</td><td rowspan="2">Processing Characteristics</td><td>Degree of Automation</td><td>Manual</td><td>Semi-Automatic</td><td colspan="2">Automatic</td></tr>
<tr><td>Automation Technology</td><td>Information Retrieval</td><td>Natural Language Processing</td><td colspan="2">Other</td></tr>
<tr><td rowspan="3">Knowledge Base Characteristics</td><td>Knowledge Origin & Volatility</td><td colspan="2">Imported / Static</td><td colspan="2">Retrieved / Dynamic</td></tr>
<tr><td>Structure</td><td colspan="2">Dictionary</td><td colspan="2">Ontology</td></tr>
<tr><td>Domain Specificity</td><td colspan="2">Domain-Specific</td><td colspan="2">Domain-Independent</td></tr>
<tr><td colspan="2" rowspan="2">Evaluation</td><td>Evaluation Approach</td><td>Proof of Concept</td><td>Case Study</td><td>Simulation</td><td>Controlled Experiment</td></tr>
<tr><td>Evaluation Constructs (Dependent Variables)</td><td>Completeness</td><td>Correctness</td><td>Efficiency</td><td>Other</td></tr>
<tr><td colspan="2" rowspan="2">Knowledge Exchange</td><td>Knowledge Grounding</td><td>General Knowledge</td><td>Design Theory</td><td>Mid-Range Theory</td><td>Formal Theory</td></tr>
<tr><td>Knowledge Contribution</td><td>Artifact Description</td><td>Nascent Design Theory</td><td colspan="2">Well-developed Design Theory</td></tr>
</table>

Figure 3: An Analysis Framework for RDS Research Works

3.2 Purpose

The purpose of RDS is the support of the requirements discovery process in the identification and transformation of requirements from NLRR (e.g., documents, issue tracking databases or emails). In 2.1, different types of identification, namely requirements identification and abstraction identification and different types of transformation, namely requirements classification and requirements interrelation have been introduced. In the following, these characteristics of the discovery process are used to characterize different classes of RDS.

3.2.1 Abstraction Identification Systems

Purpose	Identification of Abstractions	Identification of Requirements
	Classification of Requirements	Interrelation of Requirements

Figure 4: Characterization of Abstraction Identification Systems

Abstraction Identification Systems aim at the identification of abstractions from NLRR which will, for example, assist a requirements engineer in gaining an understanding of an unfamiliar domain (Berry et al. 2012). In this context, abstractions are single words within the requirements document which represent the main concepts and most significant terms of the problem and application domain (Gacitua et al. 2011). This domain knowledge can then be used as a reference and a starting point during further requirements discovery. In particular the knowledge can help to avoid information overload and to overlook important aspects that might evolve into requirements (Berry et al. 2012). Systems that support abstraction identification through automatisms have been proposed by Gacitua et al. (2011) Goldin and Berry (1997) and Sawyer et al. (2002).

3.2.2 Requirements Identification Systems

Purpose	Identification of Abstractions	Identification of Requirements
	Classification of Requirements	Interrelation of Requirements

Figure 5: Characterization of Requirements Identification Systems

Requirements Identification Systems focus on the pure identification of requirements, without subsequent discovery steps. However, most of the systems support additional activities related to requirements determination. For example, in the system presented by Kaiya and Saeki (2006), NLRR are preprocessed to identify requirements and the related concepts. A requirements engineer then manually maps these concepts to items of an ontology from the same domain (if possible). Based on these mappings, the

system then recommends further requirements to be added. Through this procedure, the overall completeness and correctness of requirements descriptions shall be improved. An enhanced version of this system is presented in Shibaoka et al. (2007). Another example is the system developed by Castro-Herrera et al. (2009). It supports the identification of requirements themes. On the basis of initial statements, which are entered manually by the customers into a web-based tool, a linguistic processing is conducted to tag each statement with illustrative terms. Based on these tags, the statements are clustered to requirements themes. For each requirements theme, a discussion forum is created to foster further discussions among stakeholders.

3.2.3 Requirements Modeling Systems

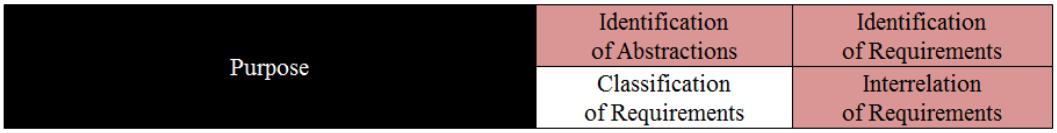

Figure 6: Characterization of Requirements Modeling Systems

Requirements Modeling Systems identify, abstract and interrelate requirements. The resulting models and their graphical representation can foster the discussion of requirements with stakeholders and enable a direct transition between requirements and design activities (Sommerville 2010). A plethora of systems has been proposed to support requirements modeling: While some systems generate standardized UML models (Ambriola and Gervasi 2006; Harmain and Gaizauskas 2003; Sampaio et al. 2007), others produce proprietary object-oriented models (Mich and Garigliano 2002), models specifically tailored to security requirements (Kiyavitskaya and Zannone 2008) or models to describe the interaction of the user with the system's user interface (Brasser and Vander Linden 2002; Lemaigre et al. 2008; Tam et al. 1998).

3.2.4 Requirements Mining Systems

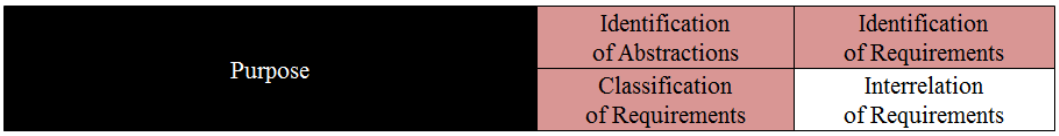

Figure 7: Characterization of Requirements Mining Systems

Requirements Mining Systems identify requirements and classify them according to an existing taxonomy. Depending on the type of knowledge generation (see 3.4.1), they can also include functionality for abstraction identification. Cleland-Huang et al. (2007) focus on non-functional requirements (NFR) as e.g. security, performance or usability requirements. Based on the notion that each sub-group of NFR has its unique keywords, the system uses different knowledge base items to find and classify NFR from each sub group. Casamayor et al. (2010) similarly aim at the detection of NFR, and employ a semi-supervised categorization approach that only needs a small set of manually classified requirements for the initial training of the classifier. In their system, the classification model is iteratively enhanced based on the users' feedback on the artifact's output. Rago et al. (2011) present QAMiner, a system that also aims at discovering NFR. The system, however, analyzes use case specifications, and relates requirements to pre-defined quality attributes (e.g., modifiability, performance, availability, etc.) to avoid that these non-functional aspects are understated in the resulting requirements specifications. Vlas and Robinson (2012) present an automated approach for the identification and classification of both functional and non-functional requirements in natural language feature requests of open source software projects.[7]

3.3 Design – Processing Characteristics

To fulfill the previously described purposes of different types of RDS, the systems provide alternative processing characteristics which will be presented in the following. The characterization is centered on the concept of automation, being the core processing concept of RDS (Cleland-Huang et al. 2007; Natt och Dag et al. 2002; Pérez-González

[7] Each of the four systems will be analyzed in more detail in the related work paragraph.

and Kalita 2002; Sampaio et al. 2007). First, a differentiation of approaches along different degrees of automation is made. After that the underlying technology to enable automation is introduced.

Design	*Processing Characteristics*	Degree of Automation	Manual	Semi-Automatic	Automatic
		Automation Technology	Information Retrieval	Natural Language Processing	Other

Figure 8: Processing Characteristics of RDS

3.3.1 Degree of Automation

While there are some research works, which present system support for purely manual requirements discovery (Abrams et al. 2006; Ossher et al. 2009), most RDS incorporate capabilities to at least partially automate the process. However, existing works show differences concerning the *degree of automation* provided. Research suggests that while system support can cause an efficiency advantage in comparison to a purely manual discovery (Cheng and Atlee 2007), a complete automation of requirements discovery tasks can lead to a loss of information or erroneous results (Goldin and Berry 1997). Berry et al. (2012) point out that the cognitive aspects of requirements discovery should not be underestimated, as RDS may omit important requirements, and fail to detect logically correct, but questionable requirements. Thus, automation approaches should additionally involve human interaction. This indicates a conflict between the benefits of automation and the necessity of human intervention. According to Parasuraman et al. (2001), the appropriate degree of automation in the support of human tasks should be chosen according to a variety of evaluative criteria, including the reliability of the automation and the costs of decision outcomes. While a full automation would replace the human analyst, a semi-automated approach would merely support him and thus rationalize requirements discovery, while still requiring an interaction with the system. In contrast to (semi-)automatic approaches, during manual requirements discovery an analyst would start the analysis from scratch, without any potential requirements recommended by the system. This said, it should be noted that in practice the degree of automation should rather be seen as a continuum than as a categorical concept. While a

fully automated approach might target to replace any manual requirements mining activity, in most cases an analyst will still double-check at least parts of the results of the automatism to make sure that requirements have been captured correctly. In this sense the differentiation between semi-automation and full automation which will be made in the framework should rather be understood as the design and usage focus of a system. Examples for semi-automatic RDS include the systems presented by Ambriola and Gervasi (2006), Casamayor et al. (2010), Rago et al. (2011) and Sawyer et al. (2002), examples for entirely automatic approaches are presented by Gacitua et al. (2011), Goldin and Berry (1997), Kiyavitskaya and Zannone (2008) and Vlas and Robinson (2012).

3.3.2 Automation Technology

Most RDS use Natural Language Processing (NLP) or Information Retrieval (IR) techniques to automate requirements discovery (Berry et al. 2012; Cheng and Atlee 2007). The according techniques can be employed to achieve each of the previously described requirements discovery purposes, which will be outlined in the following. There is plethora of different techniques from NLP, IR and other research fields which have been applied to RDS. The subsequent assembly therefore does not claim completeness, but should rather be seen as a compilation of prominent design choices for RDS systems.

3.3.2.1 Linguistic Preprocessing to Prepare Requirements Discovery

Before search techniques or other automated discovery techniques can be applied, the provided NLRRs need to be preprocessed. In this preprocessing, the texts are broken down to a list of relevant, individual and harmonized words (or even parts of words). This process is described in more detail in the following.

First, the text is split into single sentences and words, applying sentence segmentation and tokenization (Palmer 2000). *Sentence segmentation* aims at identifying sentence boundaries, which are usually indicated by punctuation marks. During *tokenization*, word boundaries are localized and used to further segment the text into single words. Even though in English texts in most cases word segmentation can be performed after

each space, there are some exceptions to this heuristic. For example, a genitive "s" (e.g. John's desk) is part of the previous word while an apostrophe "s" in verb contractions (e.g. she's) represents an additional word (is) which needs to be separated (Palmer 2000).

After tokenization has been performed, irrelevant words need to be eliminated to improve the performance and precision of subsequent processing, a process step referred to as *stop word removal* (Silva and Ribeiro 2003). Stop words represent words which are extremely common and therefore not helpful for NLP or IR processing (Manning et al. 2008). Examples for English stop words are "a", "of" or "the".

Finally, the remaining words usually need to be harmonized. *Harmonization* can help to detect duplicates and improve the results of subsequent processing steps. During searches, for example, using the exact same words as they originally occurred in a NLRR generates multiple problems. Semantically similar words might appear in varying forms, e.g. due to grammatical conjugation and declination, different spelling (e.g., American vs. British spelling) or inconsistent capitalization of words (Manning et al. 2008). Without harmonization these words would not be recognized as similar, resulting in an unsuccessful search. Thus different harmonization techniques can be employed which will be summarized in the following. First, during *normalization*, the capitalization of words is harmonized and accents, diacritics and hyphens are eliminated (Manning et al. 2008). Second, during *stemming,* words are reduced to their stems (Salton and McGill 1986). Word stems in contrast to original words do not contain grammatical alterations like plurals, gerund forms or tense suffixes.

Even though normalization and stemming can increase information retrieval success, they can come to limits if words have multiple meanings depending on their actual word class. For example, the word "order" can be used as a verb ("The system should provide functionality to order catering services") or as a noun ("The system should display details of an order"). Whereas in the first example "order" is part of an activity which should be supported by the system, in the second example "order" describes an object or data element. Similarly, it is difficult to apply stemming to irregular verbs, for example the word "went" has no common stem with "go" although they just represent different conjugations of the same verb. Therefore, alternatively to normalization and stemming,

the NLP technique of *lemmatization* can be employed. While normalization and stemming aim at the reduction of words to a common part (e.g. “production” is reduced to “produc”), lemmatization replaces the original word with a lemma. A lemma is a word, which serves as a proxy for an entire set of forms taken by this word. For example, the conjugations “choose, chose and chosen” would all be replaced by the lemma “choose”.

Lemmatizers usually require an input tuple of a) the word to be replaced and b) the word class associated with this word (e.g. noun, verb, adjective). In computer linguistics, these word classes are referred to as part-of-speech (POS) (Voutilainen 2003). *POS tagging* is the process of assigning part-of-speech labels to words (Jurafsky and Martin 2009). Additionally to the use in lemmatizers, POS tags can also be used to improve IR results (which will be described later on). Figure 9 gives an overview of the described NLP and IR techniques for linguistic preprocessing.

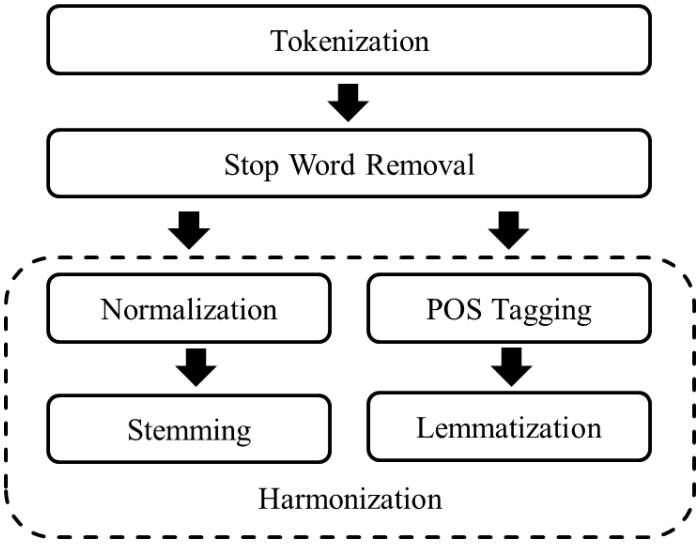

Figure 9: Linguistic Preprocessing Using NLP and IR Techniques

3.3.2.2 Frequency Profiling for Abstraction Identification

As described earlier, abstraction identification aims at identifying the main concepts and most significant terms of a requirements domain. The previously described techniques for linguistic preprocessing can help to identify and harmonize individual words within

a NLRR. However, apart from stop word removal, no filtering or selection is applied to reduce the set of words to the most important ones for a specific domain.

A common approach to achieve this is the usage of *frequency profiling* (Gacitua et al. 2011; Goldin and Berry 1997; Sawyer et al. 2002). In its basic form, frequency profiling is based on the idea that the importance of a word in a text is proportional to its frequency of occurrence (Goldin and Berry 1997). Consequently, the most frequently used words in a requirements document (apart from stop words) are identified as candidate abstractions, of which a requirements analyst could manually pick the final set of abstractions.

Although the usage of absolute frequency numbers already provides good results (Wermter and Hahn 2006), it can be improved by analyzing the relative frequency of words in the given text. Sawyer et al. (2002) describe *corpus-based frequency profiling* which is based on the assumption that words which are significant to a domain will be revealed by an increased relative frequency of appearance in the text in comparison to a normative corpus. As a normative corpus, they apply a 2.3 million-word subset of the British National Corpus which contains transcripts of spoken English. Whenever a word is strongly overrepresented in the given text (in comparison to the normative corpus) it qualifies to be identified as an abstraction. While corpus-based frequency profiling works well for single words, it cannot be applied to multiword terms (e.g. "requirements engineer"). Therefore, Gacitua et al. (2011) suggest to calculate significance values for multiword terms by using weighted averages of the individual words log-likelihood[8]. Their results show that an according approach can successfully capture multiword terms and thus help to further automate abstraction identification.

3.3.2.3 Techniques for the Interrelation of Requirements

A large variety of methods has been used in alternative combinations to support the interrelation of requirement resulting in requirements models (Ambriola and Gervasi 2006; Kof 2004; Mich 1996; Omoronyia et al. 2010). Instead of describing each technique in isolation, an exemplary approach to combine different methods as suggested by Kof (2004) is presented in the following. The interrelation of requirements

[8] Log-likelihood is a measure for the relative frequency of a word.

in a NLRR basically breaks down to an interrelation of single words within this resource. A first hint for an association between words in a document can be drawn from the structure of individual sentences. Kof (2004) suggests building parse trees from each sentence. In these parse trees, a sentence predicate and its subject and object are captured and linked to each other. The resulting set of trees is then clustered to derive further associations. First, parse trees of the same predicate are grouped into one cluster. Then, the resulting clusters are compared, searching for overlaps in their subjects or objects. Overlapping clusters are joined and result in initial taxonomies. In a last step, association mining (as suggested by Maedche and Staab (2000)) is applied. Words which often occur in the same sentences are assumed to be associated. Consequently, the taxonomies holding these words are linked to each other, resulting in an interrelated requirements model (or more specific an ontology).

3.3.2.4 IR Techniques for the Identification and Classification of Requirements

Web search engines (such as Google) are probably the most well-known applications of IR techniques. In response to a set of entered search terms, a web search engine generates a list of matching websites. Prior to the search, each of the websites has been indexed, resulting in a list of words associated with the site. During the search, instead of scanning entire websites, the search terms are applied to the lists of indexed terms resulting in a faster response time.

The same principle can be applied to requirements identification. Requirements identification in a NLRR is basically about differentiating those words which represent requirements from further content which is non-relevant from a requirements point of view. To support this task, knowledge bases which contain requirements terms are provided. These terms are assigned to requirements categories (e.g. the term “credit card number” might be assigned to the category “data requirement”). Further details about knowledge bases will be presented in Section 3.4. Figure 10 shows how IR can be applied in this scenario to support requirements identification. Each term in a NLRR can be used as a search term. Using this search term, the IR algorithm strives to identify a matching requirements category by searching the requirements terms within the

knowledge base. A term will only be successfully identified as a requirement if this search is successful, meaning that a requirements category is associated with the search term with ample probability[9]. For classification, the requirements category with the highest probability is then assigned to the identified term. If no requirements category with sufficient probability is identified, the term remains unassigned.

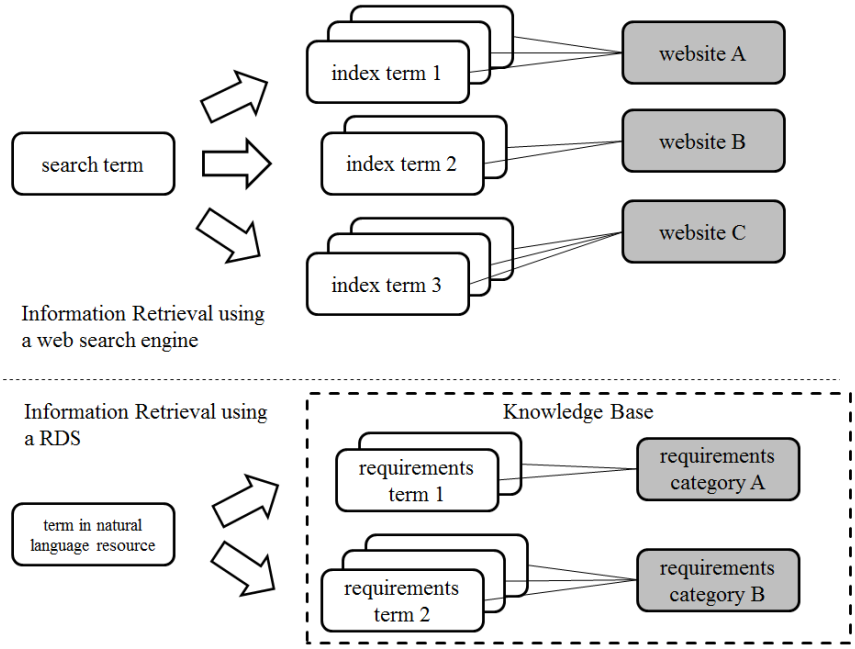

Figure 10: Comparison of IR Usage in Web Search Engines and RDS

3.4 Design – Knowledge Base Characteristics

As described earlier, many automation techniques used for requirements discovery require the existence of a knowledge base. Knowledge bases consist of knowledge items which are made up of terms and meta-information associated to these terms. Terms can be used during requirements identification to act as an index during the retrieval process. They are usually linked to further information, for example an assignment to a requirements category (Lemaigre et al. 2008; Sampaio et al. 2007). Knowledge bases can differ in the origin, volatility, structure and domain-specificity of the included knowledge which will be explained below.

[9] For this purpose usually a threshold probability is defined.

Design	*Knowledge Base Characteristics*	Knowledge Origin & Volatility	Imported / Static	Retrieved / Dynamic
		Structure	Dictionary	Ontology
		Domain Specificity	Domain-Specific	Domain-Independent

Figure 11: Knowledge Base Characteristics of RDS

3.4.1 Origin and Volatility of Knowledge

Knowledge origin describes the way the knowledge bases required for knowledge re-use are populated. The creation of knowledge is either initiated by an upload of existing knowledge to the system (referred to as "imported knowledge") or by knowledge retrieval from documents (referred to as "retrieved knowledge") (Staab et al. 2001). In contrast to imported knowledge, retrieved knowledge can usually be acquired in combination with actual usage data.

In the context of requirements discovery, this could be information about how often users have assigned a specific term to a specific requirements category. Retrieved knowledge can be added to the knowledge base as a byproduct of manual requirements discovery. For example, the data requirement "frequent flyer number" might have been overseen by automated requirements discovery and might then be identified and classified manually. This manual activity has two effects. First, it adds an additional requirement to the automatically discovered requirements from this resource. Second, it adds a potential new knowledge item to the knowledge base, consisting of the term "frequent flyer number" and the assignment to the category "data requirement". Through this mechanism a constant flow of potentially new knowledge items is created. Consequently, it has been integrated into a number of existing RDS. Cleland-Huang et al. (2007), e.g. iteratively train their non-functional requirements classifier based on the analyst's feedback. Kaiya and Saeki (2006) similarly consider a refinement of imported knowledge drawing on the information extracted from the requirements statements, thus incorporating retrieved knowledge. In contrast to the dynamic nature of retrieved knowledge provision, imported knowledge is only added if the responsible knowledge

engineer initiates a knowledge upload. Consequently, the creation of imported knowledge is rather static in comparison to the generation of retrieved knowledge.

3.4.2 Structure and Domain-Specificity of Knowledge

Knowledge bases can vary in *structure* and complexity. They often consist of either dictionaries (Lemaigre et al. 2008; Sampaio et al. 2007) which hold assignments of terms to requirements categories or ontologies (Kaiya and Saeki 2006; Vlas and Robinson 2012) which additionally include relations between different concepts. While dictionaries can help in the identification and classification of individual requirements, ontologies can be used to improve the overall discovery results. Kaiya and Saeki (2006), for example, use ontologies to improve the completeness and consistency of the discovered requirements. They achieve this by comparing the identified requirements with an existing domain ontology. For example, an analyst could manually map a requirement which specifies a train reservation capability to the "reserve" knowledge item in a domain ontology for reservation systems. In this ontology, the "reserve" item is related to the item "cancel" (it should be possible to cancel a reservation). Based on this information, the system would inform the requirements engineer to additionally consider a "cancel functionality" (if not already included in the NLRR).

The discovery of requirements premises, to some extent, the existence and application of domain knowledge (Ambriola and Gervasi 2006; Hickey and Davis 2004). Data requirements such as "purchase order number" or "material group" might be of high relevance for the domain of procurement applications, while they would be irrelevant for a human resource application. Consequently, the automated discovery of this type of requirements can profit from a *domain-specific* knowledge base which already contains corresponding knowledge items. In contrast, other types of requirements, for example performance requirements, can be identified with less domain knowledge. The requirement "The response time for this function should be faster than 10 seconds", for example, could be defined for an application in almost any kind of domain. In this case, related terms such as "response", "time" and "second" would be typical examples for domain-unspecific knowledge items. Due to these differences in domain specificity across different requirements categories, there might be domain-specific and domain-

unspecific contents within a single knowledge base (Lemaigre et al. 2008). Therefore, instead of an alternative classification in domain-specific and –unspecific knowledge bases, the proposed analysis framework allows both classifications at the same time. An example of a rather domain-unspecific knowledge base is described by Brasser and Vander Linden (2002) who present a system to capture interaction requirements, while an example for a domain-specific knowledge base is provided by Kaiya and Saeki (2006) (as depicted in the last paragraph).

3.5 Evaluation

RDS related research aims at knowledge contribution through the development and investigation of artifacts. It can therefore be associated to design research (Hevner et al. 2004; Simon 1969). Works which follow a design research approach are usually characterized by two main research phases. In the build phase an artifact is designed. Then, in the evaluation phase, the effectiveness of the artifact is assessed. To enable a holistic assessment of RDS research work, the previously introduced framework therefore includes a dimension to describe the *evaluation* phase of these works. The according framework characteristics and their values are presented in the following.

Evaluation	Evaluation Approach	Proof of Concept	Case Study	Simulation	Controlled Experiment
	Evaluation Constructs (Dependent Variables)	Completeness	Correctness	Efficiency	Other

Figure 12: Evaluation Characteristics of RDS Research Works

3.5.1 Evaluation Approach

Hevner et al. (2004) distinguish two experimental design evaluation methods: A *controlled experiment* involves studying the presented system in a controlled environment which can be done e.g. by comparing the performance of an analyst using the system with the performance of an analyst devoid its support. In contrast, a *simulation* comprises the execution of the artifact with test data (Hevner et al. 2004). In the context of RDS, a performance evaluation based on a simulation is possible by

comparing a system's output to a gold standard set of requirements, which is the output created manually by an expert or a group of experts.

Additionally to experiment evaluations, two further types of evaluations are frequently applied in the context of RDS (Meth et al. 2013a): A mere demonstration of the presented system, e.g. by an application to a real-world example without data collection and analysis is classified as a *proof of concept* in the following, while an evaluation in practice, e.g. in an industrial environment, will be denoted as a *case study.* Accordingly, the identified works will be categorized to evaluate their approaches either by 1) a controlled experiment 2) a simulation 3) a proof of concept or 4) a case study.

3.5.2 Evaluation Constructs and Measures

To evaluate the effectiveness of RDS, the assessment of the completeness and correctness of the identified requirements is a common practice (Casamayor et al. 2010; Cleland-Huang et al. 2007; Rago et al. 2011). *Completeness* ensures that all the information required for a problem definition, i.e. all properties that are desired to hold true, are found within the specification (Zowghi and Gervasi 2003). The *correctness* of a requirements specification is determined by the included share of requirements which match existing needs. The IEEE Recommended Practices for Software Requirements classify a requirements specification as correct "if, and only if, every requirement stated therein is one that the software shall meet" (IEEE 1998, p.4).

An operationalization of these constructs is possible by drawing on metrics from the information retrieval domain, specifically precision and recall (Salton and McGill 1986). *Recall* is defined as the proportion of relevant items that are actually retrieved in answer to a search query and is very commonly used as a measure for completeness (Cleland-Huang et al. 2007; Kiyavitskaya and Zannone 2008; Sampaio et al. 2007). *Precision* is the proportion of retrieved items that are relevant to the query and is often used as a measure for correctness, usually in combination with recall.

RDS strive to generate requirements descriptions with high recall and precision. However, improving recall and precision at the same time is a challenge, as maximizing the number of retrieved requirements to improve recall is often done at the cost of also retrieving more irrelevant items which reduces precision. Trading off precision for

recall or vice versa, one might argue that for RDS, recall is the more important measure of both, as errors of commission are easier to correct than errors of omission (Berry et al. 2012). While an omitted requirement needs to be identified within a potentially longer source document, requiring significant time for manual searching, a wrongly identified document can easily be deleted from the list of the all identified requirements. This requires, however, that the resulting list of requirements is significantly shorter than the source document. Accordingly, recall and precision are sometimes complemented with a third measure describing the summarization provided by the system. Summarization measures the volume of a system's output in relation to the input document size. Systems providing a high level of summarization simplify manual corrections of automatically identified requirements as the analyst can concentrate on reviewing the relatively short output of the system in contrast to its longer input document. Particularly for abstraction identification systems, summarization plays an important role, as this type of systems aims at distilling the key abstractions of an initially long document. In the analysis framework the concept summarization is subsumed under the category "*Other (Constructs)"* together with further concepts which are only seldom applied.

In addition to measures for requirements quality, which represent the outcome of the discovery process, it is also worthwhile to observe the process leading to this outcome. In various works, process *efficiency* is assessed additionally to quality aspects (Cleland-Huang et al. 2007; Kiyavitskaya and Zannone 2008; Sampaio et al. 2007). Discovery efficiency can be measured by the time required to transform an unstructured input document to a set of structured requirements. In the case of RDS, this time period can be split into two phases: the automation phase and the manual phase. While the duration of the automation phase is determined by the runtime of the automation algorithm, the duration of the manual phase represents the time for manual corrections of the algorithm's findings. It can be argued that the duration of the automation phase is less critical than the duration for manual adaptions, as the automation can run in a background job without absorbing the analyst's time. In contrast, the time for manual adaptions should be observed critically, especially in evaluations which compare automated with manual approaches. In summary, to enable a holistic evaluation of a

system's effectiveness, the analysis framework considers both aspects (requirements mining quality and efficiency).

3.6 Knowledge Exchange

Through the description of an artifact's design and evaluation, design research contributes to the body of knowledge. However, an increase in knowledge contribution can be achieved if design research is based on existing theories or even contributes theory itself (Gregor and Hevner 2013). Thorough theory grounding can extensively leverage existing knowledge and thereby increase the likelihood of designs that are actually effective. Codification and abstraction of results in a design theory can help to generalize the findings of design research. An according conceptualization extends the contribution of design research beyond the search of specific solutions to specific problems and has been intensively discussed in DSR (Baskerville and Pries-Heje 2010; Gregor and Jones 2007). Both the knowledge grounding and contribution are summarized in a fourth dimension of the analysis framework, entitled *"Knowledge Exchange"*.

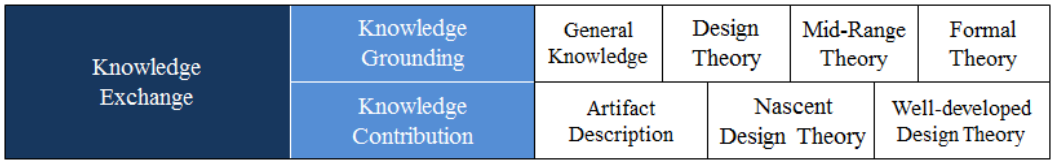

Figure 13: Knowledge Exchange Characteristics of RDS Research Works

3.6.1 Knowledge Grounding

In accordance with Gaß et al. (2012) four categories of knowledge to ground design research are differentiated: 1) formal theories 2) mid-range theories 3) design theories and 4) general knowledge. *Formal theories* (sometimes also referred to as "Kernel Theories") represent theories from within and outside the IS field, but mainly from natural and social science (Walls et al. 1992). They are mainly descriptive theories which can be used to guide the design and derive testable propositions for the evaluation of the artifact (Kuechler and Vaishnavi 2008; Walls et al. 1992). While the

grounding on kernel theories is generally regarded as a rigorous basis of DSR, it is often difficult to apply them to the specific, practical context of an artifact (Baskerville and Pries-Heje 2010). Therefore, Kuechler and Vaishnavi (2008) suggest *mid-range theories* which are based on formal theories but provide additional explanatory knowledge to increase applicability to practical problems. While formal and mid-range theories do not originate from actual design activities, the knowledge grounding can also be based on previous *design theories*. Gregor and Hevner (2013) refer to this reuse of prescriptive design knowledge as "exaptation", the extension of known solutions to new problems. Exaptation is appropriate in scenarios, where an artifact in one field is not available or suboptimal and is designed by applying prescriptive knowledge from artifacts of a different field. Finally, empirical and non-empirical *general knowledge* can be used to ground design research. Kuechler and Vaishnavi (2012) refer to this type of knowledge as "tacit theory", consisting of "insights or evidence/experience-based justifications for pursuing a novel design" (Kuechler and Vaishnavi 2012, p. 404). This informal type of knowledge enables DSR to explore domains in which more formal knowledge does not exist or is sparse (Kuechler and Vaishnavi 2012).

3.6.2 Knowledge Contribution

Kuechler and Vaishnavi (2012) classify DSR works concerning their knowledge contribution into three different groups. The first group consists of works which only present the implemented *artifact,* without further discussing how and why it works and which design practices have been employed in its implementation. Design knowledge and justification of design features in these works remain tacit and the entire knowledge is captured within the artifact. The authors state that this type of knowledge contribution is appropriate for groundbreaking innovations in which the artifact itself provides sufficient novelty to compensate scarce theoretical contributions.

The second group of works contributes additional knowledge in the form of an *Information System Design Theory* (ISDT). An ISDT as suggested by Walls et al. (1992) abstracts the design efforts to meta-requirements and design principles (meta-design) which prescriptively support the design of future instantiations within the same class of systems. Moreover, an ISDT explicitly codifies the knowledge which is

captured in an artifact which allows other researchers as well as practitioners to leverage the generated knowledge without the need to analyze the artifact itself.

As a third type of knowledge contribution (and a potential third group), Kuechler and Vaishnavi (2012) suggest the construction of a mid-range theory which they refer to as *design relevant explanatory/predictive theory* (DREPT). A DREPT should capture knowledge which cannot be adequately presented in an ISDT, namely the linking effects between kernel theory constructs and ISDT constructs. An ISDT is mainly occupied with the explanation of the build process. In contrast, a DREPT focuses on the explanation of the how and why of the observed effects.

Similarly, Gregor and Hevner (2013) differentiate three levels of knowledge contribution for DSR. Level one represents the specific implementation of an *artifact* in a specific context. Knowledge can be contributed, for example by a specific software product or process. Level two comprises more general and abstract descriptions of the design, referred to as *nascent design theory*. On this level, knowledge is contributed in the form of general operational principles or a general architecture rather than of specific characteristics and features. Components of nascent design theory might by constructs, design principles, models, methods or technological rules. Level three represents a knowledge contribution about the embedded phenomena, referred to as *well-developed design theory*. DSR projects resulting in mid-range or grand theories would be examples for this type of contribution. The different levels supposed by Gregor and Hevner (2013) are associated with increasing degrees of abstraction and knowledge maturity (rising from level one to level three).

The typology suggested by Gregor and Hevner (2013) is similarly utilized in the analysis framework for RDS works. However, on the first contribution level additionally to the artifact itself an informal description of the artifact in the corresponding paper is expected (which is usually part of the publication).

3.7 Results of Analysis

In this thesis, the design and evaluation of a Requirements Mining System (RMS) is described. Therefore, in the following description of related work, this type of RDS is focused on. The analysis comprises a detailed description of the four RMS which were

briefly introduced in section 3.2.4 and a depiction of the research gap which will be addressed.

3.7.1 Application of Analysis Framework to RMS Research Works

The system presented by Cleland-Huang et al. (2007) referred to as "NFR-classifier" supports the identification and classification of non-functional requirements. Furthermore, through the identification of abstractions it enables the creation of retrieved knowledge. Requirements statements are processed semi-automatically. Requirements can be categorized manually as well as through automation algorithms which employ IR and NLP techniques. Based on a first provision of imported knowledge, the knowledge base is iteratively extended through requirements engineers' feedback to the automation results. The knowledge base is structured as a simple dictionary consisting of a list of terms assigned to different sub-categories of NFR. Although the initially imported knowledge is domain-independent, the knowledge base can be customized to a domain through retrieved knowledge. The system is evaluated in a series of simulations, comparing the artifacts automatic results with a predefined gold standard. The evaluation uses recall and precision as measures for the completeness and correctness of the results and one additional measure (specificity). While the authors mention the time necessary to manually classify their sample set of requirements, they do not include an analysis of the time using their approach. The design is only grounded on general knowledge and contributions are restricted to a description of the artifact, without further abstraction or codification of the design. Figure 14 depicts the overall analysis result.

Purpose			Identification of Abstractions		Identification of Requirements	
			Classification of Requirements		Interrelation of Requirements	
Design	*Processing Characteristics*	Degree of Automation	Manual	Semi-Automatic	Automatic	
		Automation Technology	Information Retrieval	Natural Language Processing	Other	
	Knowledge Base Characteristics	Knowledge Origin & Volatility	Imported / Static		Retrieved / Dynamic	
		Structure	Dictionary		Ontology	
		Domain Specificity	Domain-Specific		Domain-Independent	
Evaluation		Evaluation Approach	Proof of Concept	Case Study	Simulation	Controlled Experiment
		Evaluation Constructs (Dependent Variables)	Completeness	Correctness	Efficiency	Other
Knowledge Exchange		Knowledge Grounding	General Knowledge	Design Theory	Mid-Range Theory	Formal Theory
		Knowledge Contribution	Artifact Description	Nascent Design Theory	Well-developed Design Theory	

Figure 14: Analysis Result for Cleland-Huang et al. (2007)

The approach suggested by Casamayor et al. (2010) possesses a lot of similarities to the work presented by Cleland-Huang et al. (2007). It also aims at the identification and classification of NFR in a semi-automatic approach and uses a similar knowledge base and knowledge creation approach. However, their approach differs in its processing characteristics. The authors complement IR and NLP techniques with an Expectation Maximization algorithm (EM). The core idea of this algorithm in the context of RMS is the creation of knowledge from both classified and unclassified requirements. Unlike other mechanisms it requires only a very small number of previously classified requirements in the knowledge base. The proposed system is evaluated in a simulation measuring precision and recall (to assess correctness and completeness), f-measure (a combination of precision and recall in one variable) and accuracy (the proportion of true results; both true positives and true negatives; in the population.). Again, the design is only grounded on general knowledge and contributions are restricted to a description of the artifact without further abstraction or codification of the design.

QAMiner, the system presented by Rago et al. (2011) similarly aims at the identification and classification of NFR in a semi-automated approach. However, their system follows a different knowledge base approach. Instead of a dictionary, QAMiner utilizes domain-specific ontologies, which are imported to the system before discovery starts. To evaluate their system, a simulation using the standard measurements of precision, recall and accuracy is conducted once again. Knowledge exchange is restricted to the usage of general knowledge and a description of the artifact without further theorizing. Figure 15 depicts the overall analysis result.

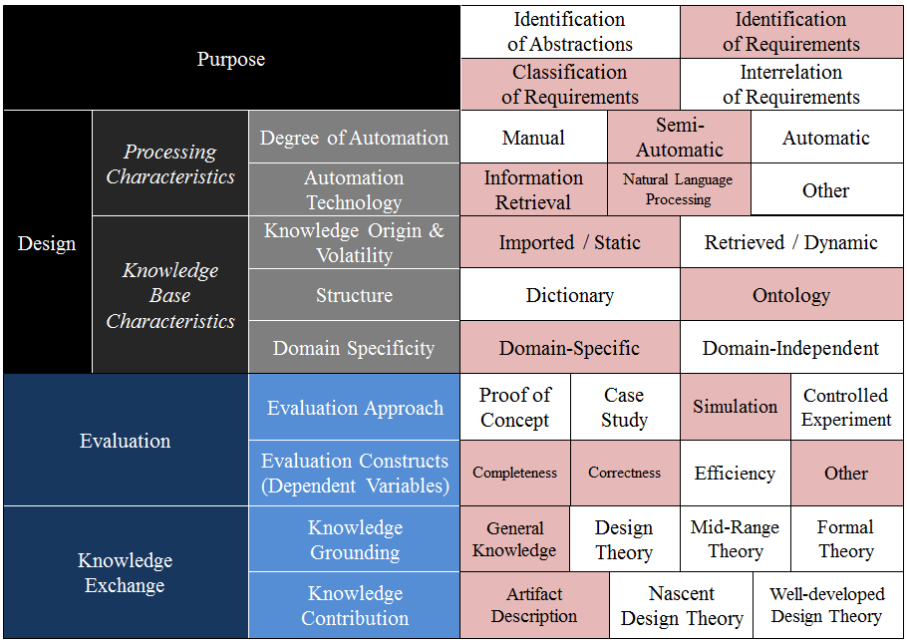

Figure 15: Analysis Result for Rago et al. (2011)

Finally, in the work by Vlas and Robinson (2012), a system to support the identification and classification of requirements for open source software is presented. Unlike the former related works, this system is not restricted to NFR and works in a fully automated fashion. It applies IR and NLP techniques, extended by additional methods to support classification. Imported knowledge in form of ontologies can be used, allowing both domain-specific and domain-independent knowledge items. The system is evaluated in a simulation measuring recall, precision and f-measure. In addition, the

time needed for the automation is measured to assess the efficiency of the approach. Although the authors explicitly claim to follow a DSR approach, knowledge exchange is restricted to the usage of general knowledge and a description of the artifact.

3.7.2 Research Gap Identification

Figure 16 shows the aggregated results for all four works within this analysis. Different shades of red visualize if a characteristic can be observed in many works (dark red), few works (lighter red) or no work (white).

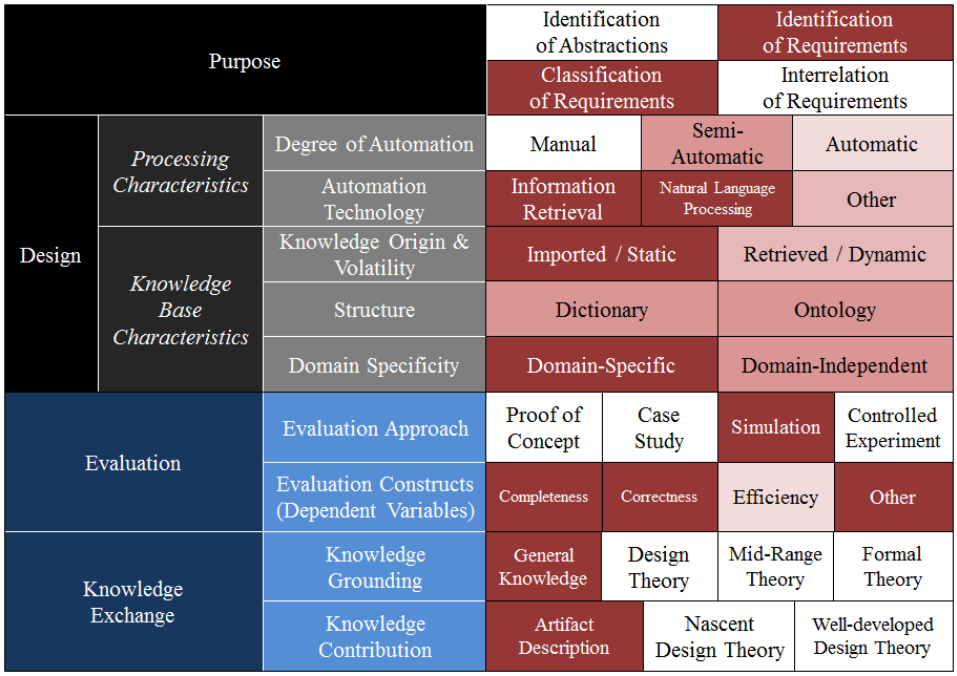

Figure 16: Aggregated Analysis Results for Related Work

The result of the analysis is twofold, showing a heterogeneous picture for the investigated design choices and a homogenous picture for the evaluation and knowledge exchange in the analyzed works. While apparently many different design choices have been investigated, evaluations are focused on simulations comparing the results of the presented system with a previously defined gold standard. Even though these evaluations allow precise measurements of absolute quality criteria, they do not allow a comparison to the as-is situation of manual discovery. Consequently, the question of whether the systems really improve requirements quality and requirements mining

efficiency cannot be answered. Unlike first intuition would tell us, even efficiently working automated requirements mining does not necessarily outperform manual requirements mining. Due to the ambiguity and inconsistency of NLRR, results of automated requirements mining in most cases require manual rework to correct mistakes of the automatism, adapt its findings, or add requirements which were overlooked (Cleland-Huang et al. 2007). Therefore, even automated approaches resulting in high (but not 100%) initial recall and precision might generate larger total efforts as manual discovery if times for rework are also taken into account. Consequently, the mentioned works could be complemented with a study investigating whether the use of an accordant system actually improves individual performance by comparing it to a manual approach.

Furthermore, while the analyzed works include detailed descriptions of their specific implementations, a codification and abstraction of the demands to be fulfilled by the system and the concepts addressing each of these demands is missing. A corresponding conceptualization has been intensively discussed in DSR (Baskerville and Pries-Heje 2010; Gregor and Jones 2007) and enables a generalization of design approaches going beyond the description of specific solutions to specific problems. Applying this approach to RMS, the theoretical contribution drawn from previous works can be extended substantially.

Finally, the suggested systems are not theoretically grounded. They are based on general empirical and non-empirical knowledge drawn from prior studies. These studies might report on situational and non-generalizable settings and experiences and thus do not provide an appropriate basis to conceptualize a design theory with significant reach.

The work described in this thesis intends to address these gaps by 1) deriving a design theory for RMS based on knowledge drawn from both theoretical and non-theoretical sources, 2) implementing an artifact according to this theory, and 3) testing the theory through an evaluation of the artifact comparing a requirements engineer’s system-supported mining productivity with manual discovery.

3.8 Summary

In this chapter, an analysis framework for RDS has been conceptualized and applied to RMS as sub-class of systems. Following an overview, the framework, individual dimensions and characteristics have been introduced and exemplified with existing research. This comprised a depiction of alternative purposes, processing and knowledge base characteristics of RDS as well as different evaluation and knowledge exchange approaches in RDS research. Finally, the framework has been applied to RMS which represent the class of systems to be focused on in the context of this thesis. Finally, the results of this analysis were used to define research gaps which will be addressed in this thesis.

4 Methodology

DSR has become an established approach to enable the conduction of rigorous, design-oriented research in the IS domain. This thesis strives to gain theoretical design knowledge about RMS based on rigorous methodology. Therefore, a DSR approach is followed which will be explicated in the following chapter. For this purpose, first an overview of DSR in IS is provided, discussing artifacts and theories as potential outcomes (or *products*) of DSR and their conceptualization in the design *process*. The dualist nature of design as product and process is then further elaborated presenting examples of process-oriented and product-oriented frameworks to conduct DSR, including a selection of frameworks to be applied in this thesis project. Using the selected process-oriented framework, the research design of the thesis is then presented and finally reflected from an ontological and epistemological perspective.

4.1 Design Science Research in IS

Design Science is rooted in the seminal work by Simon (1969) in which the idea of a science of the artificial to complement natural science is propagated. This science centers around the design (or synthesis) of artifacts by humans and was subsequently applied to IS. In the IS context, different types of artifacts can be differentiated, such as constructs, models, methods and instantiations (March and Smith 1995). According to March and Smith (1995), constructs provide the vocabulary of a domain. For example, tables and relationships are constructs within entity relationship (ER) modeling (Gregor and Jones 2007). Models visualize relationships among constructs. For example, the ER model of an entire database system is a model. Methods can be understood as activities or steps to perform a task. For example, this may be an algorithm to sort data or a guideline to be followed when loading data to a system. Finally, instantiations represent the implementation of artifacts in IS and software development systems (March and Smith 1995). In the context of this thesis, using the taxonomy, an instantiation of a RMS will be designed.

While some scholars characterized DSR as a paradigm which primarily aims at problem-solving through the creation of innovative artifacts (Hevner et al. 2004; March

and Smith 1995), other researchers emphasized the value of a design theory as the core contribution of DSR (Gregor and Jones 2007; Walls et al. 1992). As early representatives of the latter group, Walls et al. (1992) specifically called for the development of design theories, articulating prescriptive knowledge based on theoretical grounds. These prescriptions should describe how an artifact shall be designed in order to achieve a given goal. In response to this call, design theories have been articulated for a diverse range of systems, for example systems to support emergent knowledge processes (Markus et al. 2002), systems that support convergent and divergent thinking (Müller-Wienbergen et al. 2011) or process-based knowledge management systems (Sarnikar and Deokar 2009). Although the call for theoretical contributions of DSR has been emphasized in the current DSR discourse (Gregor and Hevner 2013; Kuechler and Vaishnavi 2012) other scholars have suggested to reduce the complexity of design theories (Baskerville and Pries-Heje 2010) or even questioned the concept of a design theory itself (Hooker 2004). In line with the argumentation of Gregor and Hevner (2013) the author of this thesis takes up the stance that through the abstraction and codification of prescriptive knowledge in a design theory the knowledge contribution and impact of DSR can be significantly improved. Therefore in this thesis, additionally to a RMS instantiation, a design theory for RMS is derived.

The core of the design process comprises a stepwise refinement process in which designers strive to map needs (specified in the function space) to solutions (specified in the attribute space) (Takeda and Veerkamp 1990). The elements of both: the function and attribute space appear, in different terminology, in many design theory frameworks. While elements of the function space are referred to as meta-requirements (Walls et al. 1992), general requirements (Baskerville and Pries-Heje 2010) or design requirements (Müller-Wienbergen et al. 2011), elements of the attribute space are referred to as meta-design (Walls et al. 1992), general components (Baskerville and Pries-Heje 2010) or design principles (Markus et al. 2002; Müller-Wienbergen et al. 2011). In the context of this thesis, the terms design requirement and design principle will be used. While design principles characterize solutions in a technology-agnostic fashion, the implementation of an artifact requires an additional mapping process to technology-dependent features

of the artifact. In the following, the outcome of this process will be referred to as design features.

4.2 Framework Selection and Adaption

Various frameworks have been proposed, describing how DSR should be conducted. While some frameworks take a *process* perspective, depicting for example different phases of DSR research (Nunamaker et al. 1990; Peffers et al. 2007; Sein et al. 2011; Takeda and Veerkamp 1990; Vaishnavi and Kuechler 2007) others provide a *product*-oriented structure, suggesting different components which should be included in the resulting design theory (Baskerville and Pries-Heje 2010; Gregor and Hevner 2013; Gregor and Jones 2007; Kuechler and Vaishnavi 2012; Walls et al. 1992). Baskerville and Pries-Heje (2010) draw an analogy from these two perspectives to the dual nature of theory versus theorizing. In this analogy, a design theory represents the product of theorizing about a specific artifact.

This dualist nature is also inherent to the structure of this thesis: While the research design will be described along the phases of a process-oriented framework, the resulting design theory will be depicted using a product-oriented framework. To choose appropriate process- and product-oriented frameworks, different alternatives have been analyzed. This analysis process, the reasons for selection and the performed adaptions of the original frameworks for the research design of this thesis will be described further on.

4.2.1 Process-oriented Frameworks

Process-oriented frameworks describe DSR from a procedural perspective, differentiating different phases, their sequence and the associated knowledge flows.

An early approach to structure the design process accordingly was presented by Nunamaker et al. (1990). The authors argue that system development represents a valuable research methodology which can complement existing IS research. Their *Process for Systems Development Research* consists of five phases: 1) Construction of a conceptual framework, including an investigation of requirements and the search for new approaches and ideas 2) Development of a system architecture, including the

definition of functionalities, components and their interrelation 3) System analysis & design**,** including the investigation of different design alternatives 4) Implementation of the system (or a prototype), including the actual system development and 5) Observation of system use and experimental evaluation of the system**,** investigating effects of the system's usage.

The process provided by Nunamaker et al. (1990) represents an abstract model to structure DSR activities in distinct phases. However, the actual conduction of DSR is not further explicated and therefore leaves many questions open (Peffers et al. 2007).

As a consequence, in a more recent work, Peffers et al. (2007) suggest their Design Science Research Methodology (DSRM). The comprehensive framework includes principles, practices, and procedures and is made up of six sequential phases: 1) Problem identification and motivation 2) Definition of the solution objective 3) Design and Development 4) Demonstration of the artifact 5) Evaluation of the artifact and 6) Communication of the research results. The authors point out that DSR projects can be initiated from different entry points: problem-centered, objective-centered, design and development-centered and client/context-centered. In contrast to other frameworks, Peffers et al. (2007) explicitly point out the importance of communicating disciplinary knowledge to both research and practice communities in form of publications geared towards each target group.

Moreover, they differentiate the demonstration of the artifact in a suitable context from the artifact evaluation in which its effectiveness and efficiency are measured. In the framework applied in this thesis, the latter aspect will be explicitly considered through a distinct demonstration phase between the development and evaluation of the artifact.

The framework which guided the design process of this thesis is based on the *General Methodology of Design Science Research (GMDSR)* as suggested by Vaishnavi and Kuechler (2007). The framework is an extension of the *design cycle* proposed by Takeda and Veerkamp (1990). It includes process steps, their outputs and the related knowledge flows. Starting with the "Awareness of Problem" phase, in which the motivation for the DSR project is drawn from a real-world problem, a tentative design is conceptualized in the "Suggestion" phase. Based on this concept, in the "Development" phase the artifact is implemented. After measuring the artifact's effectiveness in the

"Evaluation" phase, a final conclusion is drawn from the results and fed back to the first phase to re-iterate.

Similarly to the DSRM, Vaishnavi and Kuechler (2007) give explicit prescriptions about the conduction of DSR. In addition, the authors emphasize the explicit reflection of design principles and other design results as well as an iterative, evaluation-driven approach. These two characteristics properly match the goals of the research project at hand. First, through the continuous reflection and adaption of design results, an appropriate mechanism to derive a sound design theory is provided. Second, through multiple iterative evaluations, a tight integration of potential users can be accomplished which eases the accomplishment of the artifact's final goal to increase requirements mining productivity. Therefore, the GMDSR was selected as guiding overall approach for this research project. For the context of this thesis, the GMDSR was slightly extended by a demonstration phase between the development and evaluation of the artifact, as suggested by Peffers et al. (2007). This demonstration phase allows the collection of informal feedback from experts in addition to formal evaluations. The resulting process-oriented framework is depicted in Figure 17.

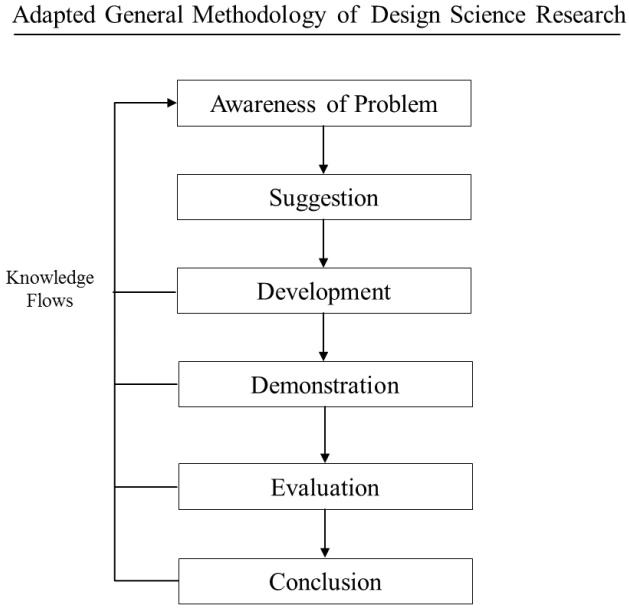

Figure 17: Adapted GMDSR, Based on Vaishnavi and Kuechler (2007)

4.2.2 Product-oriented Frameworks

The need for an ISDT was first articulated by Walls et al. (1992). Following Simon's call to develop a science of the artificial (Simon 1969), they argue that the IS discipline should articulate and develop prescriptive theories to enable the development of more effective IS. The according theories should integrate normative and descriptive theories and describe design paths to be followed. Due to their prescriptive nature, ISDT are different from explanatory and predictive theories. Walls et al. (1992) propose seven components of an ISDT out of which four describe the design product:

- *Meta-Requirements* which describe the class of goals the theory should be applied to.
- *Meta-Design* characterizing the class of artifacts to address the meta-requirements.
- *Kernel theories* including theories from natural and social science which can guide the design.
- *Testable design product hypotheses* which can be utilized to test if the meta-design actually addresses the meta-requirements.

The ISDT proposed by Walls et al. (1992) provided the common basis for various other product-oriented DSR frameworks. Gregor and Jones (2007) argue that although design work and design knowledge in IS are important for both research and practice, little attention has been paid to the problem of specifying design theory. Based on the ISDT proposed by Walls et al. (1992) and further streams of thought on design research (e.g., Simon's (1969) reflections on a science of the artificial) they suggest an anatomy of a design theory consisting of eight separate components: 1) *Purpose and scope:* This component describes "what the system is for" by depicting the set of meta-requirements or goals that specify the class of artifact to which the theory applies. Furthermore the scope, or boundaries, of the theory are defined. 2) *Constructs:* The theory's entities of interest, for example relations would be constructs in a design theory of relational databases. 3) *Principles of form and function:* The abstract "blueprint" or architecture of the associated IS artifact. 4) *Artifact mutability:* The extent to which changes to the artifact are encompassed by the theory 5) *Testable propositions:* Truth statements about

the design theory (e.g., predictions about outcomes that can be tested in experiments). 6) *Justificatory knowledge:* Underlying knowledge or theory to give a basis and explanation for the design. 7) *Principles of implementation:* A description of how to implement the theory in specific organizational contexts 8) *Expository instantiation*: The implementation of the artifact, providing both a physical representation of the theory and a vehicle to test it.

Baskerville and Pries-Heje (2010) argue that characteristics of design theories as they are discussed in other papers are overly complicated and show that for example the incorporation of kernel theories and testable propositions into design theories might not be applicable or beneficial to all DSR projects. In contrast, the authors seek the simplest possible delineation of a design theory and do this by differentiating between design practice theories which describe the building process of the artifact and explanatory design theories, describing the artifact itself. To determine the minimal components of an explanatory design theory, they collect design theory characteristics from several works. According to their analysis, design theory is assumed to be

- prescriptive, focusing on improving things in contrast to understanding things
- practical, being a basis for action to solve problems
- principles based, defining principles both to guide the development process as well as the architecture of the artifact
- a dualist construction, describing both a process and a product.

Explanatory design theories only describe the product part of this dualist construction and are limited to two components: General requirements and general components. *General requirements* can be described as conditions or capabilities that must be met by the artifact. *General components* describe the abilities or qualities which represent a generalized solution meeting the general requirements.

The resulting design theory of this thesis is presented along the eight components suggested by Gregor and Jones (2007). Unlike other product-oriented frameworks, this structure allows a complete and transparent coverage of outcomes from all phases of a DSR project. Table 1 depicts the differences. The theory components suggested by Walls et al. (1992) can only be related to three of the six phases (Awareness of the Problem, Suggestion, Evaluation). Similarly, the structure suggested by Baskerville and

Pries-Heje (2010) can only be used to describe the outcomes of two phases (Awareness of the Problem, Suggestion). In contrast, the theory components of Gregor and Jones (2007) can be mapped to each of the DSR phases, allowing a holistic description of design outcomes.

Design research phases[10]	**Design Theory Components**		
	Walls et al. (1992)	*Gregor and Jones (2007)*	*Baskerville and Pries-Heje (2010)*
Awareness of Problem	Meta-requirements	Purpose and scope, Justificatory knowledge	General requirements
Suggestion	Kernel theories, Meta-design	Justificatory knowledge, Principles of form and function	General components
Development	-	Constructs, Expository instantiation	-
Demonstration	-	Constructs, Expository instantiation	-
Evaluation	Testable design product hypotheses	Testable propositions	-
Conclusion	-	Artifact mutability, Principles of implementation	-

Table 1: Assignment of DSR Theory Components to Design Phases

4.3 Research Design

In the following, the overall research design of this thesis project will be described along the phases of the adapted GMDSR. Further details on the artifact design process will be provided in chapter 5. Details on the methodology for the artifact evaluation will be provided in chapter 6.

Design research suggests to design artifacts in an iterative fashion enabling continuous reflection and incremental refinement of the design results (Hevner et al. 2004; Takeda and Veerkamp 1990). Consequently, in this thesis project, two design cycles have been

[10] Design research phases based on the GMDSR by Vaishnavi and Kuechler (2007).

conducted as depicted in Figure 18. In the following, the utilized methods and performed activities in each of the design cycles are depicted in more detail.

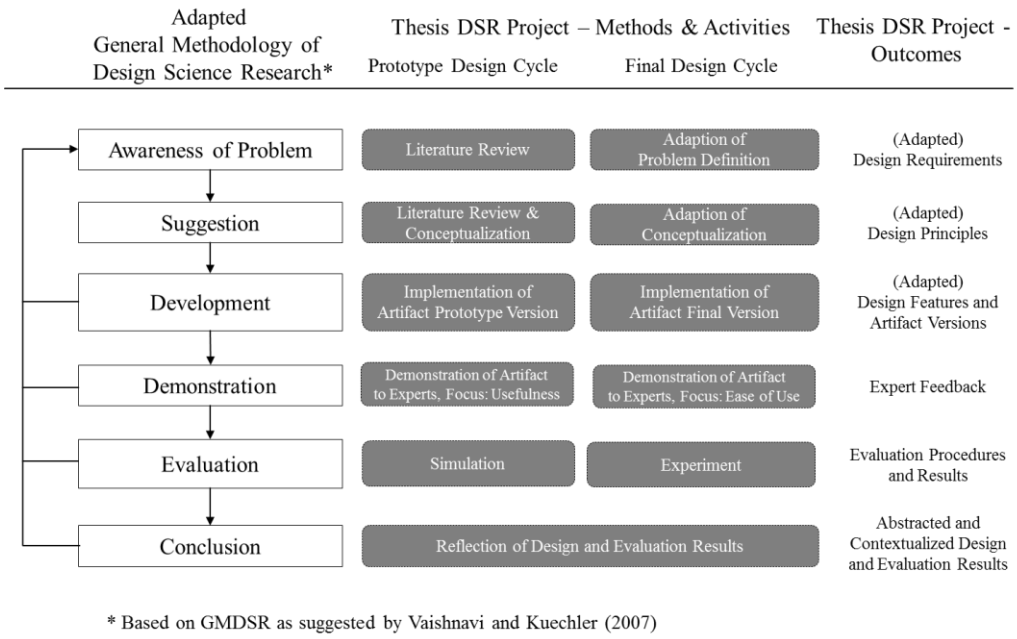

Figure 18: Research Design[11]

4.3.1 Prototype Design Cycle

The prototype design cycle of the research project was initiated by an intensive literature review to create *problem awareness* resulting in design requirements for the artifact to be built. Based on these design requirements, a second literature review was conducted to identify general knowledge and theories which can be applied to address the identified problem. Using this knowledge, preliminary design principles were conceptualized in the *suggestion* phase. These design principles were then mapped to design features and were finally implemented in a prototype version of the artifact during the *development* phase. To collect informal feedback on the artifact's usefulness, it was then presented to requirements engineering experts in several *demonstration* sessions. In the following, the prototype was analyzed in a quantitative *evaluation*. This evaluation focused on the interplay of the two main design principles which was investigated in multiple simulation runs. Results of the evaluation and the

[11] The structure of the research design follows the GMDSR by Vaishnavi and Kuechler (2007).

demonstration sessions were analyzed and reflected (along with the design results) during the *conclusion* phase.

4.3.2 Final Design Cycle

During the final design cycle, the initial problem definition and conceptualization were adapted based on the design, demonstration and evaluation results of the previous cycle. This led to an adjustment of the initial design requirements and design principles. The adapted design principles were again mapped to design features resulting in a modification of the artifact. To improve the artifact's ease of use, it was presented to usability experts in several demonstration sessions which resulted in multiple small adaptions. Then the final artifact version was evaluated in an experiment. This evaluation consisted of a lab experiment, conducted with students and a replication of the experiment in a field environment, involving experts. By these experiments, the effects of each design principle on the performance of individual requirements engineers were measured. Finally, the design and evaluation results were again abstracted and contextualized.

4.4 Ontological and Epistemological Reflections

In the following, the presented research design shall be reflected from an ontological and epistemological point of view to point out the core assumptions of the research. In this context following the definitions by Vaishnavi and Kuechler (2007), an ontological stance describes the underlying assumption about the nature of reality (e.g., what is real and what is not) while an epistemological stance describes the underlying assumption about the nature of knowledge (e.g., how knowledge can be derived).

In DSR projects, questions of ontology and epistemology are often treated rather implicitly (Niehaves 2007). Nevertheless, in the existing discourse, some scholars see Design Science as a third paradigm in addition to positivism and interpretivism (Vaishnavi and Kuechler 2007). Other researchers emphasize the compatibility of DSR with existing research paradigms, for example positivism (Marshall and Mckay 2005; Niehaves 2007). An argument for the former view is that design science aims at gaining

knowledge through the creation of artifacts which is epistemologically different from other paradigms (Vaishnavi and Kuechler 2007).

However, as Niehaves (2007) points out, the prescriptive knowledge gained in DSR is inevitably embedded in further types of justificatory knowledge such as theoretical, descriptive and empirical knowledge. Additionally, the knowledge contribution of DSR is often not restricted to the knowledge embedded in the artifact as explained in section 3.6.2 but can also comprise theoretical knowledge. Consequently, depending on the approach to gain this theoretical knowledge, DSR can be conducted following a positivistic approach (Hevner et al. 2004; March and Smith 1995) or other existing paradigms. Marshall and Mckay (2005) for example point out that interpretive or critical approaches to DSR, which aim at understanding and analyzing the impacts of an artifact's introduction and usage in the field, can similarly be applied.

The research in this thesis follows a positivistic paradigm which will be explained in the following, analyzing the general ontological assumption and the epistemological stance of this research. The basic *ontological assumption* of positivistic research is the existence of a single, objective reality, which comprises facts that can be accessed and observed by the researcher (Carson et al. 2001; Vaishnavi and Kuechler 2007; Weber 2004). In the presented research design, the identified NLR and their classification, as well as characteristics of the discovery process itself (e.g., the time needed to accomplish requirements discovery) can be seen as facts which are directly observable by the researchers. This stance is also expressed in the choice of quantitative evaluation methods like simulations and experiments which are generally associated with positivistic research (Marshall and Mckay 2005).

From an *epistemological perspective*, positivistic research predominantly aims at deriving theoretical knowledge through the definition and test of hypotheses and a research focus on generalization and abstraction (Carson et al. 2001). Furthermore, there is a concentration on description and explanation, while for example interpretative approaches rather focus on understanding and interpretation (Carson et al. 2001; Vaishnavi and Kuechler 2007). In the research at hand, assumed effects of design principles on requirements mining productivity will be formulated as hypotheses. Subsequently, through the instantiation of these design principles in an artifact, the

hypotheses can be tested. The conceptualization of the artifact using generic design principles for a class of systems (RMS) favors the generalization and abstraction of the results. The derived theoretical knowledge can help to explain how these design principles, when implemented in an artifact, can affect requirements mining productivity.

Table 2 summarizes the ontological and epistemological stance of this thesis.

Perspective	Thesis Stance
Ontological	Single, objective reality exists Facts can be accessed and observed by the researcher
Epistemological	Derive theoretical knowledge through the definition and test of hypotheses Research focus on generalization and abstraction Concentration on description and explanation

Table 2: Ontological and Epistemological Stance of the Thesis

4.5 Summary

In this chapter, an overview of DSR as the overall methodology of this thesis was provided. Design science terminology to describe different types of artifacts and elements of their conceptualization (e.g., design requirements) in the context of this thesis has been introduced. Moreover, the dualist nature of design, being both a process and a product has been discussed along the historic development of the DSR paradigm. Subsequently, process- and product-oriented DSR frameworks were presented. This illustration resulted in a selection of two frameworks which will be used in the context of this thesis, an adapted version of the GMDSR suggested by Vaishnavi and Kuechler (2007), to structure the design process and the eight components of a design theory proposed by Gregor and Jones (2007) to structure the design product. Afterwards, the research design of this thesis was depicted using the adapted GMDSR as a blueprint for two design cycles. Finally, the ontological and epistemological stance of the thesis was discussed, characterizing the positivistic nature of the study.

5 Artifact Design[12]

As previously introduced, Gregor and Jones (2007) distinguish eight components of an ISDT: (1) purpose and scope of the theory, (2) the constructs that are of interest to the theory, (3) the principles of form and function (the blueprint or architecture of the artifact), (4) the artifact's mutability (the extent to which changes to the artifact are encompassed by the theory), (5) a set of testable propositions or hypotheses, (6) justificatory knowledge to give a basis and explanation for the design, (7) principles of implementation, and (8) a physical instantiation of the artifact.

This thesis presents each of these eight components for a RMS design theory yet in a slightly adapted order and naming. The order was changed to be able to trace the artifact's conceptualization in its actual sequence. The naming was adapted to provide a consistent and homogenous terminology for the outcomes of each conceptualization phase: design requirements[13] as the outcome of the *problem awareness* phase, design principles[14] as the result of the *suggestion* phase and design features[15] as the capabilities of the artifact implemented in the *development* phase. These changes result in the following structure: In section 5.1, based on justificatory knowledge, the purpose and scope of the theory's artifact is presented and distilled to distinct design requirements. From these design requirements, applying additional justificatory knowledge, design principles are derived in section 5.2. In the final artifact conceptualization step, design principles are mapped to specific design features which are presented within their expository instantiation, including a summary of the conducted demonstration sessions (section 5.3). The depiction of the design theory will be completed with a description of the principles of implementation, the artifact's mutability and the testable hypotheses for the experiment evaluation of the artifact (sections 5.4 to 5.6).

[12] Parts of this chapter have been published in Meth et al. (2012b).
[13] Design requirements are referred to as *meta-requirements* by Gregor and Jones (2007).
[14] Design principles are referred to as *principles of form and function* by Gregor and Jones (2007).
[15] Design features are referred to as *constructs* by Gregor and Jones (2007).

5.1 Purpose and Scope

The proposed design theory has the purpose to give explicit prescriptions about how to develop systems that support requirements mining from NLRR to improve requirements mining productivity. Productivity is usually conceptualized as an input-output ratio with the work output as the numerator and the work input as the denominator of the ratio (Cosmetatos and Eilon 1983). In the case of requirements mining, the quality of the elicited requirements represents the work output whereas the invested mining effort represents the work input. The quality of requirements determined by RMS is usually assessed by a combined measurement of requirements' completeness and correctness (Casamayor et al. 2010; Cleland-Huang et al. 2007; Gacitua et al. 2011). The invested mining effort can be measured by the time required for the mining process. In general, mining productivity will be improved when either a) the requirements quality is increased or b) the mining effort is decreased. The conceptualization in the following sections will derive design requirements, design principles and design features for a RMS serving this purpose.

The proposed class of systems might be applied to a wide range of NLRR. Sources include the outcomes of formal requirements collections (e.g., from interviews or workshops), informal requirements requests (e.g., emails or blog entries), or texts which were originally created for other purposes (e.g., test protocols or support messages).

Furthermore, RMS can be applied in the context of various software and requirements engineering methodologies. For example, as outlined in section 2.2, the systems can support requirements mining in user-centric approaches focusing on a tight integration of users in the development project as well as market-driven approaches in which a myriad of informal requirements statements is submitted rather anonymously. In both cases the nature of the requirements mining task remains the same: A requirements engineer (or a system) needs to scan through the provided NLRR to identify and classify requirements. Doing this, two questions are repeatedly answered for the processed texts: Does this text passage, sentence or word represent a requirement? And if so, which kind of requirement is it? In the following section, this iterative process is further investigated, focusing specifically on system-supported requirements mining.

5.1.1 Justificatory Knowledge

Figure 19 depicts the basic steps of system-supported requirements mining which the thesis is based on. Starting from the provision of NLRR, requirements are identified and classified by the RMS in a background process resulting in *proposed* requirements. In the following, an interactive approval process is performed, driven by the requirements engineer. This process results in *approved* (and rejected) requirements.

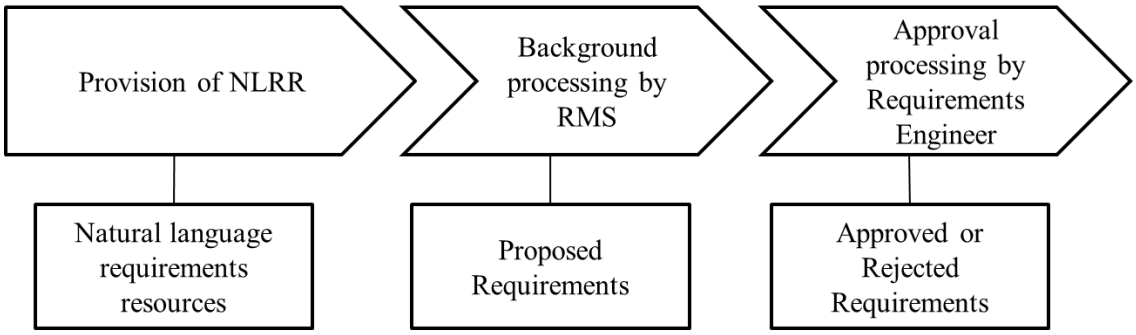

Figure 19: RMS-Supported Requirements Mining Process

Through the determination of proposed requirements, the RMS supports requirements engineers in answering the two previously formulated questions: RMS advise requirements engineers concerning what is a requirement and how to classify it. Therefore, on an abstract level, the process can be seen as a series of consecutive decision tasks in which the RMS acts as an advisor and a requirements engineer as the advice-taker. In this analogy, the assignment of a text passage to a specific requirements category can be seen as a single decision task which is repeatedly performed throughout a NLRR. Decision making theory characterizes decision tasks according to multiple characteristics, amongst others the decision task type (choice vs. judgment tasks), the number of advisors (one vs. multiple), the advice trigger (solicited vs. unsolicited advice) and the degree of interaction between advisor and judge (low vs. high interaction) (Bonaccio and Dalal 2006). Reflecting on the characteristics introduced above, RMS' support of requirements mining can be characterized as a decision process consisting of choice tasks (assignment of distinct requirements categories) given by a single advisor (the RMS) following a solicited but low interaction.

To derive specific design requirements for RMS, it is important to understand the general goals associated with the requirements mining process. The generalization and abstraction of the process to a series of decision making tasks, provides an approach to identify these general goals. Decision makers follow different goals when confronted with a decision task. First, they strive to reach a good or even optimal decision. Therefore, different strategies to optimize *decision quality* have been proposed (Wang and Benbasat 2009). However, additionally to decision quality, the idea that decision making is also influenced by considerations of cognitive effort has been discussed since the seminal works of Simon (1957). Simon coined the concept of Bounded Rationality which suggests that human decision makers are limited by multiple factors impeding the achievement of an optimal decision, including their cognitive processing capacities (Simon 1957). While Simon discusses *cognitive efforts* rather as a limitation leading to suboptimal decision results, cognitive efforts were found to also influence the choice of a decision strategy. Decision strategy selection is often explained using contingency models in which a cost and benefit tradeoff determines strategy choice (Beach and Mitchell 1978; Payne 1982). According to these models, decision makers follow the dual goal to maximize decision quality and at the same time minimize their cognitive effort.

To optimize the outcomes of this tradeoff, different types of decision support systems (DSS) have been proposed (Silver 1991) and effects of the usage of DSS on decision behavior have been investigated (Todd and Benbasat 1991, 1999). DSS aim at improving decision results through the provision of advice[16], building on the idea that advice characterized by high *advice quality* will result in decisions with a high decision quality (Gardner and Berry 1995; Yaniv 2004). Ideally, at the same time cognitive effort will decrease, as the DSS already prepares the decision and the relevant information for the decision maker. However, while DSS can improve decision quality and reduce cognitive effort, the systems may also restrict users in their decision behavior which has been termed as "system restrictiveness" (Silver 1988). *System restrictiveness* is defined as the extent to which decision strategies are pre-selected through the DSS, offering the

[16] In most studies advice is defined as a type of recommendation from the advisor, favoring a particular option (Bonaccio and Dalal 2006).

decision maker only a limited choice of strategies which may not include his (or her) preferred ones (Silver 1988). Therefore, when implementing decision aids, designers need to carefully consider that the benefits of a decision aid (e.g., reduced cognitive effort) are not overcompensated by its restrictions.
Table 3 summarizes goals of human decision makers and design requirements of DSS addressing them.

Goals of Human Decision Makers	Design Requirements of DSS
Maximize *decision quality*	Increase *decision quality* by providing advice with high *advice quality*
Minimize *cognitive effort*	Reduce *cognitive effort* of human decision maker by providing decision support
Maintain *control* over decision strategy selection	Minimize *system restrictiveness* by allowing users to control the strategy selection

Table 3: Goals of Human Decision Makers and Design Requirements of DSS

Wang and Benbasat (2009) investigated each of these design requirements as a perceived factor determining the intention to use decision aids. In their study, decision aids are components of e-commerce platforms which are used to elicit consumer preferences, automate their processing, and provide corresponding product advice. They hypothesize that perceived advice quality, perceived cognitive effort and perceived restrictiveness determine the intention to use decision aids. Based on their experimental results, all three factors showed significant effects on the intention to use a decision aid.

As previously depicted, the requirements mining process can be seen as a series of consecutive decision tasks in which the RMS acts as an advisor and a requirements engineer as the advice-taker. Therefore, the identified design requirements for systems supporting decision making in general are assumed to also be applicable to systems supporting decision making in the context of requirements mining. Consequently, in the following the identified design requirements for DSS will be related to the specific context of requirements mining, treating RMS as a sub-class of DSS.

5.1.2 Design Requirements of RMS

DSS aim at improving decision quality through the provision of high quality advice. Analogously, the quality of requirements proposed by a RMS can be expected to determine the quality of requirements approved by the requirements engineer. As introduced earlier, RMS require a knowledge base to be able to identify and categorize proposed requirements. In general, the quality of requirements proposed by RMS mainly depends on the contents of the knowledge base used for the background mining process (Casamayor et al. 2010; Cleland-Huang et al. 2007). An extensive knowledge base with correctly classified requirements has been found to result in a high quality of proposed requirements (Casamayor et al. 2010; Cleland-Huang et al. 2007). Therefore, the design focus of many RDS has been put on the improvement of *advice quality* through the provision of high quality *proposed requirements* (Gacitua et al. 2011; Goldin and Berry 1997; Kiyavitskaya and Zannone 2008). However, revisiting the analogy to decision making, high quality proposed requirements only represent a prerequisite but not the final goal of the process. Only an increase in the quality of *approved requirements* will address requirements engineers' goal of achieving a high *decision quality.* As a consequence, the following design requirement is derived:

> ***DR1. Increase quality of approved requirements.*** *The requirements mining process should be supported by systems which aim at improving the quality of approved requirements.*

To reduce the *cognitive effort* of requirements engineers during the requirements mining process, first the question needs to be answered which phases of this process depend on human cognition. Most RDS implement advice-giving in a background process without any user interaction. The proposed requirements resulting from this background process are then presented to the requirements engineer for manual approval. Consequently, during the actual mining process, the cognitive effort of the requirements engineer is only determined by the efforts to transform proposed requirements into approved requirements. In some cases, this might still involve intensive reflection. However, in

most cases, cognitive efforts will be reduced from an active consideration of all decision options to a rather passive approval of the given advice.

Additionally to the actual decision making process, taking a holistic view on the cognitive effort of the requirements engineer, manual efforts to create and maintain the knowledge base have to be taken into account as well and should be minimized. In summary, the following design requirement is derived:

> ***DR2. Decrease cognitive effort to execute and prepare requirements mining.*** *The requirements mining process should be supported by systems aiming at a decrease of the cognitive effort to transform proposed requirements into approved requirements as well as the cognitive efforts to create and maintain the underlying knowledge base.*

As presented in section 3.3.1, RDS can provide different degrees of automation. Some systems only support manual requirements discovery (Abrams et al. 2006; Ossher et al. 2009), while others restrict requirements engineers to use the system in a fully automated mode (Gacitua et al. 2011; Goldin and Berry 1997; Kiyavitskaya and Zannone 2008). Recapturing decision makers' goal to maintain control over the decision strategy selection and limit *system restrictiveness*, RMS should allow requirements engineers enough flexibility to choose an appropriate type of processing support.

Furthermore, system restrictiveness should also be limited concerning the knowledge to be used during requirements mining. As introduced in section 3.4, RDS can use different types of knowledge (e.g., imported knowledge vs. retrieved knowledge). To limit system restrictiveness, different types of knowledge should be usable during requirements mining. Consequently:

> ***DR3. Limit system restrictiveness during requirements mining.*** *The requirements mining process should be supported by systems aiming at minimal processing restrictions concerning the conduction of requirements mining.*

In the following, the process of deriving design principles from the previously identified design requirements is described.

5.2 Conceptualization

Similarly to the previous section, to derive design principles for RMS, an analogy to decision making is drawn, based on existing theory on decisional guidance.

5.2.1 Justificatory Knowledge

To address the design requirements formulated in the last section, the question arises which type of system support to choose. Previously, the requirements mining process was abstracted to a general decision making process and an analogy between RMS and DSS was drawn. This analogy shall be further elaborated in the following, introducing types of decisional guidance implemented in DSS from an existing typology. For the further conceptualization, those types of guidance will be identified, which match the previously described design requirements. Based on this selection, design principles will be derived in the subsequent sections

5.2.1.1 Types of Decisional Guidance

Silver (1991) describes decisional guidance (DG) as the way a DSS informs or influences decision makers in the structuring and execution of decision tasks. The author defines a typology of DG based on three different characteristics. First, a differentiation concerning the *targets of guidance* can be made. Silver (1991) distinguishes DG to structure the decision making process and DG to execute it. The former supports decision makers in selecting the right approach, method or strategy to make a decision. For example, structural guidance could support choosing an existing decision strategy such as additive compensation or elimination by aspects[17]. Subsequent

[17] According to Todd and Benbasat (1999), additive compensation is a strategy in which each alternative is evaluated individually along all relevant attributes. The decision maker assigns a weight and a value to each attribute and then determines the total score of an alternative. Elimination of aspects is a strategy based on a comparison of attribute values to threshold values. Alternatives are eliminated if one of their attributes does not meet a threshold

to strategy selection, executional guidance can help decision makers in the operational conduction of the decision task. For example, the system could prompt the user to enter values or calculate the overall value of an alternative. Second, the typology differentiates alternative *forms of guidance*. DG might be implemented in a suggestive or informative way. Suggestive guidance recommends decision makers which strategy to choose or which values to enter. Informative guidance on the contrary only provides decision makers with decision-relevant information without recommending a choice. For example, a description of the range of possible input values could be regarded as informative guidance. Finally, Silver (1991) distinguishes different *modes of guidance*, describing the ways DG is generated. DG can be predefined, dynamic or participative. Predefined guidance consists of context-specific information or recommendations which are defined upfront by experts or regular users and imported into a knowledge base. In contrast, dynamic guidance is an adaptive mechanism which generates information and recommendation based on the actual system usage. DG (similarly to RMS) usually utilizes knowledge bases to generate advice. Dynamic guidance iteratively builds up additional knowledge base contents. Finally, participative guidance puts a stronger focus on users' participation in the determination of guidance-specific content. For instance, in a decision task based on a decision table with different alternatives, participative guidance could be implemented by adding functionality to manipulate the table through ordering or summation. In the following, the presented types of guidance will be associated with the requirements mining process and the identified design requirements.

5.2.1.2 Associating Decisional Guidance to Requirements Mining

Investigating the *targets of guidance* in the context of requirements mining, it is worthwhile revisiting the process to be conducted. Requirements mining, as previously introduced, can be seen as a series of consecutive decision tasks in which the assignment of a text passage to a specific requirements category represents a single decision task which is repeatedly performed. Although this task requires substantial knowledge in requirements engineering and the corresponding business domain, it is a standardized procedure, executed rather similarly every time it is performed. Therefore,

unlike other decision tasks, it does hardly require support to structure the decision task in advance of each single decision. But, especially due to the large number of decisions to be made, it definitely requires execution support to reduce requirements engineers' cognitive efforts and maintain a high level of quality.

To determine appropriate *forms of guidance*, an empirical study conducted by Parikh et al. (2001) provides interesting results. The authors investigated how different forms of guidance influence decision quality and decision efficiency in an experiment study involving 141 participants. In this study, participants were asked to examine a historical data set and identify key characteristics of it. Based on the identified characteristics, they should assign a suitable forecasting model to process this data set. In its basic constituents (identification of decision-relevant information and subsequent classification of this information) the decision task resembles the decisions involved in the requirements mining process. Parikh et al. (2001) found out that suggestive guidance outperformed informative guidance concerning both, decision quality and decision efficiency. The two dependent variables used in their study (decision quality and decision efficiency) can be associated with the previously derived design requirements DR1 and DR2. Revisiting the introduced analogy to requirements mining, increased decision quality is associated with increased quality of approved requirements and increased decision efficiency can be associated with a decrease in mining efforts. Therefore, suggestive guidance is expected to be an appropriate means to address DR1 and DR2.

In the same study, Parikh et al. (2001) analyzed how different *modes of guidance* affect decision quality and decision efficiency. Their central finding was that dynamic guidance outperformed predefined guidance concerning decision quality and decision efficiency. In analogy to the argumentation for the form of guidance, by associating decision quality and decision efficiency with DR1 and DR2, dynamic guidance can be expected to result in an increased quality of approved requirements and a decrease of mining efforts. Parikh et al. (2001) investigated different modes of guidance as exclusive alternatives. However, dynamic, predefined and participative guidance can also be combined to improve results. When applied complementary to dynamic guidance, predefined and participatory guidance can provide additional advice and

hereby further increase decision quality and decision efficiency. Furthermore, revisiting the design requirement DR3, additionally applied participative guidance can allow a higher degree of freedom to the final decision maker which might reduce his perceived system restrictiveness. Therefore, in the context of requirements mining a complementary use of different modes of guidance is proposed.

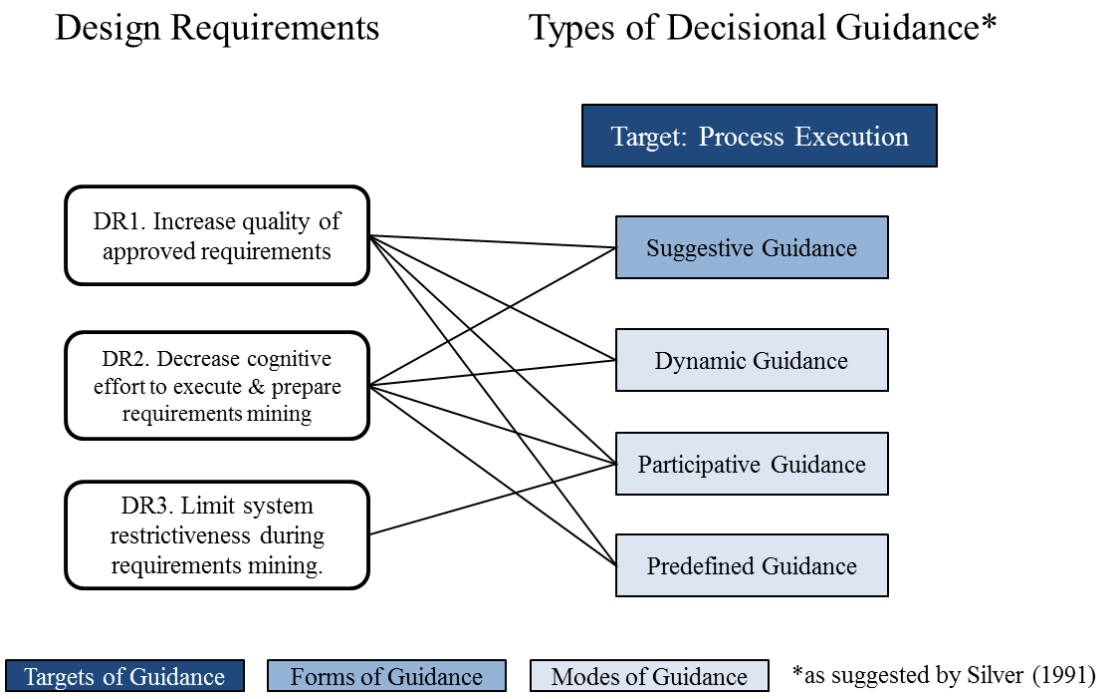

Figure 20: Associating Design Requirements to Different Types of DG

5.2.2 Design Principles of RMS

Which design principles can be derived from the identified types of DG to address the initial design requirements? In the context of requirements mining, suggestive guidance can be accomplished by means of *automation*, resulting in a set of requirements proposed by the automation algorithm. During the mining of requirements from NLRR, a text is analyzed to identify relevant words and assign them to requirements categories. This process can be decomposed into single steps which are repeatedly performed and follow specific rules (Ambriola and Gervasi 2006). Consequently, they can be translated into algorithms that can automatically be executed by a computer. Automation addresses the first two design requirements identified in the previous section. First,

automation can increase the quality of approved requirements. Reflecting the analogy to decision making, the quality of approved requirements can be expected to be positively affected by the quality of proposed requirements. A carefully developed algorithm can identify a significant percentage of the requirements within a natural language document and can identify requirements which may have been overlooked in a pure manual discovery process (Berry et al. 2012). Moreover, as the algorithm will not suffer from fatigue or decreasing motivation as a human being might do, each part of a document will be treated with equal attention which can additionally contribute to a more complete set of requirements. Second, automation should lead to a decrease in cognitive efforts, as each automatically classified requirement does not need to be identified and categorized manually by the requirements engineer.

During the manual approval of proposed requirements, the requirements engineer decides whether to follow the advice of the RMS or not. In the case of requirements mining, the ambiguity and inconsistency of NLRR often requires a third option: Requirements need to be adapted or added. In these cases, the automatism needs to be complemented with functionality supporting manual discovery (Berry et al. 2012; Kiyavitskaya and Zannone 2008). However, any manual adaptation of automatically identified requirements represents additional effort for the requirements engineer. To limit this effect, functionality for manual identification and classification should provide a high level of usability to enable efficient operations. Additionally to the effects on DR1 and DR2, capabilities for manual requirements identification and classification also represent a way to enable participative guidance. Allowing the requirements engineer further freedom in the mining process can hereby also minimize system restrictiveness (DR3). In summary, to support the mining process the following design principle is proposed:

> ***DP1. Semi-Automatic Requirements Mining:*** *RMS should support efficient automatic and manual requirements mining within NLRR.*

As illustrated earlier, automated requirements mining requires an underlying knowledge base containing terms and a categorization of these terms. Revisiting the identified

design requirements and relating them to knowledge creation, a corresponding design principle should provide answers to the following questions: 1) How can the quality of knowledge be increased and 2) How can (cognitive) efforts of the requirements engineer to create knowledge be decreased?

Starting with the first question, the quality of the knowledge base can be assessed by its completeness and correctness. A more extensive knowledge base will only conclude in better mining results if a sufficient level of correctness is sustained. One approach to augment the knowledge base with according knowledge is the supplementation of domain-specific knowledge. Documents that originate from the same domain share specific requirements elements which are not included in general knowledge (Lemaigre et al. 2008) (e.g., the data field "frequent flyer number" in the domain "traveling"). Similarly, specific writing styles or standards for single projects or entire organizations can result in needs to extend imported knowledge (Cleland-Huang et al. 2007). There are different ways how domain-specific knowledge can be generated. Addressing the design requirement behind the second question, the proposed design is supposed to support knowledge generation in a way that minimizes efforts for the requirements engineer. Therefore, additionally to predefined guidance, a mechanism to support dynamic guidance is needed. This can be realized by feeding back results of previous requirements mining activities into the knowledge base and hereby create and use retrieved knowledge additionally to imported knowledge. Although this process requires some supervision to sustain quality, this type of knowledge supplementation can be expected to be a lot more efficient than manual creation of domain-specific knowledge. Consequently, the following design principle is proposed:

> ***DP2. Usage of imported and retrieved knowledge:*** *RMS should use both manually imported and automatically retrieved knowledge during automatic mining.*

An overview of the conceptualization process from design requirements via types of DG to design principles is provided in Figure 21. The figure shows how the identified

design requirements of RMS can be addressed by different types of DG. Furthermore, it outlines which design principle of RMS is associated with which type of DG.

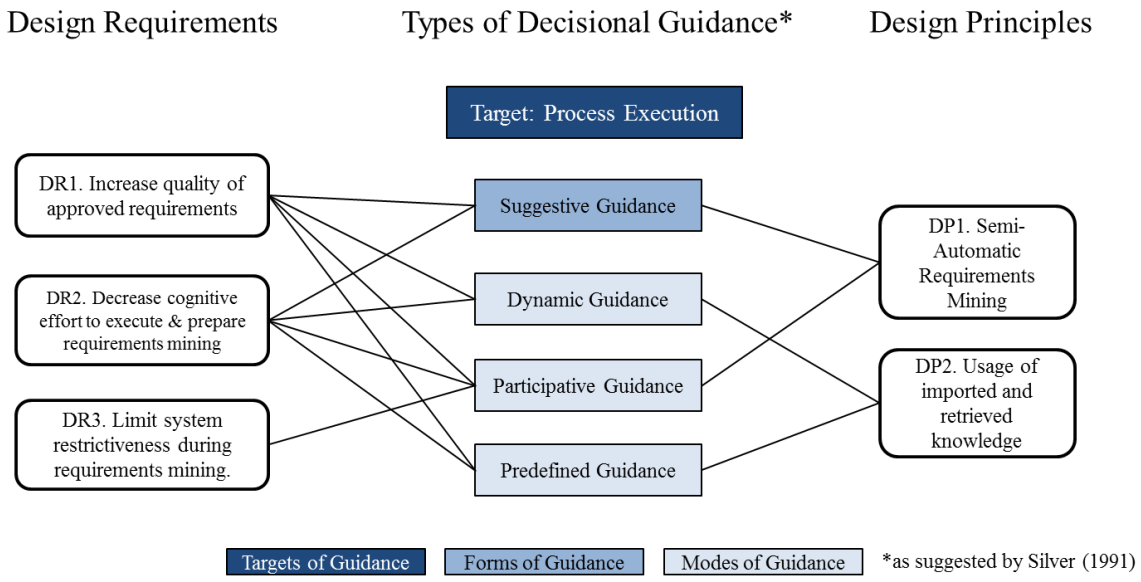

Figure 21: Deriving Design Principles from Design Requirements

5.3 Expository Instantiation

In the final step of the conceptualization, the identified design principles are mapped to design features. Design features are specific artifact capabilities to satisfy design principles, for example the algorithm chosen for automatic mining. Figure 22 summarizes the design of the artifact from design requirements via design principles to design features and illustrates the mapping between these conceptualization steps.

In allusion to the class of systems (namely RMS) and the process to be supported (requirements mining) the implemented system is referred to as “REMINER”. Similarly to former approaches (Casamayor et al. 2010; Cleland-Huang et al. 2007; Vlas and Robinson 2012), REMINER uses NLP and IR techniques to implement automatic requirements mining and additionally contains functionality to enable manual identification and classification.

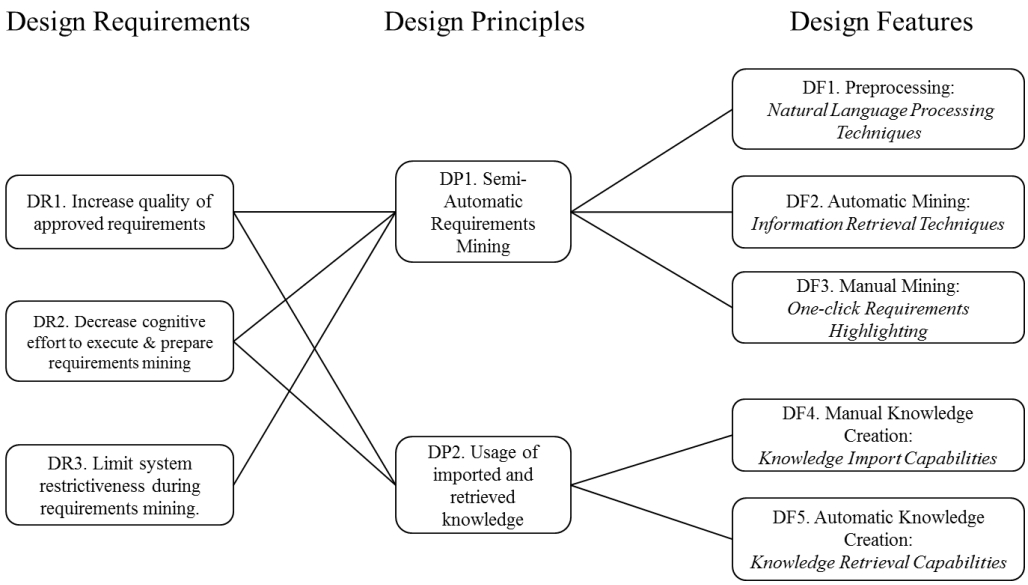

Figure 22: Mapping Design Principles to Design Requirements and Design Features

5.3.1 System Architecture

REMINER is designed as a web based client-server system implementing a three-tier architecture comprising a data tier, an application tier and a presentation tier. Figure 23 provides an overview of the system architecture. Each of the components was either implemented in the context of this thesis project or is publicly available as open source.

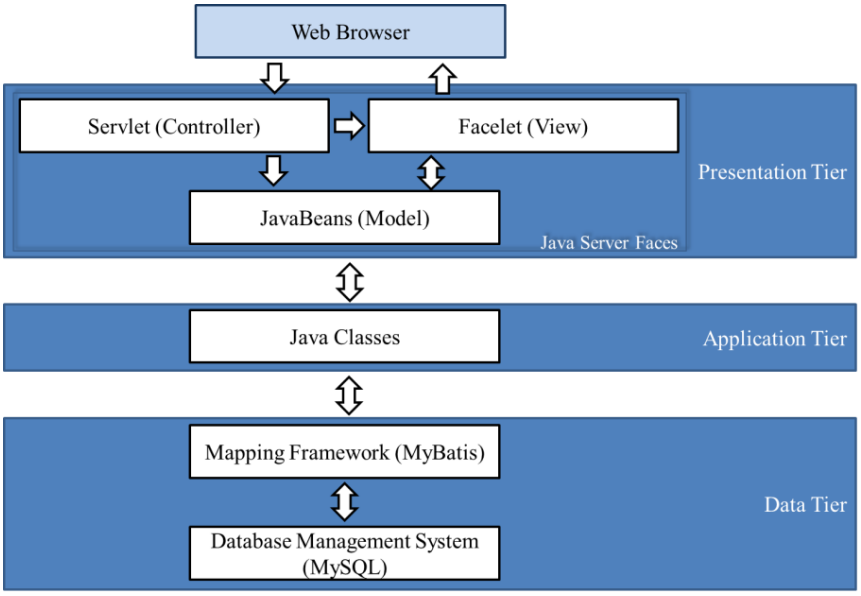

Figure 23: REMINER System Architecture

The data tier consists of two components: one to store the data and one to map data elements to the objects of the application tier. For data storage, the database management system MySQL[18] is used. MySQL was chosen due to its maturity, wide-spread usage and open source availability. For the mapping between objects and the data storage, MyBatis[19] is used. MyBatis allows encapsulating SQL[20] statements in XML[21] configuration files which drastically reduces the amount of necessary code in comparison to lower level Application Programming Interfaces (APIs) like JDBC[22] or ODBC[23].

The application tier is implemented in Java following the object-oriented programming paradigm and comprises several Java classes which control and process all system functionalities. Methods for preprocessing, as well as manual and automatic requirements mining are modularized in an Application Programming Interface (API). This API also integrates an existing NLP framework (MorphAdorner[24]) which is used during preprocessing. There are various alternative open source NLP frameworks with similar functionality such as Apache OpenNLP[25], Standford NLP[26] or Mallet[27]. MorphAdorner was chosen due to its functional completeness and comparatively elaborate documentation. The application tier strongly interacts with the JavaBeans within the presentation tier.

The presentation tier is based on Java Server Faces (JSF) and enables the user interaction and presentation of the results. JSF is a framework standard to develop graphical user interfaces for web applications. JSF was chosen as it enables a strict separation of behavior and presentation functionality, following a model-view-controller pattern. Furthermore, JSF supports the development of rich internet applications which simulate a desktop-like user experience in web applications.

[18] http://www.mysql.de/ (10.4.2013).
[19] http://mybatis.github.io/mybatis-3/ (10.4.2013).
[20] Structured Query Language
[21] Extensible Markup Language
[22] Java Database Connectivity (JDBC) is a proprietary Java-based API to access database management systems.
[23] Open Database Connectivity (ODBC) is a standard C programming language middleware API to access database management systems.
[24] http://morphadorner.northwestern.edu (10.4.2013).
[25] http://incubator.apache.org/opennlp (10.4.2013).
[26] http://nlp.stanford.edu (10.4.2013).
[27] http://mallet.cs.umass.edu (10.4.2013).

Multiple frameworks implement the JSF standard (e.g., ICEfaces[28] or RichFaces[29]). For REMINER PrimeFaces[30] was chosen due to its broad coverage of different JSF components. Choosing a JSF framework also determined the further components of the runtime environment: a web server and a servlet container which implement the JSF specifications. For this purpose, Apache Tomcat[31] was selected being a very popular open-source web server and web container which implements JSF specifications.

5.3.2 Processing

Figure 24 provides an overview of the design features implemented in REMINER and a typical process to use them[32]. In practice, variations of this process are possible, for example the provision of imported knowledge (sub-process one) could be a one-time activity just to be able to process the very first NLRR.

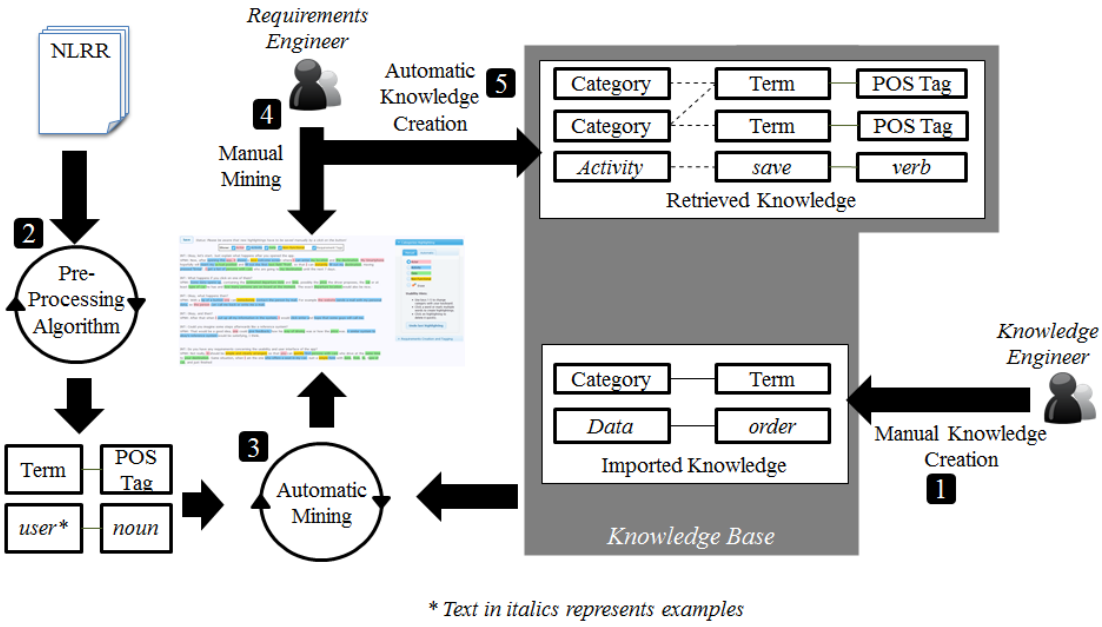

Figure 24: Requirements Mining Process Supported by REMINER

[28] http://www.icesoft.org/java/projects/ICEfaces/overview.jsf (10.4.2013).
[29] http://www.jboss.org/richfaces (10.4.2013).
[30] http://primefaces.org/ (10.4.2013).
[31] http://tomcat.apache.org/ (10.4.2013).
[32] In this figure any text in italics represents examples.

First, during *manual knowledge creation,* imported knowledge can be manually uploaded to the knowledge base by a knowledge engineer. Imported knowledge consists of terms which are associated with a specific requirements category (e.g., "credit card number" with the category "data requirement").

Second, during *preprocessing*, NLRR are transformed into single terms which serve as an input for the automatic mining algorithm. For this purpose, NLP techniques like Token Detection, Part of Speech (POS) Tagging, Stop Word Elimination and Word Lemmatizing were used. The result of this process is a set of tuples (term, POS tag), for example ("supplier","noun").

Third, *automatic mining* is enabled by an IR module that consists of various algorithms based on the vector space model as suggested by Baeza-Yates and Ribeiro-Neto (1999). The aim of the algorithms is to measure the similarity of terms extracted from the documents with terms from the knowledge base and hereby assign requirements categories. Therefore, the fundamental functioning of vector model-based search engines is adapted: The categories are indexed like documents and the terms are the search queries. Consequently, the similarity of a term to one of the categories is interpreted as the probability of the term belonging to this category. To index the categories, retrieved knowledge is used. The frequency in which a term has been assigned to a requirements category in retrieved knowledge determines its index value. Following this logic, probabilities for all terms in the NLRR are calculated based on retrieved knowledge (Step 3.a). Additionally, probabilities are calculated based on the POS of a term. For example, for a noun it is more likely to be assigned to the category "data" than to "activity". POS probability values have to be defined before running the algorithm. The POS probabilities can be defined by using the percentage of assignments in the knowledge base. For instance, if 60% of all existing assignments for the first category were verbs, the weighting factor for a verb in the first category would be 0.6 (Step 3.b). Finally, probabilities are calculated based on imported knowledge. By default, for a term which is assigned to a requirements category in imported knowledge, a probability of 1 (to belong to this category) is calculated (Step 3.c). The three probability values calculated in 3.a to 3.c are then integrated into a single, total probability value for each category (Step 3.d). The total value of the category with the

highest probability is subsequently compared to a threshold (which can be customized to a value between 0 and 1). In case the total probability exceeds the threshold, the term will be assigned to the corresponding category (and otherwise will not). Figure 25 summarizes the individual processing steps during automatic mining.

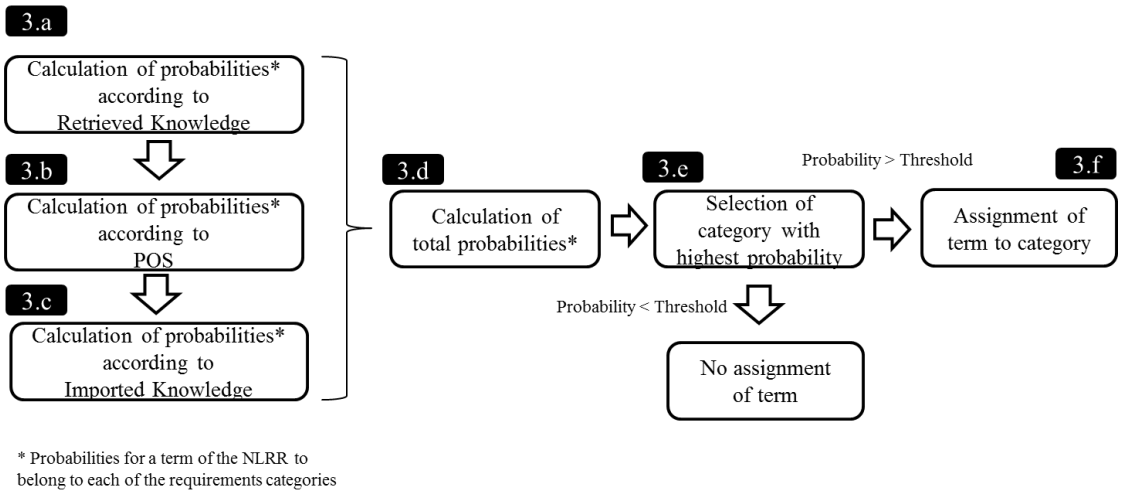

Figure 25: Individual Processing Steps During Automatic Mining

Fourth, during *manual mining*, the results of the automation process are approved. During approval, requirements that have been suggested by the algorithm can be changed or even deleted. Figure 26 shows a screenshot of the user interface for manual mining. Requirements are highlighted within NLRR like interview transcripts, workshop memos or narrative scenarios. Different requirements categories are represented by different highlighter colors, incorporating the metaphor of using text markers in physical documents. The initial list of requirements categories and thus different highlighting colors is based on the main requirements types described by (Robertson and Robertson (2006) in their Volere requirements process. Accordingly, functional requirements and non-functional requirements are differentiated. The former one is further split into the categories “data” (for text passages describing data fields or objects) and “activity” (for text passages describing a behavior of either the user or the system). Additionally, the category “actor” can be used to indicate if a requirement is rather associated to a user activity or a system activity. The text in the figure (in this case an interview transcript) contains highlightings, marking single words or entire text

passages with the highlighter color of a specific category. Users can choose a highlighter color and highlight words with a single click. Another single click deletes the highlighting again.

Figure 26: REMINER Screenshot: User interface for Manual Mining

Moreover, further requirements can be added. The finally approved requirements are then used for *automatic knowledge creation* of retrieved knowledge. Retrieved knowledge consists of terms and their associated requirements categories and POS tags. Categories are assigned through the manual mining process. For example, if one specific term or POS is often highlighted manually as one category within a domain, this category is characterized through the term or POS. As the same term could be manually assigned to different categories (e.g., by different requirements engineers), the mining algorithm can only calculate probabilities for assignments of terms to categories, based on the number of previous manual assignments. As shown in the related work chapter, most existing works concentrate on either building up requirements knowledge from NLRR (Gacitua et al. 2011; Goldin and Berry 1997; Kof 2004; Rayson et al. 2000) or use imported knowledge to support the mining itself (Ambriola and Gervasi 2006; Kiyavitskaya and Zannone 2008). In the concept presented in this thesis, these two approaches are combined in a closed loop to reduce knowledge creation efforts.

5.3.3 Artifact Demonstration

Closely aligned with the artifact's instantiation, it was demonstrated to experts to gather feedback. In the following, the procedure and results of the demonstration sessions are summarized.

In the *prototype design cycle*, the artifact was presented to requirements engineering experts to gather formative feedback towards the artifact's usefulness. To accomplish this, seven demonstration sessions were organized, involving one to four requirements engineering experts and two researchers each. In total 11 experts participated, all of them having extensive experience in requirements engineering (on average 9.7 years). The sessions lasted for about 1.5 hours and included a pre-questionnaire, the presentation of the prototype and its discussion with the experts. Each of the feedback items was traced back to the related design feature and design principle if possible. In the *final design cycle* the artifact's ease of use was assessed by usability experts, including usability consultants and professors for human computer interaction. An overall number of five sessions with 9 experts was conducted. Again, the sample consisted of experienced participants (on average 4.7 years of experience in usability engineering). The sessions were lasting for about 1.5 hours and were organized analogically to the demonstrations in the prototype cycle.

An overall number of 197 feedback items was collected during the demonstration workshops. In the following, two examples of such feedback items will be given: one from the first cycle of demonstrations (focusing on usefulness) and one from the second (focusing on ease-of use).

Concerning the functionality for requirements mining, requirements experts pointed out that for ERP[33] implementation projects it would be helpful to compare requested data fields with existing data fields of an ERP system. This feedback can be linked to the second design principle (Usage of imported and retrieved knowledge). Using both types of knowledge, general information about existing data fields of an ERP system could be uploaded as imported knowledge, while company- or project-specific ERP information could be passed from one implementation project to the next one, using retrieved knowledge. Through the provision of this knowledge, the requirements mining

[33] Enterprise Resource Planning.

algorithm automatically compares the data fields requested in a NLRR with existing data fields of an ERP system. This feedback item was also used in the preparation of the artifact's evaluation sessions. In the experiment as well as in the simulation, information about SAP data fields was used as a source for imported knowledge.

The demonstration of the artifact to usability experts led to multiple improvements of the design features for manual requirements mining. For example, the mechanism to create or delete highlightings by a single click on a word in the NLRR was suggested in one of the sessions to increase the efficiency of manual requirements mining.

In summary, the provided response in the demonstrations was primarily technology-focused and gave only few hints to the underlying design principles. However, the demonstrations provided valuable and extensive feedback to optimize specific design features of the presented artifact and hereby usefully complemented the development activities.

5.4 Principles of Implementation

Gregor and Jones (2007) describe principles of implementation as a design theory component that comprises the processes and means by which a design is introduced in a specific context. Related to REMINER, these principles could be guidelines for pilot projects within an organizational setting. As REMINER has not been implemented in according projects yet, the following principles are preliminary and subject to revision after pilot projects have been actually conducted.

Reflecting the derived design principle of semi-automatic requirements mining (DP1), the proposed interplay of automatic and manual activities should be a central component of accompanying training activities during the introduction of the system. Users of the system should be sensitized that although the RMS can propose requirements of high quality, a final manual approval of the results should be mandatory. Additionally to an improvement of the results within a specific document, manually added requirements also increase the quality of the knowledge base (through DP2). Concerning the second design principle (usage of imported and retrieved knowledge), organizations should take care to organize continuous supervisions of the knowledge base contents. As more retrieved knowledge will be automatically supplemented, a supervision of this

knowledge is mandatory to sustain a high level of quality of the knowledge base contents.

5.5 Artifact Mutability

When introduced into specific organizational contexts, for example during a pilot project at a software vendor, different adaptions of the artifact can be expected. Due to the semi-automatic supplementation of knowledge, the initially domain-independent knowledge base will be significantly changed by domain-specific knowledge. The dynamics and scope of changes depend on individual usage and access rules of the organizational context. For example, a company could use the artifact only in selected domains or companywide. Similarly, access could be restricted to domain experts or be open to all participants of a software development project. Additionally to the underlying knowledge base, it can be expected that the initial set of requirements categories will be extended or changed to cope with a specific context. For instance, the development of an application with high security standards might lead to a more detailed sub-categorization of security requirements. Similar to the principles of implementation, the expected adaptions which were described here are preliminary and subject to revision after the conduction of actual implementation projects.

5.6 Testable Hypotheses

Gregor and Jones (2007) suggest the formulation of testable propositions to be able to evaluate the presented design. In this thesis, going beyond the suggestion of Gregor and Jones (2007), specific hypotheses will be formulated. While propositions describe the relationship between general constructs, hypotheses depict relationships between specific variables (Bacharach 1989). In the following, hypotheses for the evaluation of REMINER and its underlying design will be derived. More specifically, the research model which is presented subsequently strives to measure effects of alternative combinations of the depicted design principles on multiple dependent variables. The research model is tailored to the evaluation of the final artifact version which was

conducted as an experiment. For the simulation, which was performed during the prototype design cycle, a separate model will be presented in section 6.1.2.

As introduced earlier, requirements mining productivity is conceptualized as an input-output ratio wherein the quality of the identified and classified requirements serves as the output part (numerator of the ratio) and the invested mining effort as the input part (denominator). This ratio is used as the dependent variable of this study. To evaluate the quality of automatically identified requirements, precision and recall are common measures which are similarly employed here (Casamayor et al. 2010; Cleland-Huang et al. 2007; Gacitua et al. 2011). They are calculated by comparing participants' requirements mining outputs with expert solutions (Kiyavitskaya and Zannone 2008; Vlas and Robinson 2012). Recall can be seen as a measure of completeness, comparing the number of correctly identified requirements with the total number of requirements existing in a NLRR. Precision represents a measure of correctness and is calculated as the proportion of correctly identified requirements in comparison to the number of identified requirements in a NLRR.

Variable	Explanation
Recall	$\frac{\text{Number of correctly identified requirements}}{\text{Total number of requirements}}$
Precision	$\frac{\text{Number of correctly identified requirements}}{\text{Total number of } \mathbf{identified} \text{ requirements}}$

Table 4: Measurements of Recall and Precision in the Context of RMS

The input factor requirements mining effort can be measured by the time required by a requirements engineer to conduct the mining task, i.e. transforming an unstructured input document into a set of classified requirements. As the evaluation was based on an experiment with a fixed time schedule, requirements mining effort was also fixed and only the differences in recall and precision (i.e., the quality of the identified requirements) were measured. Consequently, the evaluation measured productivity in a fixed time period, similar to the studies done by Diehl and Stroebe (1991) and Gallupe and McKeen (1990).

The conceptualization of the independent variable is directly linked to the design principles of the artifact. Both design principles can be switched on and off resulting in different RMS configurations that can be evaluated separately. For example, semi-automatic requirements mining (DP1) would be switched on, while the usage of retrieved knowledge (DP2) would be switched off. While DP1 can be switched on independently from DP2, DP2 can only be activated when DP1 is switched on. Through the separate activation of design principles, the effects of each of them can be measured individually. The resulting three RMS configurations are depicted in Table 5.

RMS configuration	Design Principle Activation	
	DP1	DP2
(1) Manual mining		
(2) Semi-automatic mining with imported knowledge	**X**	
(3) Semi-automatic mining with imported and retrieved knowledge	**X**	**X**

Table 5: RMS Configurations

Various effects of the design principles of the artifact on requirements mining productivity are expected.

5.6.1 Expected Productivity Effects of DP1 Related to Recall

Process automation is a well-known mechanism to improve productivity both for IT supported processes as well as for non-IT supported processes (Atkinson and Kuhne 2003; Jämsä-Jounela 2007). In the case of automated requirements mining, it can be expected that productivity (measured by recall in a fixed time period) will similarly improve, as an algorithm can automatically identify a large percentage of requirements without spending the requirements engineer's time during the analysis (Cleland-Huang et al. 2007; Kiyavitskaya and Zannone 2008; Vlas and Robinson 2012).

Investigating this assumption from a theoretical point of view, it is worthwhile revisiting the analogy to decision making, which was introduced in the conceptualization. Automatically proposed requirements represent one possible form of suggestive guidance. In their experimental study, Parikh et al. (2001) showed that participants with DG outperformed participants without DG concerning the achieved

decision quality and efficiency. Furthermore, suggestive guidance resulted in an increase of decision quality and efficiency in comparison to informative guidance. Being a specific instance of a decision making process, in requirements mining the application of automation mechanisms should similarly result in improvements of requirements quality and efficiency in comparison to manual requirements mining.

Investigating the assumption from a process point of view, the recall using a semi-automatic RMS can be seen as a sum of the automatism's initial recall and the recall resulting from subsequent manual adaptions and extensions. These subsequent manual activities are comparable to using a purely manual RMS: a requirements engineer needs to read a natural language text document, mark passages containing requirements and assign requirements categories to them. Therefore, no significant recall difference between semi-automatic and manual RMS is expected from these manual activities. In contrast, the initial recall provided by the automatism will remain and can be expected to have a significant effect. Consequently, the following hypothesis is derived:

> *H1: In a fixed time period, the use of RMS that support semi-automatic requirements mining with imported knowledge will result in higher recall than the use of RMS that only support manual requirements mining.*

5.6.2 Expected Productivity Effects of DP2 Related to Recall

As described above, automated requirements mining requires a knowledge base containing requirements and a categorization of these elements. Each automatically identified requirement can be traced back to a specific entry in this knowledge base. Accordingly, a more elaborate and extensive knowledge base can generally be expected to result in a higher percentage of identified requirements and therefore a reduction of manual efforts (Cleland-Huang et al. 2007). In their empirical study on the effects of DG, Parikh et al. (2001) observed similar effects concerning the usage of dynamic guidance. Dynamic guidance which is based on knowledge that is dynamically created through the usage process, outperformed predefined guidance concerning decision quality and decision efficiency.

Additionally to the size of the knowledge base, the domain-specificity of the knowledge plays an important role in the requirements mining process (Casamayor et al. 2010). Generally, a higher degree of domain-specificity is expected to deliver better mining results (Lemaigre et al. 2008), for example by including domain-specific requirements (like "physician" or "nurse") additionally to domain-independent ones (like "manager" or "worker"). As depicted earlier, two sources of knowledge to fill the knowledge base are proposed. Additionally to manually imported knowledge, which is commonly used in existing RMS (Kiyavitskaya and Zannone 2008; Vlas and Robinson 2012), the content of the knowledge base can be extended by automatically retrieved knowledge originating from documents that have been processed before. As described in the conceptualization of DP2, this should increase the size and domain-specificity of the knowledge base. Therefore the following hypothesis is derived:

> *H2: In a fixed time period, the use of RMS that support semi-automatic requirements mining with imported and retrieved knowledge will result in higher recall than the use of RMS that only support semi-automatic requirements mining with imported knowledge.*

5.6.3 Expected Productivity Effects of DP1 and DP2 Related to Precision

As described earlier, both recall and precision determine requirements quality and therefore are of utmost importance for the overall requirements mining process. However, in automated requirements mining from NLRR, recall is significantly more important than precision, as it is a much simpler activity for a requirements engineer to evaluate a set of candidate requirements and reject the unwanted ones than it is to browse through an entire document looking for entirely missed ones (Cleland-Huang et al. 2007). The same argument is used by Berry et al. (2012) who state that requirements engineering tools that treat NLRR "should be tuned to favor recall over precision because errors of commission are generally easier to correct than errors of omission" (Berry et al. 2012, p.213). Because of that, the design principles of the artifact primarily address an improvement of the recall rate and do not target precision improvements.

Moreover, while the recall rate is predominantly determined by the automatism's ability to find as many relevant words and text passages as possible, the precision rate is strongly linked to the quality of the judgments following the identification of a word/text passage. A significant precision improvement could therefore only be realized if the algorithm provided better judgments than a human. However, as the requirements proposed by the RMS are based on knowledge created by humans, this cannot be expected.

Therefore the following hypothesis is derived:

> *H3: In a fixed time period, the use of manual RMS, RMS that support semi-automatic requirements mining with imported knowledge and RMS that support semi-automatic requirements mining with imported and retrieved knowledge does not result in significant differences in precision.*

Figure 27 summarizes the hypotheses in a comprehensive research model.

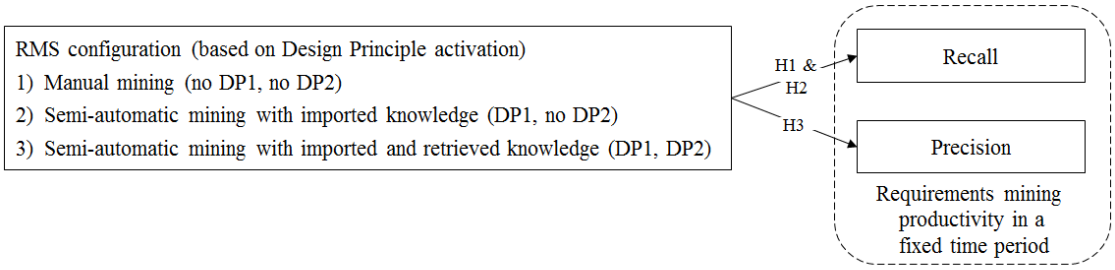

Figure 27: Research Model for Ex-Post Evaluation

5.7 Summary

In this chapter a design theory for RMS was presented, along the design theory components proposed by Gregor and Jones (2007). Starting with the purpose and scope of the artifact, distinct design requirements were derived based on decision making theory. The decision making process provides a significant analogy to requirements mining which has been exploited for the justification of the proposed design. In the subsequent conceptualization step, design principles were mapped to the requirements,

again utilizing justificatory knowledge from decision making theory. Then, design principles were mapped to actual design features of an artifact which has been instantiated as part of this thesis project. Subsequently, principles of implementation and the artifact's mutability were described. Finally, the proposed research model consisting of three hypotheses was conceptualized. Table 6 summarizes the contents of the derived theory and relates it to the original design theory components of Gregor and Jones (2007).

Design Theory Component[34]		**Reference to this component in the presented theory**
(1)	Purpose and scope	The presented design theory aims to give explicit prescriptions about how to develop systems that support requirements mining from NLRR to improve requirements mining productivity. The proposed class of systems might be applied to a wide range of NLRR and in the context of various software and requirements engineering methodologies. According to this purpose and scope, design requirements have been derived.
(2)	Constructs	Specific design features for RMS have been presented.
(3)	Principles of form and function	Design principles to support the requirements mining process as well as knowledge creation and maintenance processes have been derived.
(4)	Artifact mutability	Contents of the knowledge base used for automatic mining as well as the underlying scheme for requirements categorization depend on the context of use and can therefore be adapted.
(5)	Testable propositions	Three hypotheses were formulated to test the effects of different configurations of design principles on requirements mining recall and precision.
(6)	Justificatory knowledge	Design requirements and design principles were derived from decision making theory and general requirements mining knowledge.
(7)	Principles of implementation	Two principles of implementation were formulated: Mandatory final approval of the results and the organization of continuous supervisions of the knowledge base contents.
(8)	Expository instantiation	REMINER, an expository instantiation of an RMS has been presented, including a depiction of its system architecture and a typical process to conduct system-supported requirements mining.

Table 6: Components of a Design Theory for RMS

[34] According to Gregor and Jones (2007).

6 Artifact Evaluation

Pries-Heje et al. (2008) distinguish two types of DSR evaluation approaches, depending on the time they are conducted: *ex ante* and *ex post* evaluations. While ex ante evaluations are performed before the system is implemented, ex post evaluations take place after the system construction (Pries-Heje et al. 2008). However, in DSR projects consisting of multiple iteration cycles, a third form of evaluation takes place which will be referred to as *interim evaluation*. This type of evaluation is conducted on the basis of an artifact prototype version. Although a prototype may already be an implemented artifact, it does not represent the final design product (and therefore would not qualify for an ex post evaluation).

The design theory presented in chapter 5 represents the result of the final design cycle. However, as depicted in section 4.3, the artifact in this thesis project was designed in two iteration cycles, with two separate evaluations. Therefore, in the following, the methodology and results of both evaluations will be presented. The former represents the assessment of the artifact prototype version (an interim evaluation), while the latter represents the artifact's ex post evaluation.

6.1 Interim Evaluation[35]

The interim evaluation was carried out to investigate the interplay of the preliminary design principles[36]. More specifically, the effects of different amounts and types of knowledge in corporation with the usage of automatic requirements mining were investigated.

The decision to perform this evaluation in form of a simulation was based on two factors: First, a simulation allows precise measurements of effects in a laboratory environment, whilst controlling other factors which are not of interest. Second, in comparison to evaluation approaches which rely on human interaction, simulations provide the flexibility to explore different factors (in this case different amounts and types of knowledge) with comparatively low effort.

[35] Parts of this section are based on Meth et al. (2013b).

[36] The design principles presented in 5.2.2 represent the final conceptualization as a result of the final design cycle. In the preceding prototype design cycle, preliminary design principles were derived.

Prior to the description of the evaluation results, the simulation setup will be outlined in the following section, including the utilized evaluation dataset, the underlying research model for the simulation and the procedure to conduct the simulation.

6.1.1 Dataset

The simulation was based on a dataset[37] which is made up of multiple natural language requirements documents and the knowledge to be used for automatic requirements mining. Furthermore, a gold standard has been used which is the expert solution to assess the results of automatic requirements mining.

The natural language requirements documents consisted of previously conducted interview transcripts. These interviews were carried out with 12 potential end-users to gather their requirements for two projects. Both projects intend to implement smartphone apps associated with the "travel management" domain. The first application is a train reservation app which allows users to make reservations for regional and national trains, while the second application is a car sharing app which allows users to get in touch with other people for the purpose of joint car drives to similar destinations. To demonstrate the commonalities of these two apps and how they are both associated to the traveling domain, it is worthwhile to investigate corresponding example websites for train reservations[38] and car sharing[39]. The main functionality of both websites is very similar: they offer functionality to enter information about the origin and destination of the travel, the start date and time and whether a direct connection is required. However, beyond these domain-specific similarities (which would also be typical for a flight reservation website as another example for a traveling app), there are also differences. For example, on the train reservation website, different types of rail cards can be selected and the option to use a sleeper train can be chosen. Similarly, on the car sharing website features to select "women-only lifts" or "smoking allowed lifts" are provided.

[37] Appendix B and C of this thesis contains the interview transcripts and imported knowledge of the dataset which was used in the simulation. Parts of this dataset were also used for the experiment evaluation.

[38] http://www.nationalrail.co.uk/ (28.01.2013).

[39] http://www.carpooling.co.uk (28.01.2013).

In the interviews, participants should verbalize their requirements using scenario methodology, describing a typical process of using the smartphone app. This description should comprise each single interaction step and include the data to be exchanged, the activities to be performed and the non-functional aspects which are of importance (e.g., usability concerns). Each interview lasted 5-10 minutes and transcripts of about one page per interview were created. The participants were students, who had no specific requirements engineering knowledge. They were on average 23.7 years old, six of them male and six female. From the 12 conducted interviews, 9 were finally selected for the simulation, four of them referring to a train reservation project and five to a car sharing project.

The knowledge used for the automation algorithm consisted of both imported and retrieved knowledge. Imported knowledge was uploaded from different data sources, depending on the requirements category: for the role category, a list of pronouns from the Oxford Dictionary was extracted. For the activity category, a list of action verbs from Hart (2004) was used. For the data category the master data of a SAP Travel Management application was used (SAP AG 2012). For the non-functional category, an extract of usability goals and design behaviors from Sharp et al. (2007) was imported. Two different sets of retrieved knowledge were applied: one set was retrieved from texts about the train reservation app and one set from texts about the car sharing app.

To derive a gold standard, each of the 9 interviews was manually highlighted by three requirements engineering experts. After resolving conflicts and contradictions, the final agreed-upon solution of the experts was taken as the gold standard.

6.1.2 Research Model for Interim Evaluation

As explained earlier, the research model, which was derived in section 5.6 is tailored to the evaluation of the final artifact version which was conducted as an experiment. For the simulation described here, a separate model has been developed and will be presented in the following.

The goal of this evaluation was to investigate how the amount and type of knowledge which is used for automated requirements mining affects the quality of the results. As described earlier, to evaluate the *requirements mining quality*, recall is a common

measure (Casamayor et al. 2010; Cleland-Huang et al. 2007; Gacitua et al. 2011) which was equally applied in this evaluation. It was calculated by comparing the automatism's outputs with the gold standard introduced in the last section. In the simulation conducted here, requirements mining efficiency was not of interest, as the evaluation focused on the outcomes of automatic requirements mining and therefore did not involve human interaction through requirements engineers.

The independent variable *amount of knowledge* is operationalized by the number of documents used to build up the knowledge base. The study thereby simulates how knowledge would probably be extended in practice: starting from an initial, imported amount of knowledge, the knowledge base would be gradually augmented through retrieved knowledge from already processed NLRR.

The type of knowledge is represented by two independent variables: Origin and project-specificity of knowledge. *Origin of knowledge* is operationalized by using different contents within the knowledge base: only imported knowledge, only retrieved knowledge, or a combination of both. *Project-specificity of knowledge* is operationalized by using retrieved knowledge for either the same or a different project. Both projects, however, belong to the same domain as projects from different domains may restrict reuse of knowledge to specific types of requirements (e.g. non-functional requirements). The resulting evaluation model is depicted in Figure 28.

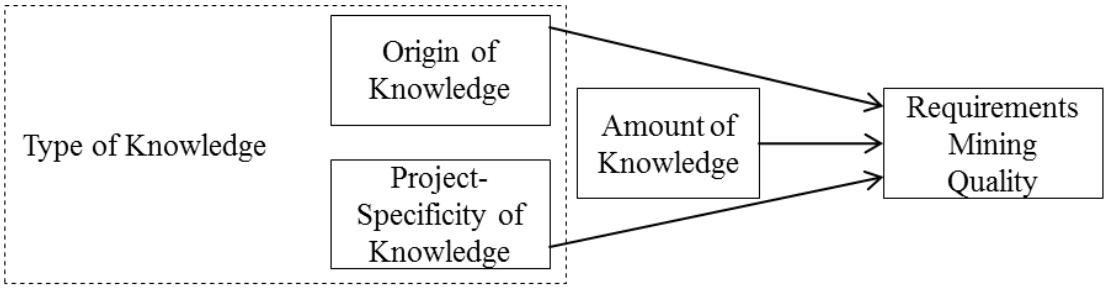

Figure 28: Research Model for Interim Evaluation

6.1.3 Evaluation Procedure

Based on the introduced dataset and evaluation model, two simulations were performed. In both simulations, requirements were automatically elicited from four exemplary interview transcripts. The resulting recall rates were then averaged to a single result. Subsequently, the simulations were repeated with a different amount of retrieved knowledge. For each result, the recall rate was examined by comparing the results of the automatism to the gold standard.

The first series of simulations focused on the effects of different origins of knowledge on requirements mining quality. Additionally to the origin of knowledge, the amount of retrieved knowledge was varied by using a different number of texts to populate the knowledge base with retrieved knowledge. This resulted in a series of 11 different simulation runs. The first run only used imported knowledge, the following five runs only used retrieved knowledge (for 0-4 texts) and the final five runs a combination of both (for 0-4 texts). The analyzed natural language documents as well as the retrieved knowledge for this series of simulation originated from the project for the car sharing application, resulting in a constantly high project-specificity of the knowledge. Table 7 summarizes the performed simulation runs.

Simulation Run #	Origin of Knowledge	Number of texts[40]
1	Imported Knowledge	-
2 to 6	Retrieved Knowledge	0 to 4
7 to 11	Imported & Retrieved Knowledge	0 to 4

Table 7: Simulation Runs for Variable Origin of Knowledge

The second series of simulations (Table 8) focused on the effects of project-specificity of knowledge on requirements mining quality. For the project-specific simulation runs, only retrieved knowledge from the car sharing project was taken. For the project-independent runs, only retrieved knowledge from the train reservation project was taken. The interviews to be analyzed were related to the car sharing project. Similar to the first series, the amount of retrieved knowledge was additionally varied. This resulted in a series of 10 different simulation runs. The first five runs simulated a project-

[40] Only related to Retrieved Knowledge.

specific knowledge base (for 0-4 texts) the next five runs simulated a knowledge base with knowledge from a different project (for 0-4 texts). In this series, the origin of knowledge was kept constant, as only retrieved knowledge was used.

Simulation Run #	Project-Specificity	Number of texts[3]
1 to 5	Project-Specific Knowledge	0 to 4
6 to 10	Project-Independent Knowledge	0 to 4

Table 8: Simulation Runs for Variable Project-Specificity of Knowledge

6.1.4 Evaluation Results

Figure 29 depicts the results of the first series of simulation runs which focused on the effects of different origins of knowledge on mining quality. As expected, the results suggest that a positive correlation between the number of texts used for the creation of the retrieved knowledge and the resulting recall rate can be assumed. In addition, it can be observed that for new projects, which have not identified requirements from NLRR yet, an initial amount of imported knowledge is necessary to achieve a relevant recall rate. However, it can be seen that in the conducted simulation the recall from retrieved knowledge approximately equaled the recall from imported knowledge after three documents had been analyzed and outperformed it for more than three documents. Additionally, it is interesting to notice, that imported knowledge in the simulation seemed to have no further effect if more than three documents had been used for retrieved knowledge: The recall rate for the combination of imported and retrieved knowledge approximately equals the recall rate for retrieved knowledge if more than three documents had been used.

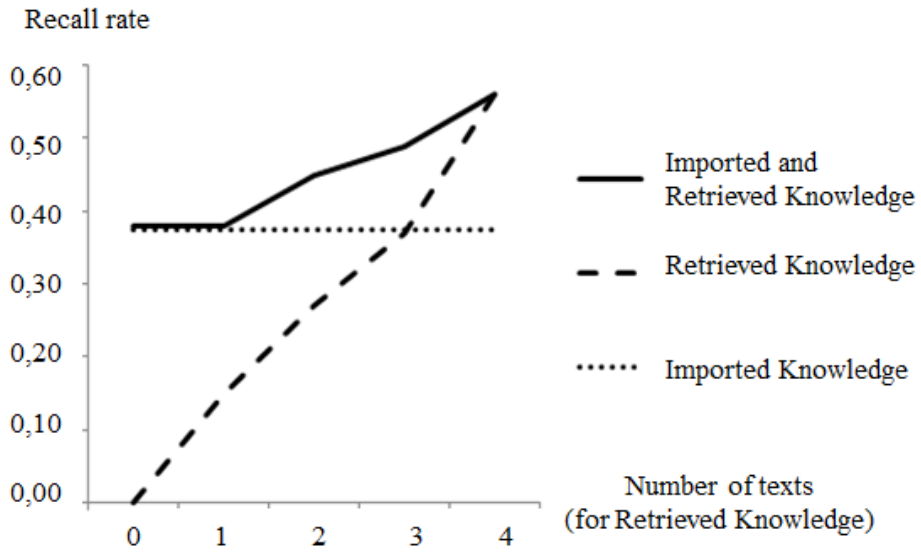

Figure 29: Effects of Origin of Knowledge on Requirements Mining Quality

Figure 30 depicts the results of the second series of simulation runs which focused on the effects of project-specificity of knowledge on mining quality.

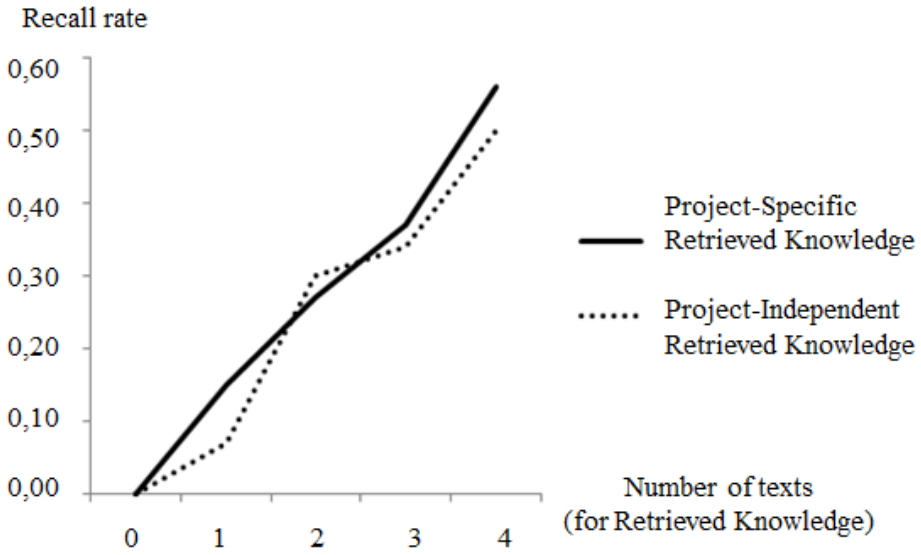

Figure 30: Effects of Project-Specificity of Knowledge on Req. Mining Quality

Interestingly, in the conducted simulation series, project-specificity of knowledge had an ambiguous effect. Recall rates again seem to develop in positive correlation with the

amount of knowledge, but no clear difference could be observed concerning project-specificity itself. Even though both projects are related to the same domain (travel management), this was an unexpected result, as the two applications which the interviews based on (a train reservation and a car sharing application) provided significant differences. These observations allow the interpretation that automated requirements mining can significantly benefit from an exchange of requirements knowledge across projects within the same domain. In section 7.1.1, the results of the simulation will be discussed in more detail.

6.2 Ex-Post Evaluation [41]

The ex-post evaluation was carried out to test the effect of the two (final) design principles (DP1, DP2) on requirements mining productivity. As described in the related work chapter, previous research on RMS evaluations focused on simulations, comparing the results of the corresponding systems with a previously defined gold standard. Although simulations allow precise measurements of dependent variables in a controlled setting, they do not incorporate human interaction. RMS are supposed to be used by requirements engineers, who should be consequently involved in the evaluation of the systems to be able to compare the outcomes of system-supported requirements mining with the as-is situation of manual discovery. Therefore, for the ex-post evaluation of REMINER, an experiment evaluation as suggested by Hevner and Chatterjee (2010) was conducted. By using a laboratory experiment, design principles can be accurately adjusted and their impacts on requirements mining productivity can be measured while controlling for potential influential factors (e.g., requirements mining knowledge, motivation). The results achieved through system-supported requirements mining can then be compared to manual discovery, addressing the research gap described above.

The ex-post evaluation was based on the hypotheses and research model derived in the previous chapter. Following these hypotheses, the identified design principles (DP1 and DP2) were expected to improve requirements mining productivity. The actual outcomes

[41] Parts of this section have been published in Meth et al. (2012a).

of the experiment will be specified in the following, after a description of the evaluation methodology.

6.2.1 Evaluation Methodology

The overall experiment consisted of a laboratory experiment and a field experiment. First, the artifact was evaluated in a laboratory setting with student participants. By using student participants, a relatively large sample size can be obtained with reasonable efforts and adequate statistical power can be achieved (Gallupe and McKeen 1990). Second, to evaluate the generalizability of findings from the student participants, the same experiment was carried out with a small sample of requirements engineers in a field setting. By comparing the behavioral patterns of the two groups of participants, the external validity of the results from the laboratory setting can be evaluated. It should be noted that it was not intended to merge the two samples to test the hypotheses, but only to use the results of the small sample of requirements engineers as an examination of the student sample's external validity. All conclusions from the experiment should be reliably drawn from the relatively large sample of students.

A single factor within-subject design was used for both the laboratory experiment and the field experiment to increase statistical power for each experimental setting and reduce error variance introduced by individual differences (Hill and Lewicki 2006). The within-subject factor is the RMS configuration. This factor has three levels: manual requirements mining, semi-automatic requirements mining with imported knowledge, and semi-automatic requirements mining with imported and retrieved knowledge.

6.2.1.1 Pilot Test

A pilot test was conducted to estimate the necessary sample size and appropriate length of the interview transcripts used in the experimental tasks. The same single factor within-subject design was applied in the pilot test as in the main experiment, and three graduate students participated in the pilot test. The results indicated that the lowest correlation among the repeated measures was 0.35. Calculated with G*Power 3 (Faul et al. 2007) to detect a medium effect (f= 0.25) at the significance level of 0.05 with a sufficient statistical power (about 0.80) (Cohen 1988) the sample size should be at least

35. Thus, the *sample size* for the laboratory experiment was set to be 40 to detect a medium effect on recall and on precision.

In the pilot test, within the experimental time for each task (5 minutes), the maximum amount of words that the participants processed was 247, 277, and 328 for manual requirements mining, semi-automatic requirements mining with imported knowledge, and semi-automatic requirements mining with imported and retrieved knowledge respectively. Accordingly, the *length of the interview transcripts* used in the main experiment was set to be 325 words. With this length, most of the participants are expected not to be able to completely process all the text within the experimental time, but they can achieve their optimal recall and precision while working at their normal pace. A very small number of participants might be extraordinarily fast in requirements mining and be able to complete the first round of requirements mining within the experimental time, allowing them to further improve recall and precision in the remaining time by checking the first round results. The interview transcripts were not set up to be of a length that no participant could possibly complete the first round of requirements mining because the impact of the automatically mined requirements on the achieved recall and precision should be limited. Participants should be able to read and check the automatically mined requirements within the task time which aligns to the application situation in practice.

6.2.1.2 Participants

According to the sample size calculation, 40 participants were recruited for the laboratory experiment. The participants were graduate students enrolled in a master level IS course in a public university at an average age of 25.4 years (SD=2.07).

Age	Gender	Major	Computer Experience	Experience in Requirements Mining
25.4 years (avg.)	Male: 32 students Female: 8 students	Master of Business Informatics: 36 students Master of Management: 4 students	4.75 (avg. of max. 5)	1.79 (avg. of max. 5)

Table 9: Participants' Descriptive Data (Average Values)

Thirty-two of the participants were male and eight of them were female. Most of the students (36 of 40) are pursuing a master of business informatics, while four students are enrolled in a master of management program. The participants have a comparatively extensive general computer experience (on average 4.75 points on a five point Likert scale) and low requirements mining experience (1.78 point on a five point Likert scale). Participants were evenly assigned to six time slots on three experimental days, with 6 or 7 participants per time slot.

6.2.1.3 Experimental Procedure

The experiment was carried out in a multimedia classroom at the university. A lecturer of the IS course introduced the experiment as an exercise for a course-related assignment with the objectives of understanding different requirements categories relevant for Business Intelligence and learning how to use a web application to perform requirements mining from text documents. No participant had access to the RMS before the experiment and all participants were unaware of the purpose of the experiment.

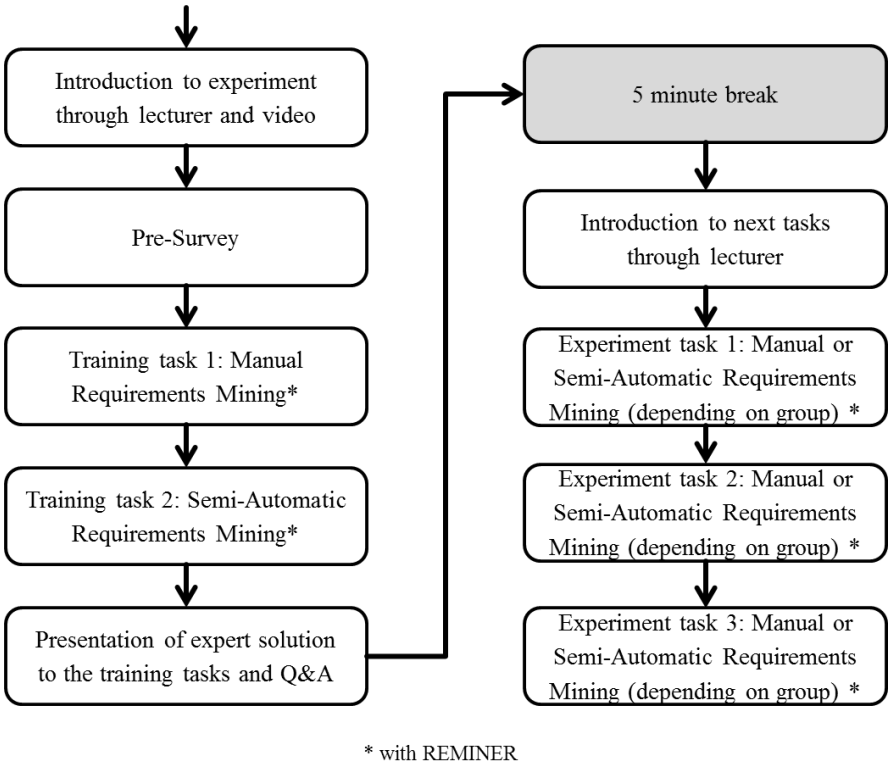

Figure 31: Experimental Procedure

To teach how to perform requirements mining and how to use the web application for this purpose, the lecturer presented a tutorial video to the participants. Then, the participants were asked to fill in a brief questionnaire about their demographic information, computer experience and requirements mining experience. Next, the lecturer guided the participants through a training session to make them familiar with requirements mining. The participants were asked to perform requirements mining using an interview transcript about requirements of a train reservation application for smartphones. The transcript was chosen from the series of transcribed interviews, which were described in 6.1.1. In the first five minutes, participants conducted requirements mining manually. In the next five minutes, they performed requirements mining within the same transcripts again but with a few automatically mined requirements at the beginning. Afterwards, the lecturer presented the expert requirements mining results for the transcript and answered any question raised by the participants. Then the participants were allowed a five-minute break.

After the break, the lecturer asked the participants to practice their requirements mining skills with a different set of interview transcripts which contained three transcripts about requirements of a car sharing application for smartphones. Again, these transcripts were chosen from the series of transcribed interviews described in 6.1.1. By design, requirements mining within the three interview transcripts was supported with three different RMS configurations. To compensate for learning and fatigue effects in the within-subject design, the presentation order of the three RMS configurations was fully counterbalanced across the participants, yielding a total of six orders. The participants were randomly assigned into one of the six orders of RMS configurations. For each interview transcript, the participants were given five minutes to perform the requirements mining. Then they were instructed to switch to the next interview transcript and start requirements mining on it.

In the field experiment, participants were five requirements engineers (targeted users of the RMS) recruited from a large high-tech company. The practitioner sample consisted of three males and two females at an average age of 34.8 (SD=3.56) and an average experience of 5.0 years (SD=5.83) and 3.6 years (SD= 1.14) in requirements engineering and requirements mining respectively. The participants in the field

experiment followed similar experimental procedures as the ones in the laboratory experiment, with a few necessary modifications. Firstly, the participants were randomly assigned into one of the orders of the RMS configurations. Since only five participants were involved in the field experiment and each participant got a different order of the RMS configurations through randomization, five among the six orders of the RMS configurations were covered in the field experiment. Secondly, the purpose of the study was introduced as "to get experts' opinions on future design of RMS". No participant had access to the RMS before the experimental tasks and the participants were unaware of the real purpose of the experiment. The participants were told to work at their normal working pace in different tasks. All the other procedures in the field experiment were the same as the ones in the laboratory experiment.

6.2.1.4 Experimental Tasks and Materials

To set up the experimental tasks, the following three steps were performed: choose a knowledge domain, select interview transcripts, and set up the semi-automatic requirements mining.

A knowledge domain determines the area of knowledge that participants and the semi-automatic RMS rely on in order to identify and classify requirements. Some knowledge domains require specialized knowledge and expertise (e.g., computer aided design), while others only require routine knowledge that can be easily acquired in ordinary life (e.g., online shopping). Similarly to the simulation, in the experiment, "travel management" was chosen as the knowledge domain, since this domain does not require specialized knowledge, and the student participants would be able to identify and classify requirements by their routine knowledge.

In the experiment, participants were provided a training session to get used to the RMS before the experimental tasks. To reduce the practice effect, interview transcripts on different applications were specified for the training and for the experimental tasks respectively. As already mentioned, transcripts were selected from the interviews described in 6.1.1. These interviews included requirements descriptions for two smartphone applications, one for car sharing and one for train reservations. In the training, a short transcript about requirements of the train reservation application was

provided (238 words). In the experimental tasks, transcripts about requirements of the car sharing application were used. For the transcripts used in the experimental tasks, the length, readability, and the distribution of requirements were controlled on. Each transcript was edited to contain 325 words without sacrificing the integrity and meaningfulness of the interview content. Examined by the Flesch-Kincaid score, the three transcripts have similar and high readability (M=75.1, SD=3.50) which indicates that all the transcripts were highly readable for university students at master level (Kincaid et al. 1975). To examine the distribution of the requirements in the transcripts, two requirements mining experts analyzed the transcripts independently. Their requirements mining results were compared and any inconsistency was discussed and resolved. The converged expert solutions showed that the three transcripts contained a relatively equal amount of requirements (M=70.3, SD=2.09) and that the requirements were evenly distributed across the complete text of each transcript.

Finally, semi-automatic requirements mining was set up within the "travel management" requirements domain. Participants were instructed to perform requirements mining within interview transcripts using three different RMS configurations: 1) Manual mining 2) Semi-automatic mining with imported knowledge and 3) Semi-automatic mining with imported and retrieved knowledge. The first configuration was based on unprocessed interview transcripts, no automatically mined requirements. In contrast, the second and third configurations were based on transcripts which already contained automatically generated requirements. Due to the additional retrieved knowledge utilized in configuration 3, the according setup resulted in more proposed requirements than configuration 2 which is depicted in Figures 32 and 33.

To prepare semi-automatic requirements mining, the same imported knowledge which was used in the interim evaluation, was employed in this evaluation. After knowledge import, automatic requirements mining was performed to generate requirements in the selected interview transcripts for the experimental task. The resulting average recall and precision was 54.0% (SD=9.4%) and 79.0% (SD=6.9%) respectively.

Figure 32: Requirements Document After Automatic Processing in Configuration 2

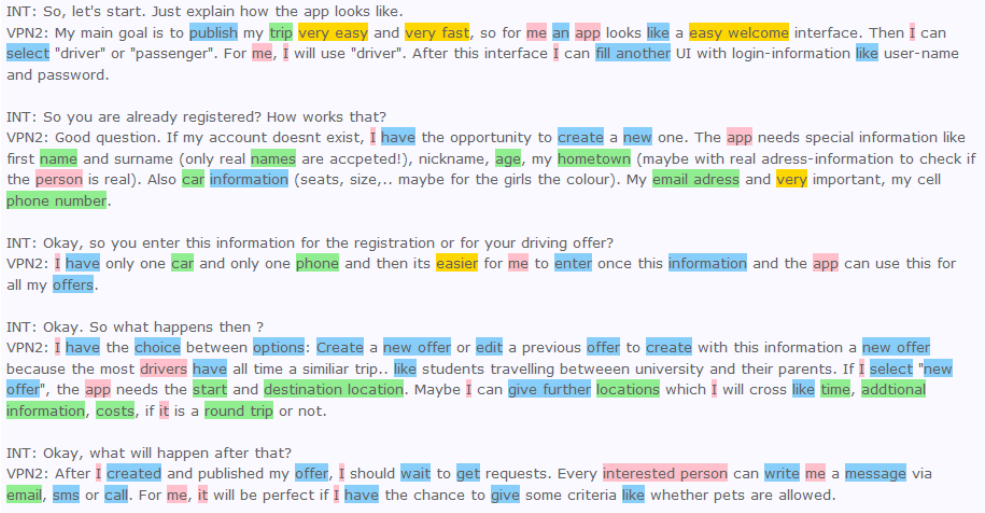

Figure 33: Requirements Document After Automatic Processing in Configuration 3

In contrast to imported knowledge, retrieved knowledge does not require additional efforts to be acquired. Retrieved knowledge for a requirements domain is acquired automatically by the RMS when users perform requirements mining on any text document within the specific requirements domain. To acquire the retrieved knowledge for the “travel management” requirements domain, one requirements mining expert performed requirements mining with the RMS on a set of interview transcripts about the train reservation application. The choice of the transcripts ensured that knowledge was

retrieved within the same knowledge domain (travel management), but for an application different from the car sharing application used in the experimental tasks which made the knowledge retrieving process closely aligned to the real situation in practice. With imported and retrieved knowledge, an average recall of 75.0% (SD=4.2%) and an average precision of 75.0% (SD=4.0%) was achieved after running the automatic requirements mining on the three interview transcripts for the car sharing application. In the experimental tasks, the order of the three interview transcripts was randomized across the participants.

6.2.1.5 Measurements of the Dependent Variables

As illustrated earlier in the description of the research model, requirements mining productivity was measured by the achieved quality within a fixed time frame. Participants' requirements mining quality was evaluated with two variables: recall and precision. Following the approach by Kiyavitskaya and Zannone (2008) and Vlas and Robinson (2012), recall and precision were obtained by comparing participants' requirements mining outputs with the expert solutions. Within a text document, if a participant identified a text segment as one requirement, no matter in which requirements category the participant classified this requirement, it was counted as one "identified requirement." If the participant identified a text segment as one requirement and assigned it to a requirements category in the same way as shown in the expert solution, this requirement was counted as one "correctly identified requirement." As shown in Table 10, a participant's achieved recall for a text document was calculated as a ratio of the number of correctly identified requirements by the participant to the total number of requirements contained in this text document according to the expert solution. A participant's achieved precision for a text document was calculated as a ratio of the number of correctly identified requirements by the participant to the total number of identified requirements by the participant. To reduce the bias introduced by document analysts, two requirements mining experts analyzed 10% of the participants' outputs independently and achieved an inter-rater reliability of 98.97%; afterwards, the two experts spilt the remaining outputs and analyzed them separately.

Variable	Explanation
Recall	$\frac{\text{Number of correctly identified requirements by the participant}}{\text{Total number of requirements in the expert solution}}$
Precision	$\frac{\text{Number of correctly identified requirements by the participant}}{\text{Total number of } \textbf{identified} \text{ requirements } \textbf{by the participant}}$

Table 10: Measurements of the Dependent Variables

6.2.2 Data Analysis and Results

All the data analysis was conducted using SPSS for Windows Version 16.0. First, the data obtained from the laboratory experiment was examined and used to test the hypotheses. Then, as an estimation of the external validity of the laboratory experiment, the data from the field experiment was analyzed and compared with the data from the laboratory experiment.

6.2.2.1 Preliminary Analysis

Table 11 presents the means and standard deviations of the dependent variables in different experimental conditions for the laboratory experiment and the field experiment respectively. For manual requirements mining, the practitioner sample appeared to achieve a relatively lower recall than the student sample. The reasons could be that the students were more motivated and concentrated during the experimental task than the practitioners, or the small sample of practitioners might not be evenly distributed on both sides of the true value of the population mean. In hypotheses testing, only the data from the laboratory experiment was used to achieve a sufficient power and get reliable conclusions.

As explained in the research model, requirements mining recall and precision are conceptually independent variables. The hypotheses predict that the RMS configurations exert effects on recall and precision in different directions. Thus hypotheses on recall and precision should be tested separately with univariate repeated measures of analysis of variance (RMANOVA) (Huberty and Morris 1989).

	Manual		**Semi-automatic with imported knowledge**		**Semi-automatic with imported and retrieved knowledge**	
	Mean	***SD***	**Mean**	***SD***	**Mean**	***SD***
Lab experiment (student participants, *N*=40)						
Recall	50.7%	12.0%	69.8%	9.8%	79.5%	8.0%
Precision	71.0%	8.5%	72.0%	6.7%	73.2%	6.5%
Field experiment (practitioner participants, *N*=5)						
Recall	37.6%	12.9%	68.6%	6.0%	77.8%	3.9%
Precision	70.1%	14.5%	72.7%	3.5%	68.5%	5.3%

Table 11: Recall and Precision for Different RMS Configurations

6.2.2.2 Hypotheses Testing

With the data from the laboratory experiment, RMANOVA was performed to examine the impacts of the design principles on requirements mining recall and on precision respectively.

As shown in Table 12, participants' recall was significantly influenced by the RMS configurations at the significance level of 0.05. To test hypothesis 1 and hypothesis 2, pairwise comparisons were performed on the main effects of RMS configurations. A Bonferroni correction was applied to control on the family-wise error rate (Vasey and Thayer 1987). The multiple comparisons results are shown in Table 13. All the pairwise comparisons were significant at the level of 0.05: participants using semi-automatic requirements mining with imported knowledge achieved significantly higher recall than using manual requirements mining, and using semi-automatic requirements mining with imported and retrieved knowledge achieved significantly higher recall than using semi-automatic requirements mining with imported knowledge only. Thus, hypothesis 1 and hypothesis 2 are supported.

Hypothesis 3 was also supported by the RMANOVA on precision (see Table 12): no significant difference in precision across the three RMS configurations was detected in the experiment.

DV	Source	*DF*	*MS*	*F*	*p*	η^2	Cohen's *f*
Recall	RMS Config.	2	0.861	129.76	< .001	.77	1.82
	Error	78	0.007				
Precision	RMS Config.	2	0.005	1.36	.263	.03	0.19
	Error	78	0.004				

Table 12: Results of RMANOVA for Recall and Precision

Pair comparison		Mean difference	*p**	95% confidence interval*	
				Lower	Upper
Semi-automatic with imported knowledge	Manual	19.2%	< .001	14.4%	23.9%
Semi-automatic with imported and retrieved knowledge	Semi-automatic with imported knowledge	9.7%	< .001	5.8%	13.6%

Table 13: Results of Pairwise Comparisons for Recall

6.2.2.3 External Validity Evaluation

In the previous section, the hypotheses were tested with the data obtained from student participants in a laboratory setting. Since the results shall be generalized to requirements engineers who carry out requirements mining activities in workplaces, external validity is a concern for the laboratory experiment with students. However, prior studies suggest that causal relationships are more generalizable across populations than specific characteristics (Pedhazur and Schmelkin 1991) which indicates that the causal relationships between the design principles of RMS and improved requirements mining productivity may remain across different samples.

A RMANOVA on recall was performed to compare the effects of different RMS configurations. The result showed a significant difference on participants' recall when the RMS configuration varied ($F (2, 8) = 31.74$, $p < .001$, $\eta 2 = .89$, $f = 2.82$). The pairwise comparisons with Bonferroni corrections indicated that semi-automatic requirements mining with imported knowledge outperformed manual requirements mining on recall (mean difference = 31.0, p = .007, 95% CI [13.4%, 48.7%]), but no significant difference was detected between semi-automatic requirements mining with imported knowledge and semi-automatic requirements mining with imported and retrieved knowledge (mean difference = 9.1%, p = .301, 95% CI [-7.8%, 26.1%]). When

analyzed with a more powerful paired t-test, the difference between the two semi-automatic RMS configurations was marginally significant ($t(4) = 2.13$, $p = .100$, 95% CI [-2.8%, 21.0%], $d = 0.95$). The observed effect size was classified as a large effect according to Cohen (1988). Thus, the insignificant result might stem from the very small sample size used in the field experiment. A post-hoc power analysis was conducted with G*Power 3 (Faul et al. 2007). The result showed that to detect this effect size ($d = 0.95$) with paired t-test, a sufficient power (e.g., 0.80) can be achieved by adding 7 more participants to the practitioner sample, resulting in a total sample size of 12. As expected, no significant difference was detected on precision with the practitioner sample analyzed by RMANOVA ($F (2, 8) = 0.34$, $p = .723$, $\eta 2 = .08$, $f = 0.29$).

In addition, a RMANOVA was performed with the pooled data from the laboratory and the field experiment and specified "role" as a between-subject factor to differentiate the student sample and the practitioner sample. Not surprisingly, at the significance level of 0.05, RMS configurations demonstrated the same significant effects on recall ($F (2, 86) = 84.78$, $p < .001$) and no effects on precision ($F (2, 86) = 0.39$, $p = .682$); neither a main effect of role nor an interaction effect between the RMS configurations and role was detected on recall and precision.

The above analyses did not reveal evidence that the practitioner sample demonstrated a different behavioral pattern on recall and precision when using the different RMS configurations compared with the student sample. There was no evidence showing that the conclusions drawn from the laboratory experiment could not be generalized to practitioners in a field setting. However, due to the small size of the practitioner sample used in the field experiment, the results have to be treated with caution.

6.2.2.4 Analysis of Additional Data

Based on the previously introduced definition, requirements mining productivity is the quality of the identified requirements divided by the invested requirements mining effort. Since the invested requirements mining effort was measured with time and was kept constant in the experiment, the results support that the deployment of the two design principles can improve requirements mining productivity. Alternatively, the

invested requirements mining effort can also be measured by the frequency of keystrokes and mouse clicks which is often termed as physical effort (Tamir et al. 2008). In the student experiment, a screen capture tool was installed on participants' computers that automatically captured their keystrokes and mouse clicks during the experiment. Tested with RMANOVA, the frequency of keystrokes and mouse clicks was significantly different across different RMS configurations ($F(2, 78) = 50.15$, $p < .001$, $\eta 2 = .56$, $f = 1.14$). The pairwise comparisons with Bonferroni corrections showed that the use of semi-automatic requirements mining with imported knowledge significantly reduced the frequency of keystrokes and mouse clicks from an average of 251.2 (SD = 62.61) to an average of 185.0 (SD = 55.94) (mean difference = 66.2, $p < .001$, 95% CI [42.2, 90.2]). The use of semi-automatic requirements mining with imported and retrieved knowledge further reduced the frequency of keystrokes and mouse clicks to an average of 157.1 (SD = 50.58) (mean difference = 28.0, $p = .013$, 95% CI [4.9, 51.0]). The invested requirements mining effort measured by frequency of keystrokes and mouse clicks was reduced by 37.5% with the deployment of the two design principles. Consequently, requirements mining productivity measured by recall per keystroke or mouse click was significantly improved by the use of semi-automatic requirements mining with imported knowledge (mean difference = 0.20%, $p < .001$, 95% CI [0.15%, 0.25%]) and further improved by the use of semi-automatic requirements mining with imported and retrieved knowledge (mean difference = 0.21%, $p = 0.025$, 95% CI [0.02%, 0.39%]). This finding confirms the improvement of requirements mining productivity by the deployment of the two design principles and provides support for reduction of physical efforts by the design principles. Overall, to achieve a certain level of quality of the identified requirements, participants with the semi-automatic RMS require shorter time and invest lower physical effort.

6.3 Summary

In this chapter, the methodology and results of two evaluations were presented: an interim evaluation, conducted as a simulation and an ex post evaluation, conducted as an experiment.

The simulation investigated the interplay of RMS' knowledge base characteristics and processing characteristics. More specifically, it was explored how the amount and type of knowledge affect requirements mining quality in two consecutive simulations. While the amount and origin of knowledge significantly affected requirements mining quality, results for the effects of project-specific knowledge were ambiguous.

The experiment evaluation focused on an analysis of the final artifact version's effectiveness. More concretely, it was investigated how the design principles of RMS described in section 5.2.2 affect requirements mining productivity. Results indicate that both design principles, semi-automatic requirements mining (DP1) and the usage of imported and retrieved knowledge (DP2) have significant positive effects on requirements mining productivity. The outcomes of these evaluations as well as the results of the prior artifact design will be discussed in the following chapter.

7 Discussion

In this chapter, first the evaluation results and then the general results of this research will be discussed. The chapter will accordingly be structured along three guiding questions: First, how can the evaluation results be explained and what can be learned from them? Second, how can the overall results of the study be assessed? And third, to which extent did the research address the depicted research gaps?

7.1 Discussion of Evaluation Results

How can the evaluation results be explained and what can be learned from them? To answer this question, first the simulation results and then the experiment results will be reflected.

7.1.1 Simulation Results[42]

The first series of simulation runs demonstrated the effects of the variable "origin of knowledge" on requirements mining quality. Interestingly, the simulation results showed that the usage of retrieved knowledge outperformed the usage of imported knowledge already after three documents. A possible explanation for this can be derived from the different degrees of domain-specificity of the utilized knowledge. Retrieved knowledge can potentially provide a higher degree of domain-specificity than imported knowledge. While imported knowledge provides a solid basis of terms generally associated to the core domain (in this case travel management), this domain can be divided into sub-domains using their own vocabulary. In the exemplary travel management domain, terms like "destination", "start date" or "direct connection" can be associated to general domain knowledge. However, more specific terms like "type of rail card" or "smoking allowed lifts" are specific to the sub-domains of train transport and shared car transport. Consequently, while imported knowledge can be used to correctly identify and classify a core set of general requirements (resulting in a recall of almost 0.4 in the simulation), more specific requirements (which required sub-domain knowledge) were only captured after using retrieved knowledge.

[42] Parts of this section are based on Meth et al. (2013b).

In the second series of simulation runs, it was investigated how the project-specificity of knowledge affected requirements quality. Although one could expect that the usage of project-specific knowledge would outperform project-independent knowledge, this effect was surprisingly not observable in the conducted simulation. An explanation for the different outcomes of the second simulation series could be that the project-specific texts used in the simulations did not provide sufficient additional knowledge which was not already contained in the project-independent documents. This interpretation is depicted in Figure 34.

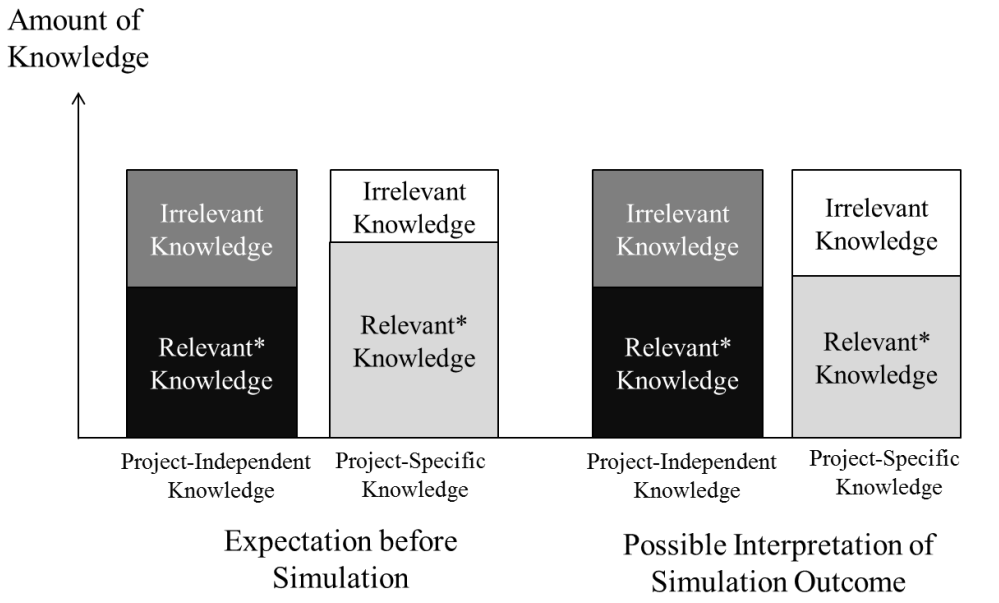

Figure 34: Distribution of Relevant Knowledge

In general, for each NLRR to be analyzed, some knowledge items in the knowledge base are relevant and others not. By using project-specific knowledge, the amount of *relevant* knowledge for a specific NLRR can rise, but does not have to. As depicted in Figure 34, before conducting the simulation, it was expected that project-specific knowledge would contain a larger amount of relevant knowledge than project-independent knowledge. This would have resulted in an increased recall. Figure 34 also shows a possible explanation why this effect has not been observed. For the documents

used in the simulation, the differences in relevant knowledge might have been smaller than expected. Consequently, this resulted in small recall differences as well.

The question whether to build knowledge "bottom-up" by a group of regular project-members (as conducted in the simulation with retrieved knowledge) or "top-down" by individual domain experts (as conducted in the simulation with imported knowledge) has been widely discussed in general knowledge engineering and knowledge management literature (Alavi and Leidner 2013; Markus 2001; Schreiber et al. 1999). Initially, the knowledge engineering field proposed a systematic top-down approach to acquire and maintain knowledge from stakeholders. Various knowledge engineering methodologies, such as Common-KADS (Schreiber et al. 1994) and tools such as Protégé (Eriksson and Musen 1993) have been suggested. To reduce knowledge acquisition efforts, one important principle from the very early beginning was the establishment of reusable knowledge bases (Patil et al. 1997). Complementing manual knowledge engineering, advanced knowledge discovery techniques to extract knowledge from source data such as documents have been suggested. For example, the field of ontology learning (Maedche and Staab 2001) extracts and suggests ontological structures from existing domain data to the knowledge engineer. Recently, the rather expert-driven knowledge engineering approach for establishing knowledge has been complemented by an end-user-driven bottom-up approach following a Web 2.0 paradigm; user-generated classifications, also known as folksonomies (Wu et al. 2006) represent one important example. Following this approach, users incrementally build knowledge bases by themselves. These bottom-up knowledge bases can be leveraged to create suggestions. An according approach is followed by the social bookmarking and citation management system Bibsonomy (Benz et al. 2010).

Looking at these different paradigms, the question arises how to build and maintain knowledge for advanced RMS. The evaluation results provide evidence for the major potential of following a bottom-up approach. Supplying an initial knowledge base positively impacts recall at the beginning of a requirements mining process. However, the bottom-up approach outperformed the top-down pre-defined knowledge base approach already after three documents. The second interesting insight of the results is that reusing knowledge across different software development projects within the same

or similar domains seems to be a promising approach. Both, software vendors and customer companies may leverage this potential. First, from a vendor perspective, software development projects can reuse knowledge across releases. Second, from a customer perspective, knowledge can be accumulated within a Line-of-Business such as a procurement department running multiple IS implementation projects within this domain. While the simulation, using a small dataset, already resulted in recall rates about 60%, even larger values are possible using more extensive datasets (Casamayor et al. 2011; Cleland-Huang et al. 2007). Although these results show that automated requirements mining cannot fully replace manual efforts performed by a requirements engineer, it can significantly support humans and thereby reduce the number of overseen and omitted requirements (Berry et al. 2012).

7.1.2 Experiment Results[43]

The evaluation aimed at measuring the effects of different design principles of RMS on requirements mining productivity in comparison to manual requirements mining. More specifically, it was investigated how semi-automatic requirements mining (DP1) and the combined usage of imported and retrieved knowledge (DP2) affect requirements mining recall and precision in a fixed time period.

Concerning DP1, it was found that the use of semi-automatic requirements mining significantly improved requirements mining recall. Different explanation patterns can be applied to this result. First, the automation process provided the participants with an initial set of identified requirements that already represented a substantial recall (54.0%). Therefore, in comparison to the manual requirements mining task, in which participants started with an unprocessed document, a higher final recall could be assumed, provided that participants trust the suggestions of the automatism. The increase of recall from the initial 54.0% (provided by the automatism) to the final 69.8% indicates that participants trusted the recommendations of the automatism sufficiently enough to let them use at least a part of their time to increase recall through manual requirements mining of additional requirements (rather than using the entire time to correct potential mistakes of the automatism).

[43] Parts of this section have been published in Meth et al. (2012a).

As expected, DP1 did not significantly affect precision. The automatism resulted in an initial precision of 79.0% using imported knowledge which is comparable to the average precision value achieved during the manual requirements mining task (71.0%). Manual adaptations and extensions that were made during the experiment task slightly reduced the initial precision, resulting in a value of 72.0%. This value is between the precision values of the automatism and the value of manual requirements mining which reflects the semi-automatic nature of the task.

Concerning DP2, it was found that the additional use of extracted knowledge further improved requirements mining recall. A possible explanation for this effect is that a more extensive and domain-specific knowledge base results in a higher initial recall provided by the automatism. This assumption could be confirmed, as the initial recall of the automatism rose from 54.0% to 75.0% through the activation of DP2. To assess the generalizability of these results, it is important to revisit the corresponding preconditions of the findings. The extension of the knowledge base through extracted knowledge resulted from a previous, manual requirements mining conducted by a domain expert. This manual requirements mining was based on different documents and a different application context than the experiment itself, but referred to a similar domain (travelling). These quality pre-conditions (extension of knowledge done by an expert and using documents of the same domain) enabled the automatism to determine requirements with significantly increased recall and with a precision comparable to manual requirements mining. Consequently, the final result also showed this recall/precision pattern. To achieve comparable results in a field setting, it is therefore important to enforce the described quality pre-conditions which can be supported by the RMS itself (e.g., through specific expert user roles and the mandatory assignment of documents to domains), or by organizational enforcement (e.g., through recurrent, mandatory quality checks of the knowledge base).

Similar to DP1, DP2 did not significantly affect precision. The automatism resulted in an initial precision of 75.0% using imported knowledge which is again comparable to the average precision value achieved during the manual requirements mining task (71.0%) and therefore can be explained analogously.

7.2 Discussion of Overall Results

How can the overall results of the thesis project be assessed? Similar to the analysis of related work in section 3.7, the research presented in this thesis will be analyzed using the conceptualized analysis framework for RDS research works (see Figure 35).

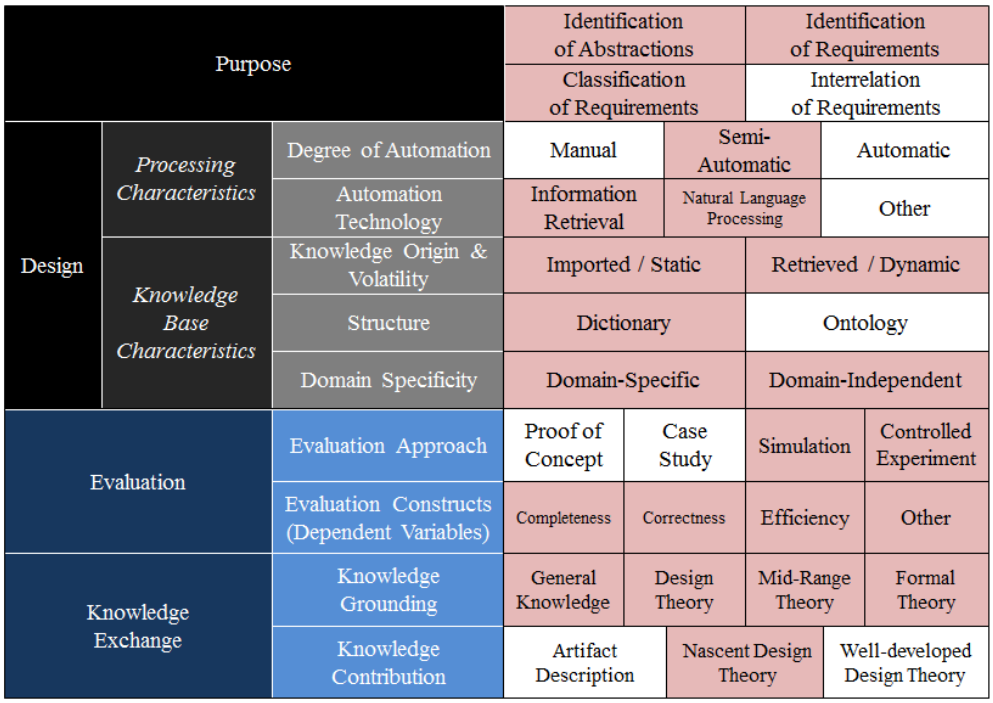

Figure 35: Analysis Result for Research Conducted in Thesis Project

The presented artifact REMINER aims at the identification and classification of requirements. Like former approaches, the system identifies abstractions to enrich the knowledge base. To improve requirements mining productivity, semi-automatic processing of NLRR is supported. Requirements are proposed by an automated algorithm applying IR and NLP techniques and can then be manually adapted. The provided knowledge base holds imported, static knowledge which is subsequently complemented with retrieved, dynamic knowledge items. The knowledge base is structured as a dictionary and holds both domain-specific and domain-independent knowledge. The implemented artifact has been evaluated in a simulation, investigating the interplay of processing and knowledge base characteristics during the first design cycle. In the second design cycle the final design principles of the system have been

evaluated in a controlled experiment, including actual system usage. In the simulation, requirements completeness has been evaluated through the measurement of recall. In the experiment, requirements mining productivity has been assessed, combing measurements of requirements completeness (recall) and requirements correctness (precision) both in a fixed time frame (to incorporate requirements mining efficiency).

Different types of knowledge have been used to justify and ground the artifact design. Based on formal, behavioral decision theory, goals of decision makers have been identified and the basic relationship between advice quality and decision quality has been explained to derive design requirements. Subsequently, design and mid-range theory on DSS has been utilized as an analogy to predict the impact of different types of DG on the requirements mining process and its outcome. From this theoretical basis, applying additional general knowledge to the requirements mining process and the design of RMS, design principles for RMS have been derived.

The contributed knowledge exceeds a pure description of the artifact. The design product has been abstracted and generalized, presenting knowledge as operational principles and a blueprint architecture. Components of a nascent design theory such as design requirements, principles, features and constructs have been presented. Going beyond the typical constituents of nascent design theory, testable hypotheses have been derived and tested. Possible extensions to this research to further develop the proposed design theory will be described in section 8.2.

7.3 Discussion of Research Gap Congruence

To which extent does the research address the depicted research gap? Based on the analysis of the overall results, the three research gaps identified in 3.7.2 shall be revisited to assess congruence with them.

The first research gap, referred to the current state of RMS evaluation. Related work in the area of RMS has been evaluated through simulations, comparing the results of automated requirements mining with a predefined gold standard. Comparative evaluation results, investigating if RMS improve requirements quality and requirements

mining efficiency in comparison to manual discovery, are hardly available. This thesis project addresses this gap by conducting a comparative, experiment-based evaluation.

The second research gap was related to the knowledge contribution of existing RMS publications. While the related work which has been analyzed contains detailed descriptions of specific implementations, a codification and abstraction of the demands to be fulfilled by RMS and the concepts addressing these demands is missing. In this thesis project, RMS design has been abstracted and generalized to design requirements and design principles which are applicable to different instantiations of RMS and which are independent of a specific technology. Although the general design has been instantiated in concrete design features of an artifact, the design requirements and principles are transferable to other systems of this class.

Finally, the third research gap referred to the theoretical grounding of existing RMS. Related work in the field of RMS is based on general empirical and non-empirical knowledge, but lacks theoretical justification. Therefore, it is difficult to assess if the proposed design approaches really provide good or even optimal solutions for the given problem. In this thesis project, the artifact design is grounded on a broad basis of different types of knowledge from formal theories to practical requirements mining experiences.

Decision making theory has been used to deriving design requirements for RMS from general goals of human decision makers. Subsequently, design principles addressing these requirements were identified based on the application of different types of DG to the requirements mining process. Furthermore, results of existing RMS research have been incorporated, providing additional general and design knowledge.

7.4 Summary

In this chapter, results of both the simulation and the experiment evaluation have been reflected, discussing possible explanations for the observations and providing additional evidence corroborating the findings. Then, the analysis framework conceptualized in chapter 3 has been applied to the research presented in this thesis to discuss the overall results and assess congruence with the identified research gaps. In the next chapter, concluding thoughts on this thesis project will be shared.

8 Conclusion

In this chapter, the results of this thesis will be summarized, limitations and future work will be discussed and contributions of the conducted research will be outlined.

8.1 Summary

As depicted in the introduction chapter, this research project aimed at attaining three goals. First, a theoretically grounded design theory for RMS should be derived. Second, an artifact based on this design theory should be implemented. Third, it should be evaluated, if requirements mining supported by this artifact actually results in an increased productivity (in comparison to manual discovery). These three goals were summarized in the following research question: How can a system be designed, which aims at improving requirements mining productivity over manual discovery?

To answer this question, chapter 2 provided a conceptual basis. First, general definitions of requirements, requirements engineering and the specific process of requirements discovery were provided. Then, requirements discovery was related to existing software development and requirements engineering approaches to outline contextual differences and specificities.

In the third chapter, an analysis framework for RDS was developed, introducing different dimensions and characteristics and exemplifying them with existing RDS research. The framework classifies RDS according to their purpose, processing and knowledge base characteristics. Moreover, RDS research can be categorized concerning its different evaluation and knowledge exchange approaches. The presented framework has then been applied to existing RMS research, summarizing the related work for this thesis and outlining the research gaps to be addressed.

In chapter four, the methodology applied in this research project was presented. Starting from an introduction to DSR as the underlying research paradigm, the dualist nature of design as a process and a product has been discussed. Building on this differentiation, alternative process- and product-oriented DSR frameworks were presented, resulting in a selection of two frameworks to be applied in this thesis: one process-oriented framework and one product-oriented. In the following, the specific research design of

the thesis was depicted followed by an ontological and an epistemological reflection of this approach.

Chapter five presented one of the core contributions of the thesis, a design theory for RMS. The presentation was structured along the eight design theory components proposed by Gregor and Jones (2007). Based on decision making theory, design requirements and design principles for RMS were derived. Then these principles were implemented in actual design features of an expository instantiation. Additionally to the conceptualization of this artifact, principles of implementation and the artifact's mutability were described. Finally, a research model, consisting of three testable hypotheses was conceptualized based on decision making theory and general requirements mining knowledge.

In chapter six, the specific methodology and results of two artifact evaluations were presented. First, the results of a simulation, representing an interim evaluation, were provided. In this simulation, the interplay of RMS' knowledge base and processing characteristics was investigated, exploring the effects of different amount and types of knowledge on requirements mining quality. Eventually, the experiment results, investigating the artifact's effectiveness, were described. In this experiment, the effects of different RMS design principles on requirements mining productivity were analyzed. Both design principles were found to improve requirements engineers' individual requirements mining productivity.

In chapter seven, results of the two evaluations were reflected, discussing possible explanations. Subsequently, the analysis framework introduced in chapter 3 was applied to the research presented in this thesis to discuss the overall results and assess congruence with the identified research gaps.

8.2 Limitations and Future Research

In order to adequately interpret the implications of the findings, the following limitations of the thesis need to be considered. The discussion of the limitations will be oriented towards the structure and outcomes of the study starting with the analysis framework for RDS and its content, via the conceptualized design theory to the final evaluations in a simulation and an experiment.

Reflecting the presented analysis framework for RDS and its content, it needs to be considered, that the classification of the related work was based on the author's specific judgment and experience and that other researchers might have judged differently. In addition, the content and structure of the *analysis framework* itself can only represent an excerpt of interesting characteristics to be investigated. Future research might complement the literature analysis, classifying the same or similar sets of papers according to additional dimensions and characteristics.

In the conceptualization of the *design theory*, leveraging decision making theory, a specific theoretical viewpoint was applied to underpin design requirements and design principles. Choosing alternative theoretical viewpoints could result in additional design requirements and principles. However, the results of the evaluation confirm that 1) both design principles positively affected the quality of approved requirements and 2) the quality of approved requirements (the decision which has been taken) was strongly determined by the quality of proposed requirements (the given advice) which is in accordance with decision making theory. Therefore, there is evidence that the theory provides an appropriate basis for the design of RMS and the derivation of meaningful design requirements and design principles. Concerning the knowledge contribution, the self-assessment of this research project presented in section 7.2 classified the derived theory as a "nascent design theory". To reach the next level of knowledge contribution in the analysis framework (a transformation to a well-developed design theory), additional research could be conducted. For example, further behavioral aspects of requirements mining, such as trust, could be investigated, aiming to extend the explanatory power of the design theory and increase the understanding of embedded phenomena.

Additional limitations apply to the conducted *simulation*. First, assessing external validity, the conducted simulation series was performed using one specific system (REMINER) which might limit generalizability. However, due to the generic design principles which were followed in the conceptualization of the system, results should be generalizable to other knowledge-based RMS. Furthermore, although a specific domain (travel management) was used, this domain is comparable to a large amount of other domains of similar complexity. Future work could complement the conducted study by

a replication of the simulations in a more complex domain. Second, evaluating internal validity, the model did not include variables which capture additional characteristics of the utilized requirements documents (like readability and length). Instead of varying these variables, documents of comparable readability and length were used. Future research, however, might investigate how these two variables affect requirements mining quality. Moreover, the definition of the gold standard used in the simulations involves subjective interpretations. This risk was mitigated by involving three different experts in the definition. Third, assessing construct validity, the number of documents was used as a measure for the amount of knowledge. Although it can be assumed that additional documents added further knowledge and the results show that in fact more documents led to a larger amount of recognized requirements, alternative measurements (e.g. a direct variation of the number of knowledge items) could be applied. However, these alternatives were not chosen in order to approximate the simulation to real life conditions in which entire documents instead of single knowledge items would be added to retrieved knowledge. Nevertheless, future work might investigate if a more direct alteration of the amount of knowledge through the number of knowledge items results in the same effects as the presented simulation.

Reflecting the conduction of the *experiment* evaluation, a further limitation can be seen in the fact that the laboratory experiment sessions were conducted with master IS students, not with experts, which constrains the external validity of the findings. However, the replication of the experiment with a small group of experts showed evidence that the same results pattern which has been observed in the laboratory setting can be expected in a field setting as well. Another limitation can be seen in the analysis of the experiment text data which was based on manual document analysis. Although this analysis was thoroughly conducted, manual analysis is error-prone and can reduce reliability. Yet, the fact that results were analyzed by two researchers independently and with a high inter-rater reliability (98.97% in the documents which were coded twice) provides evidence that this did not have a major impact.

There are many possible extensions to this work. Agreeing with Hevner et al. (2004) that DSR is inherently iterative, future research could extend the presented theory through the conduction of additional design cycles. During these cycles, alternative

theoretical lenses could be applied or a more intensive observation of the artifact's usage in an actual implementation project (for example in form of a case study) could be performed. Both extensions promise interesting adaptions and enrichments of the identified design theory components. From an evaluation point of view, a replication of the experiment study in a different domain could also add interesting insights. In the experiment, the traveling domain was adopted which is reusable for a wide range of applications. When the domain is highly specific and dynamic, domain-specific knowledge becomes scarce and cannot easily be acquired and imported into the RMS. In this case, the RMS might be less useful since many requirements need to be manually established and might not be reused in further requirements elicitation. Future research could use a more sophisticated domain and differentiate participants according to their domain knowledge, specifically examining the moderating effects of participants' domain knowledge on the relationships between design principles and requirements elicitation productivity.

Furthermore, an extension of the artifact's functional scope could be investigated. For example, the artifact could be augmented to support an integration of the outcomes of requirements mining to subsequent requirements engineering or general software development activities. Reflecting the specificities of different software development approaches presented in 2.2, it would be interesting to find out, how requirements mining outcomes need to be modified or extended to enable a seamless integration into these specific processes. For instance, user-centered approaches, which often follow a task-oriented approach to requirements elicitation, might need other requirements categories than system-centered approaches.

8.3 Contributions

The contributions of this thesis will be summarized in the following from a theoretical and practical point of view.

8.3.1 Theoretical Contributions

From a theoretical perspective, the study provides the following key contributions: First, it derives an analysis framework for works in the area of RDS, going beyond the basic

classification provided by Berry et al. (2012). Besides the application in this paper, the framework might be used to classify and evaluate future research in this area. Based on this framework, the current state of the art in RMS has been depicted. Providing an overview of existing works, this compilation might be useful as a starting point for scholars who are about to research in this area.

Second, the results of the thesis extend the design theory body of knowledge for software development systems. More specifically, a design theory for RMS has been conducted. Due to the abstraction and codification of the design to generic design requirements and design principles, the findings are generalizable from the specific artifact to the class of RMS. The prescriptive theoretical findings of the study may guide future research in designing efficient RMS.

Third, as described earlier, RMS should improve requirements engineers' productivity in the corresponding process to provide an added value in comparison to manual requirements mining. The conducted study complements existing research on RMS, investigating if this expected productivity improvement can actually be observed. Complementing these experiment results, the outcomes of the conducted simulation series provide further insights about the impact of different forms of background knowledge on requirements mining quality (which is one of the determinants of productivity).

Finally, beyond the topical aspects of the thesis, a contribution to the ongoing methodological discussion in the design science context is aspired. Based on the conceptualization of design principles, an experimental evaluation was designed and conducted that allows quantifying the effects of each principle on a dependent variable. Going beyond an assessment of the artifact's overall effect, this procedure allows precise inference from the evaluation back to the design process. This approach could inform other design researchers in the evaluation of their artifacts and the underlying design principles.

8.3.2 Practical Contributions

From a practical point of view, software vendors and customer companies can use the following results and insights of the thesis.

First, the overview of different RDS capabilities provided in the related work chapter of this thesis can be used by requirements engineering software vendors to get an overview of existing research about systems supporting requirements discovery. This state-of-the-art overview could help them to identify worthwhile areas for the functional extension of their products. While the related work (in contrast to the derived design theory) does not provide technological details of each class of RDS, it could still be used to gather information for strategic decisions, for example as an additional input for portfolio management sessions or to complement market research.

Second, the simulation and the experiment showed the potential benefits of integrating requirements and knowledge engineering activities. The evaluations provide evidence that the reuse of knowledge across different software development projects within the same or similar domains can result in better requirements specifications. Software vendors could accordingly benefit from reusing knowledge across different products of the same product group. Similarly, customer companies could share knowledge across different applications of the same Line-of-Business. Apart from using an RMS, knowledge reuse in requirements engineering can also be fostered by other technologies (e.g., domain-specific wikis), directories (e.g., glossaries) or organizational means (e.g., lessons learned sessions or specific roles in the development team).

Finally, the conducted study can help requirements engineering software vendors to improve their software packages with regard to automated requirements mining capabilities. While support for manual requirements mining has been incorporated to selected commercial software packages (e.g., IBM Rational Doors), automated mining support is still scarce. The depicted design theory can inspire and guide future commercial implementations by constraining the solution space for RMS and hereby improving design outcomes. When implemented in commercially available software and applied in a requirements mining process, the design prescriptions of the derived design theory can help to increase the individual productivity of requirements engineers and hereby address a considerable problem of current requirements engineering practice.

Appendix A: Publications

Publications related to this thesis

1. Meth, H., Maedche, A., and Einoeder, M. 2012. "Exploring Design Principles of Task Elicitation Systems for Unrestricted Natural Language Documents," in *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, pp. 205-210.
2. Meth, H., Li, Y., Maedche, A., and Mueller, B. 2012. "Advancing Task Elicitation Systems - An Experimental Evaluation of Design Principles," in *Proceedings of the International Conference on Information Systems (ICIS 2012)*, Paper 3.
3. Meth, H., Li, Y., Maedche, A., and Mueller, B. 2012. "Understanding Design Principles of Task Elicitation Systems - An Experimental Evaluation," in *Proceedings of JAIS Theory Development Workshop*, Paper 22.
4. Meth, H., Brhel, M., and Maedche, M. 2013. "The State of the Art in Automated Requirements Elicitation," *Information and Software Technology*, forthcoming.
5. Meth, H., Maedche, A., and Einoeder, M. 2013. "Is Knowledge Power? The Role of Knowledge in Automated Requirements Elicitation," in *Proceedings of the 25th International Conference on Advanced Information Systems Engineering (CAiSE 2013)*, forthcoming.

Further Publications

1. Meth, H., Maedche, A. 2010. "User-centered requirements elicitation for Business Intelligence solutions," in *Proceedings of the FKBI 10*, pp. 39-44.
2. Botzenhardt, A., Meth, H., Maedche, A. 2011. "Cross-Functional Integration of Product Management and Product Design in Application Software Development: Exploration of Success Factors," in *Proceedings of the International Conference on Information Systems (ICIS 2011)*, Paper 10.
3. Maedche, A., Botzenhardt, A., and Meth, H. 2012. "Usability und User-Centered Design," *Das Wirtschaftsstudium : WISU* (41:8/9), pp. 1074-1077.

4. Scheiber, F., Wruk, D., Oberg, A., Britsch, J., Woywode, M., Maedche, A., Kahrau, F., Meth, H., Wallach, D., and Plach, M. 2012. "Software Usability in Small and Medium Sized Enterprises in Germany: An Empirical Study," in *Software for People*. 1st Edition. Berlin, Germany: Springer, pp. 39-52.

5. Gaß, O., Meth, H., Maedche, A. 2013. "PaaS characteristics for productive software development - An evaluation framework," *IEEE Internet Computing*, forthcoming.

Appendix B: Interview Transcripts[44]

Car Sharing Interview 1

INT: So, let's start. Just explain how the app looks like.

VPN2: My main goal is to publish my trip very easy and very fast, so for me an app looks like an easy welcome interface. Then I can select "driver" or "passenger". For me, I will use "driver". After this interface I can fill another UI with login information like user-name and password.

INT: So you are already registered? How works that?

VPN2: Good question. If my acc doesn't exists, I have the opportunity to create a new one. The app needs special information like first name and surname (only real names are accepted!), nickname, age, my hometown (maybe with real address-information to check if the person is real). Also care information (seats, size,.. maybe for the girls the colour). My email address and very important, my cell phone number.

INT: Okay, so you enter these information for the registration or for your driving offer?

VPN2: I have only one care and only one phone and then its easier for me to enter once these information and the app can use these for all my offers.

INT: Okay. So what happens exactly if you want to start a new offer?

VPN2: I have the choice between options: create a new offer or edit previous offer to create with this information a new offer because the most drivers have all time an similar trip. Like students travelling between university and their parents. If I select "new offer", the app needs the start and destination location. ;Maybe I can give further locations which I will cross like time, additional information, costs, if it is an round trip or not.

INT: Okay, so now you started your offer and what will happens after that?

VPN2: After I created and published my offer, I should wait to get requests. Every interested people can write me a message via email, sms or call. Ah, one additional point for "creating offer": For me it will be perfect if I have the chance to give some criteria like whether pets are allowed, whether I prefer more male or female passengers

[44] Parts of this data have been utilized in Meth et al. (2012a) and Meth et al. (2013b).

and that stuff.

INT: Okay, fine. Does the process end after somebody found you and sent you a message or are there other steps?

VPN2: yes. There are more steps like I could cancel the offer or maybe the app/portal have a comment or ranking system to check the persons like at couchsurfing. Because nobody trusts an unknown person.

INT: Ah okay, I see. So after the drive, the driver and the passengers leave references at each other?

VPN2: Exactly.

INT: Okay, that's all?

VPN2: Yes. Now I have the main functionalities included.

INT: Okay, that's perfectly fine. Thank you.

Car Sharing Interview 2

INT: Okay, let's start. Just start explaining what happens if you open the app on your iPhone?

VPN3: Okay. I have the app new so I first need to registrate in the system and I have a car so I would like to be the driver. So first I make the rest of the registration and then I am a member of the community.

INT: What do you fill out for the registration?

VPN3: The name and the place where I am living now and the two places between which I will drive, on which time and at which date I normally drive. Maybe my gender. My age. Maybe also preferences concerning the guys I will take with me, for example if I only want to take girls with me.

INT: Also your contact information?

VPN3: Ahm…it depends. If it is public for all, I wouldn't do it. Only if is in the system but not everyone who enters the homepage can see my contact details. Then I am in the system and for me it's important that the app is well-structured because if not it's too complicated to get through the system and you could get lost in all the information. Also important for me is that it runs fast and that it doesn't take too long to load up things. And it should also be nice to look at.

INT: Okay. And after you logged in to your account, what do you do then?

VPN3: I will first enter the dates and the rides I will make in the next weeks or months if I know them already.

INT: So at the registration you already mentioned your data, so do you have to write them new or can you take them over from these information?

VPN3: No I would say the registration is only to get a little overview over it. Okay. The users can see like I am driving home each Saturday normally. But to get concrete information they need to go to a site or so.

INT: And then after entering your information, what kind of other information is necessary?

VPN3: It's important the date, then when will I leave, which cities will I pass through or which cities I will pass but not stop or maybe how many places are available in my car. If I want to take people with me: who are smokers or not.

INT: Okay, and then?

VPN3: After that when I put up all my information in the system, I would click enter and hope that some guys will call me.

INT: Okay, if they find you, what can they do then?

VPN3: They can write me a personal message. I think it's a good opportunity to get to know each other on a personal way. So maybe you think: Oh the message doesn't look very nice, I won't take him with me.

INT: Okay. Do you have any other option besides the email and messages?

VPN3: Yeah, maybe if you have some friends or some guys you had contact with them for a longer period, you can have like in your email account a folder with all your friends. You could put them in so you know: Ah it's your friend, you can trust him and can go with him.

INT: Okay, so we are done. Thank you!

Car Sharing Interview 3

INT: Just explain what will happen after you open the app!

VPN6: As a driver I would expect a start page. But it does not matter what is the start

page. I think the main point for a driver is to have an easy access to insert your ride. So, if you want to add a ride to have a menu item to add a ride or something like that.

INT: Can you click on that one?

VPN6: Yeah.

INT: Ok, and then?

VPN6: And then you have a few data fields, that you can really easy and intuitive add a ride and you need starting position, destination and the time and date. I don‘t think, that you need more information for inserting a ride. You just have to say where you want to start and where to go to and when you want to drive. I would prefer if you also can specify the exact position which is nowadays realized at mitfahrgelegenheit.de. So, for example if you have as starting point Mannheim you can choose in a drop-down-menu or in the i-phone in the menu some positions like nearly 90% of the rides start at the post office. So, you can choose Mannheim as position and then the exact position would be the post office. And that would be perfect if they have some recommendations.

INT: Do they also have to specify additional data like contact-details?

VPN6: As I think in the app the insert fields should be very few, so it should be easy to add something like that. I would have something like a management-function. You have least information what car do you have or what license you have, that the others find you. These information you have to insert only once. So, you would have something like management fields, menu item, where you can add this information and when you later add an ride the others will always see the same.

INT: And after you inserted your ride, what happens then if somebody finds you?

VPN6: You would also insert your phone number in this management area, so the others will see the number. You can choose in this area which numbers you will show, for example only the cellphone number or something like that and then the other will contact you. Perfectly it would be if you could also include this booking-service of mitfahrgelegenheit.de. So the others could simply book your ride. That you can see this person booked your ride and you get something like a push message.

INT: What do you have to do for booking a ride?

VPN6: As someone who wants to get a ride?

INT: Yes.

VPN6: He has to search for a ride. So, I want from this to this position at this time, we have a time slot most of the case. You can‘t write, I want to go at 3 p.m. or something like that, but between 1 and 4. And then you see all the drivers and you can see how many free places are in the car and you can simply book the ride. The driver will get a booked-message and will know, ok, he booked the ride. The problem here is that you don‘t have the context. If somebody calls you, you have at least one minute to talk to this person, who will take a ride, but if you have simply this booking-system you don‘t have any contact before the ride. The problem is, that you don‘t know which person will show up. This problem could probably been solved by this new identity card, where you have to insert real data in your account with your real name or something like that.

INT: And after the ride, is there any additional functionality?

VPN6: You could include something like a rating-system like holiday-check, but really easy to use. Like he is a nice guy and he has a clean car. Not is he nice or is he good-looking. Just the facts you need. The car is ok and nothing special.

INT: Ok, that‘s all. Thank you.

VPN6: No problem.

Car Sharing Interview 4

INT: Okay. So, what happens if you open the app?

VPN10: Okay. First I need to enter where I am going. So I like it if it is very easy to enter this. So it should go from difficult into simple or from overview into detail. So it should start with I enter whether I do this drive regularly or only once. And this should be like once I selected, it should automatically switch to the next category so I do not need to push another enter button.

INT: Okay.

VPN10: And from there I can enter if it is in Germany or in Europe. And then I can enter the concrete start and end point and which points are in between. And I would like it if, I mean I have my own account. So I like it if they memorize what route I usually drive so that they could propose it to me. So I don’t have to type in Mannheim over Nuremberg to Erlangen every time I enter and I drive.

INT: Okay. Then you click on the button?

VPN10: Yah. And then I click on the next button and then it shows all my telephone number and email.

INT: So these are already stored in the system?

VPN10: Yes, these are already stored but I would like to be able to change with numbers are shown by default. I don't like to deactivate every number over again. Because there are more numbers in my case and in the application. And then I like it if it is very easy if the application automatically adds the location in Google maps so I don't need to that. I just need to say if I want to do this or not. And would be really useful for me as a driver is that I have an easy access to my announce or my offer. And once I have somebody who told me that he is going with me, I can decrease the amount of people I can take with me to one or to zero.

INT: So you have an overview about everybody who wants to join you?

VPN10: No. I mean I have an overview by myself but I don't want to be called by people if my seats are already full.

INT: Ah okay.

VPN10: So it would be perfect. And right now I am not doing it this because I need to log in again and search for my offer and then click on alter and click on only one seat left. And that takes just too much time. So I want to do that really quick.

INT: And do you want to do the booking over the application or by telephone?

VPN10: I would like to have it by telephone. Because I mean I need to talk to them anyway, then I can remember how many people. But I would like to have it shown. I don't know if I do it over the application with the booking. If it then shows online that my drive is already full.

INT: Okay. And after the drive, is there anything to do after that?

VPN10: For me not. But for somebody who drives with me it might me a good idea to automatically remind him that he can rate me. So he can say how well I drove or if it was expensive or if I was in time. And I mean, maybe I can give some feedback on the people who drove with me. So other drivers can see if they should take that person with them.

INT: Okay, that's all. Thank you!

Car Sharing Interview 5

INT: Okay, let's start. Just explain what happens after you opened the app.

VPN4: Now, after opening the app, it shows a nice welcome screen where I can enter my location and the destination. My Smartphone hopefully will insert my actual position and fill out the first text field "from", so that I can instantly fill out my destination. Having pressed "Enter", I get a list of persons with cars who are going to my destination until the next 7 days.

INT: What happens if you click on one of them?

VPN4: Some data opens up, containing the estimated departure date and time, possibly the price the driver proposes, the car or at least type of car he has and how many persons are on board at the moment. The exact departure location would also be nice.

INT: Okay, what happens then?

VPN4: With a tip of a button one can immediately contact the person by mail. For example the website sends a mail with my personal data, so the person can call me back or write me a mail.

INT: Fine. Now after you contacted the person and finally made the trip, could you imagine some steps afterwards like a reference system?

VPN4: That would be a good idea, one could give feedback, how his way of driving was or how the price was. A similar system to ebay's reference system would be satisfying, I think.

INT: Could you please specify that?

VPN4: That means a system from one to five and a short text field. This could be realized in a list in the app, where all trips are listed one made.

INT: Sounds good. And in general, do you have any requirements concerning the usability and user interface of the app?

VPN4: Not really, it should be simple and clearly arranged, so that you can quickly find persons with cars who drive at the same time to your destination. Same situation, when I am the one who offers a seat in my car. Just a simple form with date, from, to, type of car, and just finished

INT: Okay, I think that's all. Thank you!

Train Reservation Interview 1

INT: So, what happens after you opened the app?

VPN7: A field where I can enter my departure location and my arrival as well as destination location opens.

INT: So you enter these and what is your next step?

VPN7: Then I enter the departure time and date.

INT: Okay, after this, you click on a button?

VPN7: Yes, I press enter and possible connections appear. This should happen quite fast (performance).

INT: How are these connections displayed?

VPN7: All connections within a period of one hour compared to the entered time should appear. The duration of the journey, the departure and arrival time, the type of train like ICE or RE should be displayed in a suitable design so that I have a good overview.

INT: Can you please specify suitable design? Is it a list?

VPN7: I think a list would be best in a chronological order depending on departure time.

INT: Okay! Now you found a suitable train, what happens after that?

VPN7: I forgot to mention something. So I would like to have the option to see where and how long the trains stops if I select a possible connection.

INT: Okay.

VPN7: After I found a suitable train I would like to have the option to either book the train and to get something like an alternative in case the train is too late. Or just to set an alter in case I want to buy the ticket at the train station.

INT: I see. How works the booking of a train?

VPN7: I select number of persons, age, possible reductions with BahnCard. Then I should have the possibility to decide if I want to pay with credit or deposit card and enter details.

INT: Which details do you have to enter?

VPN7: Credit card number and type, security number on the back and expiration date.

INT: OK. And after entering all your details?

VPN7: I confirm and then I have the choice to save the electronic ticket or send it via

email.

INT: So you can click on a button to save it? Any other functionality?

VPN7: Before I confirm I would like to have an overview of the entered data just to make sure. And as I already said I would like to have an automatic alert in case the train is too late. All the functionality should happen in a fast way and in a comprehensible manner.

INT: Okay, so in the end you confirm and then your booking is done, right?

VPN7: Correct.

INT: That's all. Thank you!

Train Reservation Interview 2

INT: Just start to explain what happens after you opened the app and what you can do then?

VPN8: Well, first of all it is important that the app starts quickly so I don't have to wait very long. And once the app is started, I want a quick overview over the possible fields, like where to start the travel, where it ends, of course possible time to start for the travel. Maybe some options to select the train. So is it a local train or a fast train, that is very important. What else? Maybe some options to indicate whether I have a bonus card or not. So that the actual price calculation is already calculated right. And after all this is entered, I want to have a clear big button to push on to see the possible connections.

INT: Okay. Now you clicked on the button and what happens then?

VPN8: Well after I entered all the information and after I clicked the button, I want to see the possible connections. All possible connections. And of course it would be helpful if those connections would be displayed which I do not have to switch the trains very often. That should be displayed properly. And of course in an easy to view manner. So not very complex so that I can quickly see all connections. Of course in a list so that I can scroll down.

INT: And after that?

VPN8: After that I want to pick one connection. Maybe that I can see further information for that connection. So the starting time, end time and maybe possible inter connections. And after selecting that one, I want that it comes quickly to the booking

options.

INT: Okay. And what kind of booking options do you have?

VPN8: Well, of course maybe there is the possibility to set up an account so I can login and I don't need to or have to enter all information every time from beginning. If that is the case, well, it would be perfect if there is some kind of one-click-solution like we know it from Amazon. So that all my data, name and all the stuff is already entered and I just need to confirm with the travel request. And then it should be quickly again. So not ten buttons. Like confirm here, confirm there. I just want to pick one option and then get a quick confirmation.

INT: And if you are not registered yet?

VPN8: Well of course there need to be the necessary fields to enter the credit card number or other payment options.

INT: What fields do you have to enter?

VPN8: Well, a radio button first to select the different options of payment. Maybe a transaction or credit card and after selecting one option further fields for the credit card number and expiration date and the CV code and other things.

INT: Okay. Then you fulfill the registration and then you get the ticket by email or how does it work?

VPN8: Yeah, of course via email is very essential and in an optimal case it would be great if there is some kind of QA code which is sent by SMS so that I have the ticket directly on my mobile phone and do not need to print out any further information because it is not always the case that one has access to a printer. So when I use a mobile app, of course I want to have the final ticket directly on my mobile phone.

INT: Okay, I think that's all. Thank you!

Train Reservation Interview 3

INT: If you open the app on your mobile phone, what do you see and what happens then?

VPN9: Oh, I think first when you see it on your iPhone, occasionally it should pop up deals like top offers, something like that. If you open it, it should be fast. But it should be like clear and it should be really easy if you open and use. So be user-friendly. And I think it will be helpful if you could just type in like when you need to go. And it would pull up like, you know, eventually the settings such as where do you need to go. How fast do you need to get there. And then where. And then obviously by price.

INT: So you get a result list?

VPN9: Yah. I think you should. But for me, I think it would be good to get the top results for one. Because you things as fast as you can. So I think you just need to type in when you need to leave, so like you can give a date or a time and then you drop the search options. As far as I can see, for design is like red.

INT: Okay. So you have a listing of all your trains and select one. What happens then?

VPN9: It would forward you to booking. I think if you have an iPhone now, it would be kind of cool if you could already have your billing information. Now I know that would be intense if you lost your iPhone. But I think there could be some kind of special like login so you could just click by, by, by. Maybe like within 30 seconds. You even don't have to type in your credit card number. It would be somehow safe. So but it have to be secure that nobody can just type in and order your own train on your phone.

INT: Okay. And after you entered all your details, what happens then or what do you have to do?

VPN9: It should give you an automatic receipt via email, I think. Saying that you bought it. At first a screenshot. It should show you like an example of exactly how it looks like if you would pick it up at the station and also how it would look like online. So it would show you like here is what it looks like if you print it out and here is what it looks like if you pick it up at the station and give you like further information.

INT: So you can print it out or pick it up on the station, you have the choice?

VPN9: Mhm, Yes.

INT: And any other options like, you know, send it via email or anything else?

VPN9: I think that maybe, if there are any updates like say your train is too late. It should pop up on the app itself. Or if any other interferences so like say that there is some kind of strike in Italy. Because last time we were in Italy there was a strike. It would tell you so you would know like here you need to refund it today.
INT: So could you also have some refund options? How would look that like?
VPN9: I think if you buy in person. If you picked it up in person, I think you should get your cash back in person. But there should also be a way to just put it directly back on your credit card. So if you missed your train there will be like an option on there like a pull-down that says like past rides, I guess. So you could see like if you missed this one and ask to refund it. And they would forward you to refunding.
INT: Ah, okay, I see. That's all. Thank you!

Train Reservation Interview 4

INT: So just start to explain what happens after you opened the app.
VPN11: So just opened the website. You have to enter your destination and where you start and where you want to go. And you just choose the time. And they will show you a timetable.
INT: So you have to click on Enter or Search first?
VPN11: Yah. At the date, you just press a button and they will show you a calendar where you can choose which date you want to travel. And also there is a selection you can just choose where and when like 12 o'clock. And if you search, they can show you all the results. And if you pick one result, they can show you how long do you have to go there and when do you have to change trains and where.
INT: So these are all listed in one table?
VPN11: Yah, in one table. And I think the better one is if there is a map, you can just press and they can show you a map like where you have to go. Like you are here and the destination is there and they can just show you how you go there. And also where you have to change your train on a map. And especially the city centers. So if you press Mannheim, they can show you a little bit around the main railway station. Ah okay. They will make it the customer easy to find if that is really where they want to go.
INT: Ah okay. So you click on a name of a city and then it opens a map?

VPN11: Yes, they can just show you. You just press the button and they can show you detail information. And if you just press this one I want to buy, they can show you all the prices they offer. So if you are student and they have these special offers for students, you can just press which one you can buy.

INT: Okay. So you have some reduced prices and you click on them and then you get an explanation about differences.

VPN11: Yeah, they are just afraid if somebody buys the wrong ticket.

INT: Ah, okay.

VPN11: And then, maybe you can combine some like an insurance or car rental at the next step.

INT: And if you want to buy one of these tickets, how works that?

VPN11: First to login. If you have an account you can log in. And if you log in they can show you detail information and they can check if this address is really your address or email address or your telephone. I think the better kind is that you can choose if you want to that they send tickets or on mobile phone or something like that to show you your ticket. And also you have to insert like your name. They have to confirm your credit card information.

INT: So what do you have to enter for that?

VPN11: Your name and also your telephone, your birthday. To confirm that is really you booked the ticket, for security.

INT: And then you click on the button and what happens after that?

VPN11: After that, they will confirm the payment way, how you will pay the ticket.

INT: And how do you receive the ticket then?

VPN11: Maybe one is, you can just pick the ticket on the main entrance station. Or they can send you. Or just use the email. Or use the mobile phone.

INT: Okay, that’s all. Thank you!

Appendix C: Imported Knowledge[45]

Term	Requirements Category
he	actor
I	actor
it	actor
she	actor
they	actor
we	actor
you	actor
accept	activity
adapt	activity
add	activity
analyze	activity
approve	activity
arrange	activity
assign	activity
build	activity
cancel	activity
choose	activity
click	activity
collaborate	activity
collect	activity
compare	activity
compute	activity
conduct	activity
confirm	activity
create	activity
design	activity
detect	activity
edit	activity
enter	activity
establish	activity
evaluate	activity
examine	activity
execute	activity
experiment	activity
fill	activity

[45] Parts of this data have been utilized in Meth et al. (2012a) and Meth et al. (2013b).

insert	activity
install	activity
interact	activity
join	activity
list	activity
maintain	activity
manage	activity
mark	activity
model	activity
observe	activity
open	activity
operate	activity
perform	activity
pick	activity
plan	activity
present	activity
press	activity
put	activity
report	activity
review	activity
search	activity
see	activity
select	activity
show	activity
test	activity
write	activity
accommodation	data
address	data
address	data
attribute	data
bank	data
card	data
cost	data
credit	data
data	data
date	data
date	data
deduction	data
departure	data
destination	data
entry	data

km	data
location	data
meal	data
miles	data
name	data
number	data
numbers	data
order	data
phone	data
position	data
price	data
privileges	data
receipt	data
stopover	data
text	data
time	data
travel	data
trip	data
vehicle	data
easy	non-functional
effective	non-functional
effectiveness	non-functional
efficiency	non-functional
efficient	non-functional
learn	non-functional
learnability	non-functional
memorability	non-functional
safe	non-functional
simple	non-functional
simply	non-functional
utility	non-functional

Table 14: Imported Knowledge Used for Simulation and Experiment

Bibliography

[Abrams et al. 2006]

Abrams, S., Bloom, B., Keyser, P., Kimelman, D., Nelson, E., Neuberger, W., Roth, T., Simmonds, I., Tang, S., and Vlissides, J. 2006. "Architectural thinking and modeling with the Architects' Workbench," *IBM Systems Journal* (45:3), pp. 481-500.

[Agerfalk et al. 2009]

Agerfalk, P. J., Fitzgerald, B., and Slaughter, S. 2009. "Introduction to the Special Issue - Flexible and Distributed Information Systems Development: State of the Art and Research Challenges," *Information Systems Research* (20:3), pp. 317-328.

[Alavi and Leidner 2013]

Alavi, M., and Leidner, D. E. 2013. "Knowledge Management and Knowledge Management Systems: Conceptual Foundations," *MIS Quarterly* (25:1), pp. 107-136.

[Ambriola and Gervasi 1997]

Ambriola, V., and Gervasi, V. 1997. "Processing Natural Language Requirements," in *Proceedings of the 12th IEEE International Conference on Automated Software Engineering*, pp. 36-45.

[Ambriola and Gervasi 2006]

Ambriola, V., and Gervasi, V. 2006. "On the Systematic Analysis of Natural Language Requirements with CIRCE," *Automated Software Engineering* (13:1), pp. 107-167.

[Appan and Browne 2012]

Appan, R., and Browne, G. J. 2012. "The Impact of Analyst-Induced Misinformation on the Requirements Elicitation Process," *MIS Quarterly* (36:1), pp. 85-106.

[Atkinson and Kuhne 2003]

Atkinson, C., and Kuhne, T. 2003. "Model-driven development: a metamodeling foundation," *IEEE Software* (20:5), pp. 36-41.

[Bacharach 1989]

Bacharach, S. B. 1989. "Organizational Theories : Some Criteria for Evaluation," *Academy of Management Review* (14:4), pp. 496-515.

[Baeza-Yates and Ribeiro-Neto 1999]
Baeza-Yates, R., and Ribeiro-Neto, B. 1999. *Modern Information Retrieval*. 1st Edition. Boston, USA: Addison Wesley.

[Baskerville and Pries-Heje 2010]
Baskerville, R., and Pries-Heje, J. 2010. "Explanatory Design Theory," *Business & Information Systems Engineering* (2:5), pp. 271-282.

[Beach and Mitchell 1978]
Beach, L., and Mitchell, T. 1978. "A Contingency Model for the Selection of Decision Strategies," *The Academy of Management Review* (3:3), pp. 439-449.

[Beck et al. 2001]
Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. 2001. "Manifesto for Agile Software Development". http://agilemanifesto.org. Accessed on 04/16/2013.

[Benz et al. 2010]
Benz, D., Hotho, A., Jäschke, R., Krause, B., Mitzlaff, F., Schmitz, C., and Stumme, G. 2010. "The social bookmark and publication management system bibsonomy," *The VLDB Journal* (19:6), pp. 849-875.

[Berry et al. 2012]
Berry, D., Gacitua, R., Sawyer, P., and Tjong, S. F. 2012. "The Case for Dumb Requirements Engineering Tools," in *Requirements Engineering: Foundation for Software Quality*. 2012 Edition. Berlin / Heidelberg, Germany: Springer, pp. 211-217.

[Beyer and Holtzblatt 1998]
Beyer, H., and Holtzblatt, K. 1998. "Contextual design: defining customer-centered systems," in *Proceedings of the SIGCHI conference on Human factors in computing systems CHI '90*, pp. 329-336.

[Boehm and Basili 2001]
Boehm, B., and Basili, V. 2001. "Software Defect Reduction Top 10 List," *IEEE Computer* (34:1), pp. 135-137.

[Bonaccio and Dalal 2006]
Bonaccio, S., and Dalal, R. S. 2006. "Advice taking and decision-making: An integrative literature review, and implications for the organizational sciences," *Organizational Behavior and Human Decision Processes* (101:2), pp. 127-151.

[Brasser and Vander Linden 2002]
Brasser, M., and Vander Linden, K. 2002. "Automatically eliciting task models from written task narratives," in *Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces*, pp. 1-6.

[Cao and Ramesh 2008]
Cao, L., and Ramesh, B. 2008. "Agile Requirements Engineering Practices: An Empirical Study," *IEEE Software* (25:1), pp. 60-67.

[Carmel 1997]
Carmel, E. 1997. "American Hegemony in Packaged Software Trade and the 'Culture of Software'," *The Information Society* (13:1), pp. 125-142.

[Carson et al. 2001]
Carson, D., Gilmore, A., Perry, C., and Gronhaug, K. 2001. *Qualitative Marketing Research*. 1st Edition. London, England: SAGE Publications Ltd.

[Casamayor et al. 2010]
Casamayor, A., Godoy, D., and Campo, M. 2010. "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *Information and Software Technology* (52:4), pp. 436-445.

[Casamayor et al. 2011]
Casamayor, A., Godoy, D., and Campo, M. 2011. "Mining textual requirements to assist architectural software design: a state of the art review," *Artificial Intelligence Review* (38:3), pp. 173-191.

[Casey and Richardson 2006]
Casey, V., and Richardson, I. 2006. "Uncovering the Reality Within Virtual Software Teams," in *Proceedings of the 2006 international workshop on Global software development for the practitioner*, pp. 66-72.

[Castro-Herrera et al. 2009]
Castro-Herrera, C., Duan, C., Cleland-Huang, J., and Mobasher, B. 2009. "A recommender system for requirements elicitation in large-scale software projects," in *Proceedings of the 2009 ACM symposium on Applied Computing - SAC '09*, pp. 1419-1426.

[Cheng and Atlee 2007]
Cheng, B. H. C., and Atlee, J. M. 2007. "Research Directions in Requirements Engineering Research Directions in Requirements Engineering," in *Proceedings of the 2007 Future of Software Engineering (FOSE '07)*, pp. 285-303.

[Cleland-Huang et al. 2007]
Cleland-Huang, J., Settimi, R., Zou, X., and Solc, P. 2007. "Automated classification of non-functional requirements," *Requirements Engineering* (12:2), pp. 103-120.

[Cohen 1988]
Cohen, J. 1988. *Statistical Power for the Behavioral Sciences.* 2nd Edition. Hillsdale, USA: Lawrence Erlbaum Associates, Inc.

[Cohn 2004]
Cohn, M. 2004. *User stories applied: For agile software development.* 1st Edition. Boston, USA: Addison-Wesley.

[Cooper et al. 2007]
Cooper, A., Reimann, R., and Cronin, D. 2007. *About Face 3: The Essentials of Interaction Design*. 3rd Edition. Indianapolis, USA:Wiley.

[Cosmetatos and Eilon 1983]
Cosmetatos, G. P., and Eilon, S. 1983. "Effects of productivity definition and measurement on performance evaluation," *European Journal of Operational Research* (14:1), pp. 31-35.

[Cybulski and Reed 1998]
Cybulski, J. L., and Reed, K. 1998. "Computer-assisted analysis and refinement of informal software requirements documents," in *Proceedings of the 1998 Asia Pacific Software Engineering Conference*, pp. 128-135.

[Davis 1982]
Davis, B. 1982. "Strategies for information requirements determination," *IBM Systems Journal* (21:1), pp. 4-30.

[Davis et al. 2006]
Davis, A. M., Dieste, O., Hickey, A., Juristo, N., and Moreno, A. M. 2006. "Effectiveness of requirements elicitation techniques: Empirical results derived from a systematic review," in *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, pp. 176-185.

[De Lucia and Qusef 2010]
De Lucia, A., and Qusef, A. 2010. "Requirements Engineering in Agile Software Development," *Journal of Emerging Technologies in Web Intelligence* (2:3), pp. 212-220.

[Diehl and Stroebe 1991]
Diehl, M., and Stroebe, W. 1991. "Productivity loss in idea-generating groups: Tracking down the blocking effect," *Journal of personality and social psychology* (61:3), pp. 392-403.

[Eriksson, H., and Musen 1993]
Eriksson, H., and Musen, M. 1993. "Metatools for knowledge acquisition," *IEEE Software* (10:3), pp. 23-29.

[Faul et al. 2007]
Faul, F., Erdfelder, E., Lang, A.-G., and Buchner, A. 2007. "G*Power 3: a flexible statistical power analysis program for the social, behavioral, and biomedical sciences," *Behavior Research Methods* (39:2), pp. 175-91.

[Fayyad et al. 1996]
Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996. "Knowledge Discovery and Data Mining: Towards a Unifying Framework," in *Proceedings of the KDD-96*, pp. 82-88.

[Gacitua et al. 2011]
Gacitua, R., Sawyer, P., and Gervasi, V. 2011. "Relevance-based abstraction identification: technique and evaluation," *Requirements Engineering* (16:3), pp. 251-265.

[Gallupe and McKeen 1990]
Gallupe, R. B., and McKeen, J. D. 1990. "Enhancing Computer-Mediated Communication: An experimental investigation into the use of a Group Decision Support System for face-to-face versus remote meetings," *Information & Management* (18:1), pp. 1-13.

[Gardner and Berry 1995]
Gardner, P. H., and Berry, D. C. 1995. "The Effect of Different Forms of Advice on the Control of a Simulated Complex System," *Applied Cognitive Psychology* (9:7), pp. 55-79.

[Gaß et al. 2012]
Gaß, O., Koppenhagen, N., Biegel, H., Maedche, A., and Müller, B. 2012. "Anatomy of Knowledge Bases used in Design Science Research," in *Proceedings of the 7th International Conference on Design Science Research in Information Systems (DESRIST 2012)*, pp. 328-344.

[Geisser et al. 2007]
Geisser, M., Heinzl, A., Hildenbrand, T., and Rothlauf, F. 2007. "Verteiltes, internetbasiertes Requirements-Engineering," *Wirtschaftsinformatik* (49:3), pp. 199-207.

[Goguen and Linde 1993]
Goguen, J., and Linde, C. 1993. "Techniques for Requirements Elicitation," in *Proceedings of IEEE International Symposium on Requirements Engineering*, pp. 152-164.

[Goldin and Berry 1997]
Goldin, L., and Berry, D. M. 1997. "AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation," *Automated Software Engineering* (4:4), pp. 375-412.

[Gould and Lewis 1985]
Gould, J. D., and Lewis, C. 1985. "Designing for Usability : Key Principles and What Designers Think," *Communications of the ACM* (28:3), pp. 300-311.

[Gregor and Hevner 2013]
Gregor, S., and Hevner, A. R. 2013. "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly* (37:2), forthcoming.

[Gregor and Jones 2007]
Gregor, S., and Jones, D. 2007. "The anatomy of a design theory," *Journal of the Association for Information Systems* (8:5), pp. 312–335.

[Harmain and Gaizauskas 2003]
Harmain, H. M., and Gaizauskas, R. J. 2003. "CM-Builder : A Natural Language-Based CASE Tool for Object-Oriented Analysis," *Automated Software Engineering* (10:2), pp. 157-181.

[Hart 2004]
Hart, A. 2004. *801 Action Verbs for Communicators: Position Yourself First with Action Verbs for Journalists, Speakers, Educators, Students, Resume-Writers, Editors & Travelers.* Lincoln, USA: iUniverse.

[Hartson and Hix 1989]
Hartson, H. R., and Hix, D. 1989. "Toward empirically derived methodologies and tools for human-computer interface development," *International Journal of Man-Machine Studies* (31:4), pp. 477-494.

[Herbsleb and Moitra 2001]
Herbsleb, J., and Moitra, D. 2001. "Global Software Development," *IEEE Software* (18:2), pp. 16-20.

[Hevner et al. 2004]
Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design science in information systems research," *MIS Quarterly* (28:1), pp. 75-105.

[Hevner and Chatterjee 2010]
Hevner, A., and Chatterjee, S. 2010. "Design science research in information systems," in *Design Research in Information Systems*. 2010 Edition. Boston, USA: Springer, pp. 9-22.

[Hickey and Davis 2004]
Hickey, A. M., and Davis, A. M. 2004. "A unified model of requirements elicitation," *Journal of Management Information Systems* (20:4), pp. 65-84.

[Hildenbrand et al. 2009]
Hildenbrand, T., Heinzl, A., Geisser, M., Klimpke, L., and Acker, T. 2009. "A Visual Approach to Traceability and Rationale Management in Distributed Collaborative Software Development," in *Lecture Notes in Informatics - PRIMIUM - Process Innovation for Enterprise Software*, Bonn, Germany: Koellen-Verlag, pp. 161-178.

[Hill and Lewicki 2006]
Hill, T., and Lewicki, P. 2006. *Statistics: Methods and Applications*. Tulsa, USA: StatSoft.

[Holmström et al. 2006]
Holmström, H., Fitzgerald, B., Ågerfalk, P. J., and Conchúir, E. Ó. 2006. "Agile Practices Reduce Distance in Global Software Development," *Information Systems Management* (23:3), pp. 7-18.

[Hooker 2004]
Hooker, J. N. 2004. "Is Design Theory Possible?," *Journal of Information Technology Theory and Application* (6:2), pp. 73-82.

[Huberty and Morris 1989]
Huberty, C. J., and Morris, J. D. 1989. "Multivariate analysis versus multiple univariate analyses," *Psychological Bulletin* (105:2), pp. 302-308.

[Huffman Hayes et al. 2005]
Huffman Hayes, J., Dekhtyar, A., and Sundaram, S. 2005. "Text Mining for Software Engineering: How Analyst Feedback Impacts Final Results," in *Proceedings of the 2005 International Workshop on Mining Software Repositories (MSR '05)*, pp. 1-5.

[IEEE 1990]
IEEE. 1990. "610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology". http://standards.ieee.org/findstds/standard/610.12-1990.html. Accessed on 04/12/2013.

[IEEE 1998]
IEEE. 1998. "830-1998 - IEEE Recommended Practice for Software Requirements Specifications". http://standards.ieee.org/findstds/standard/830-1998.html. Accessed on 04/12/2013.

[ISO 1998]
ISO. 1998. "Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability," http://www.iso.org/iso /catalogue_detail. htm?csnumber=16883. Accessed on 04/16/2013.

[ISO 2010]
ISO. 2010. "Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems," http://www.iso.org/iso/ catalogue_detail.htm?csnumber=52075. Accessed on 04/16/2013.

[Jämsä-Jounela 2007]
Jämsä-Jounela, S.-L. 2007. "Future trends in process automation," *Annual Reviews in Control* (31:2), pp. 211-220.

[John and Dörr 2003]
John, I., and Dörr, J. 2003. "Elicitation of Requirements from User Documentation," in *Proceedings of the 9th International Workshop on Requirements Engineering - Foundation of Software Quality (REFSQ'03)*, pp. 17-26.

[Jurafsky and Martin 2009]
Jurafsky, D., and Martin, J. H. 2009. *Speech and Language Processing.* 2nd Edition. New Yersey, USA: Pearson Prentice Hall.

[Kaiya and Saeki 2006]
Kaiya, H., and Saeki, M. 2006. "Using Domain Ontology as Domain Knowledge for Requirements Elicitation," in *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*, pp. 189-198.

[Karlsson et al. 2002]
Karlsson, L., Dahlstedt, Å. G., Natt och Dag, J., Regnell, B., and Persson, A. 2002. "Challenges in Market-Driven Requirements Engineering - an Industrial Interview Study," in *Proceedings of the Eights International Workshop on Requirements Engineering: Foundation for Software Quality*, pp. 101-112.

[Kincaid et al. 1975]
Kincaid, J. P., Fishburn, R. P., Rogers, R. L., and Chissom, B. S. 1975. *Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel Research Branch Report.* Millington, USA: National Technical Information Service.

[Kiyavitskaya and Zannone 2008]
Kiyavitskaya, N., and Zannone, N. 2008. "Requirements model generation to support requirements elicitation: the Secure Tropos experience," *Automated Software Engineering* (15:2), pp. 149-173.

[Kof 2004]
Kof, L. 2004. "An Application of Natural Language Processing to Domain Modelling – Two Case Studies," *International Journal on Computer Systems Science Engineering* (20:1), pp. 37-52.

[Kuechler and Vaishnavi 2008]
Kuechler, B., and Vaishnavi, V. 2008. "On theory development in design science research: anatomy of a research project," *European Journal of Information Systems* (17:5), pp. 489-504.

[Kuechler and Vaishnavi 2012]
Kuechler, W., and Vaishnavi, V. 2012. "A Framework for Theory Development in Design Science Research: Multiple Perspectives," *Journal of the Association for Information Systems* (13:6), pp. 395-423.

[Larman and Basili 2003]
Larman, C., and Basili, V. R. 2003. "Iterative and incremental developments. a brief history," *IEEE Computer* (36:6), pp. 47-56.

[Leffingwell 2011]
Leffingwell, D. 2011. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. 1st Edition. Boston, USA: Addison-Wesley Professional.

[Lemaigre and Vanderdonckt 2008]
Lemaigre, C., García, J. G., and Vanderdonckt, J. 2008. "Interface Model Elicitation from Textual Scenarios," in *Proceedings of the Human-Computer Interaction Symposium*, pp. 53-66.

[Li and Maedche 2012]
Li, Y., and Maedche, A. 2012. "Formulating Effective Coordination Strategies in Agile Global Software Development Teams," in *Proceedings of the International Conference on Information Systems (ICIS 2012)*. Paper 36.

[Lipnack and Stamp 1997]
Lipnack, J., and Stamp, J. 1997. *Virtual Teams: Reaching Across Space, Time And Originating With Technology*. 1st Edition. New York, USA: John Wiley & Sons.

[Maedche and Staab 2000]
Maedche, A., and Staab, S. 2000. "Discovering conceptual relations from text," in *Proceedings of the 14th European Conference on Artificial Intelligence*, pp. 321-325.

[Maedche and Staab 2001]
Maedche, A., and Staab, S. 2001. "Ontology learning for the Semantic Web," *IEEE Intelligent Systems* (16:2), pp. 72-79.

[Manning et al. 2008]
Manning, C. D., Raghavan, P., and Schuetze, H. 2008. *Introduction to Information Retrieval*. 1st Edition. Cambridge, England: Cambridge University Press.

[March and Smith 1995]
March, S. T., and Smith, G. F. 1995. "Design and natural science research on information technology," *Decision Support Systems* (15:4), pp. 251-266.

[Markus 2001]
Markus, M. L. 2001. "Toward A Theory of Knowledge Reuse : Types of Knowledge Reuse Situations and Factors in Reuse Success," *Journal of Management Information Systems* (18:1), pp. 57-93.

[Markus et al. 2002]
Markus, M. L., Majchrzak, A., and Gasser, L. 2002. "A design theory for systems that support emergent knowledge processes," *MIS Quarterly* (26:3), pp. 179–212.

[Marshall and Mckay 2005]
Marshall, J., and Mckay, P. 2005. "A Review of Design Science in Information Systems," in *Proceedings of the ACIS 2005*. Paper 5.

[Mayhew 1999]
Mayhew, D. J. 1999. *The usability engineering lifecycle: a practitioner's handbook for user interface design*. 1st Edition. San Francisco, USA: Morgan Kaufmann.

[Menten et al. 2010]
Menten, A., Scheibmayr, S., and Klimpke, L. 2010. "Using Audio and Collaboration Technologies for Distributed Requirements Elicitation and Documentation," in *Proceedings of the Third International Workshop on Managing Requirements Knowledge (MARK)*, pp. 51-59.

[Meth et al. 2012a]
Meth, H., Li, Y., Maedche, A., and Mueller, B. 2012. "Advancing Task Elicitation Systems – An Experimental Evaluation of Design Principles," in *Proceedings of the ICIS 2012*. Paper 3.

[Meth et al. 2012b]
Meth, H., Li, Y., Maedche, A., and Mueller, B. 2012. "Understanding Design Principles of Task Elicitation Systems - An Experimental Evaluation," in *Proceedings of the JAIS Theory Development Workshop 2012*. Paper 22.

[Meth et al. 2013a]
Meth, H., Brhel, M., and Maedche, A. 2013. "The State of the Art in Automated Requirements Elicitation," *Information and Software Technology*, forthcoming.

[Meth et al. 2013b]
Meth, H., Maedche, A., and Einoeder, M. 2013. "Is Knowledge Power? The Role of Knowledge in Automated Requirements Elicitation," in *Proceedings of the 25th International Conference on Advanced Information Systems Engineering (CAiSE 2013)*, forthcoming.

[Mich 1996]
Mich, L. 1996. "NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA," *Natural Language Engineering* (2:2), pp. 161-187.

[Mich et al. 2004]
Mich, L., Franch, M., and Novi, I. P. 2004. "Market research for requirements analysis using linguistic tools," *Requirements Engineering* (9:1), pp. 40-56.

[Mich and Garigliano 2002]
Mich, L., and Garigliano, R. 2002. "NL-OOPS : a requirements analysis tool based on natural language processing," in *Data Mining III*. Southampton, UK: WIT Press, pp. 321-330.

[Müller-Wienbergen et al. 2011]
Müller-Wienbergen, F., Müller, O., Seidel, S., and Becker, J. 2011. "Leaving the Beaten Tracks in Creative Work–A Design Theory for Systems that Support Convergent and Divergent Thinking," *Journal of the Association for Information Systems* (12:11), pp. 714-740.

[Natt och Dag et al. 2002]
Natt och Dag, J., Regnell, B., Carlshamre, P., Andersson, M., and Karlsson, J. 2002. "A Feasibility Study of Automated Natural Language Requirements Analysis in Market-Driven Development," *Requirements Engineering* (7:1), pp. 20-33.

[Natt och Dag et al. 2004]
Natt och Dag, J., Gervasi, V., Brinkkemper, S., and Regnell, B. 2004. "Speeding up Requirements Management in a Product Software Company : Linking Customer Wishes to Product Requirements through Linguistic Engineering," in *Proceedings of the 12th IEEE International Requirements Engineering Conference*, pp. 283-294.

[Neill and Laplante 2003]
Neill, C. J., and Laplante, P. A. 2003. "Requirements Engineering: The State of the Practice," *IEEE Software* (20:6), pp. 40-45.

[Niehaves 2007]
Niehaves, B. 2007. "On Epistemological Pluralism in Design Science," *Scandinavian Journal of Information Systems* (19:2), pp. 93-104.

[Norman 1988]
Norman, D. 1988. *The Psychology of Everyday Things*. 1st Edition. New York, USA: Basic books.

[Norman and Draper 1986]
Norman, D., and Draper, S. W. 1986. *User centered system design; new perspectives on human-computer interaction*. 1st Edition. Hillsdale, USA: L. Erlbaum Associates Inc.

[Nunamaker et al. 1990]
Nunamaker, J. F., Chen, M., and Purdin, T. 1990. "Systems Development in Information Systems Research," in *Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences*, pp. 631-640.

[Omoronyia et al. 2010]
Omoronyia, I., Sindre, G., Stålhane, T., Biffl, S., Moser, T., and Sunindyo, W. 2010. "A Domain Ontology Building Process for Guiding Requirements Elicitation," in *Requirements Engineering: Foundation for Software Quality*. 2010 Edition. Berlin / Heidelberg, Germany: Springer, pp. 188-202.

[Ossher et al. 2009]
Ossher, H., Amid, D., Anaby-Tavor, A., Bellamy, R., Callery, M., Desmond, M., De Vries, J., Fisher, A., Krasikov, S., Simmonds, I., and Swart, C. 2009. "Using tagging to identify and organize concerns during pre-requirements analysis," in *Proceedings of the 2009 ICSE Workshop on Aspect-Oriented Requirements Engineering and Architecture Design*, pp. 25-30.

[Palmer 2000]
Palmer, D. 2000. "Tokenisation and Sentence Segmentation," in *Handbook of natural language processing*. 1st Edition. CRC Press, pp. 11-36.

[Parasuraman et al. 2001]
Parasuraman, R., Sheridan, T. B., and Wickens, C. D. 2001. "A model for types and levels of human interaction with automation," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* (30:3), pp. 286-297.

[Parikh et al. 2001]
Parikh, M., Fazlollah, B., and Verma, S. 2001. "The Effectiveness of Decisional Guidance : An Empirical Evaluation," *Decision Sciences* (32:2), pp. 303-331.

[Park et al. 2000]
Park, S., Kim, H., Ko, Y., and Seo, J. 2000. "Implementation of an efficient requirements-analysis supporting system using similarity measure techniques," *Information and Software Technology* (42:6), pp. 429-438.

[Patil et al. 1997]
Patil, R. S., Fikes, R. E., Patel-Schneider, P. F., McKay, D., Finin, T., Gruber, T., and Neches, R. 1997. "The DARPA knowledge sharing effort: progress report," in *Readings in agents*. 1st Edition. San Francisco, USA: Morgan Kaufmann Publishers Inc., pp. 243-254.

[Payne 1982]
Payne, J. W. 1982. "Contingent decision behavior.," *Psychological Bulletin* (92:2), pp. 382-402.

[Pedhazur and Schmelkin 1991]
Pedhazur, E., and Schmelkin, L. 1991. *Measurement, design, and analysis: An integrated approach*. Student Edition. Hillsdale, USA: Lawrence Erlbaum Associates, Inc.

[Peffers et al. 2007]
Peffers, K., Tuunanen, T., Rothenberger, M. a., and Chatterjee, S. 2007. "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems* (24:3), pp. 45-77.

[Pérez-González and Kalita 2002]
Pérez-González, H. G., and Kalita, J. K. 2002. "Automatically generating object models from natural language analysis," in *Proceedings of the 17th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications - OOPSLA '02*, pp. 86-87.

[Petersen and Wohlin 2010]
Petersen, K., and Wohlin, C. 2010. "The effect of moving from a plan-driven to an incremental software development approach with agile practices," *Empirical Software Engineering* (15:6), pp. 654-693.

[Pohl 2010]
Pohl, K. 2010. *Requirements Engineering: Fundamentals, Principles, and Techniques*. 2010 Edition. Berlin / Heidelberg: Springer.

[Pries-Heje et al. 2008]
Pries-Heje, J., Baskerville, R., and Venable, J. R. 2008. "Strategies for Design Science Research Evaluation," in *Proceedings of the ECIS 2008*. Paper 87.

[Pries-Heje and Pries-Heje 2011]
Pries-Heje, L., and Pries-Heje, J. 2011. "Agile & Distributed Project Management: A Case Study revealing why SCRUM is useful," in *Proceedings of the ECIS 2011*. Paper 217.

[Rago et al. 2011]
Rago, A., Marcos, C., and Diaz-Pace, J. A. 2011. "Uncovering quality-attribute concerns in use case specifications via early aspect mining," *Requirements Engineering* (18:1), pp. 67-84.

[Rajlich 2006]
Rajlich, V. 2006. "Changing the paradigm of software engineering," *Communications of the ACM* (49:8), pp. 67-70.

[Ramesh et al. 2007]
Ramesh, B., Cao, L., and Baskerville, R. 2007. "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal* (20:5), pp. 449-480.

[Rayson et al. 2000]
Rayson, P., Garside, R., and Sawyer, P. 2000. "Assisting requirements engineering with semantic document analysis," in *Proceedings of the RIAO 2000*, pp. 1363-1371.

[Regnell et al. 1998]
Regnell, B., Beremark, P., and Eklundh, O. 1998. "A market-driven requirements engineering process: Results from an industrial process improvement program," *Requirements Engineering* (3:2), pp. 121-129.

[Robertson and Robertson 2006]
Robertson, S., and Robertson, J. 2006. *Mastering the Requirements Process*. 2nd Edition. Boston, USA: Pearson Education.

[Robey et al. 2001]
Robey, D., Welke, R., and Turk, D. 2001. "Traditional, iterative, and component-based development: A social analysis of software development paradigms," *Information Technology and Management* (2:1), pp. 53-70.

[Robinson and Kalakota 2004]
Robinson, M., and Kalakota, R. 2004. *Offshore Outsourcing: Business Models, ROI and Best Practices*. 2nd Edition. Alpharetta, USA: Mivar Press.

[Salton and McGill 1986]
Salton, G., and McGill, M. J. 1986. *Introduction to Modern Information Retrieval*. New York, USA: McGraw-Hill Book Company.

[Sampaio et al. 2007]
Sampaio, A., Rashid, A., Chitchyan, R., and Rayson, P. 2007. "EA-Miner: Towards Automation in Aspect-Oriented Requirements Engineering," in *Transactions on Aspect-Oriented Software Development III*, Berlin / Heidelberg: Springer, pp. 4-39.

[SAP AG 2012]
SAP AG. 2012. "SAP Travel Management application" http://help.sap.com/printdocu/core/print46c/en/data/pdf/FITVPLAN/FITVGENERIC.pdf. Accessed on 01/02/ 2012.

[Sarnikar and Deokar 2009]
Sarnikar, S., and Deokar, A. 2009. "Towards a Design Theory for Process-Based Knowledge Management Systems," in *Proceedings of the ICIS 2009*, pp. 1-10.

[Sawyer et al. 2002]
Sawyer, P., Rayson, P., and Garside, R. 2002. "REVERE: Support for Requirements Synthesis from Documents," *Information Systems Frontiers* (4:3), pp. 343-353.

[Sawyer 2000]
Sawyer, S. 2000. "Packaged Software : Implications of the Differences from Custom Approaches to Software Development," *European Journal of Information Systems* (9:1), pp. 47-58.

[Scacchi 2002]
Scacchi, W. 2002. "Understanding the requirements for developing open source software systems," *IEE Proceedings Software* (149:1), pp. 24-39.

[Schreiber et al. 1994]
Schreiber, G., Wielinga, B., de Hoog, R., Akkermans, H., and Van de Velde, W. 1994. "CommonKADS: a comprehensive methodology for KBS development," *IEEE Expert* (9:6), pp. 28-37.

[Schreiber et al. 1999]
Schreiber, G., Akkermans, H., Anjewierden, A., Hoog, R. de D., Shadbolt, N. R., Velde, W. van V. de, and Wielinga, B. J. 1999. *Knowledge Engineering and Management: The Common KADS Methodology*. Cambridge, USA: The MIT Press.

[Sein et al. 2011]
Sein, M. K., Henfridsson, O., and Rossi, M. 2011. "Action Design Research," *MIS Quarterly* (35:1), pp. 37-56.

[Seresht et al. 2008]
Seresht, S. M., Ormandjieva, O., and Sabra, S. 2008. "Automatic Conceptual Analysis of User Requirements with the Requirements Engineering Assistance Diagnostic (READ) Tool," in *Proceedings of the Sixth International Conference on Software Engineering Research, Management and Applications*, pp. 133-142.

[Sharp et al. 2007]
Sharp, H., Rogers, Y., and Preece, J. 2007. *Interaction design: beyond human-computer interaction*. 2nd Edition. Chichester, USA: John Willey & Sons Ltd.

[Shibaoka et al. 2007]
Shibaoka, M., Kaiya, H., and Saeki, M. 2007. "GOORE : Goal-Oriented and Ontology Driven Requirements Elicitation Method," in *Advances in Conceptual Modeling – Foundations and Applications*. 2007 Edition. Berlin / Heidelberg: Springer, pp. 225-234.

[Sillitti et al. 2005]
Sillitti, A., Ceschi, M., Russo, B., and Succi, G. 2005. "Managing Uncertainty in Requirements : a Survey in Documentation-driven and Agile Companies," in *Proceedings of the 11th IEEE International Symposium on Software Metrics*, pp. 10-17.

[Silva and Ribeiro 2003]
Silva, C., and Ribeiro, B. 2003. "The Importance of Stop Word Removal on Recall Values in Text Categorization," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1661-1666.

[Silver 1988]
Silver, M. S. 1988. "User perceptions of DSS restrictiveness: an experiment," in *Proceedings of the Twenty-First Annual Hawaii International Conference on System Sciences*, pp. 116-124.

[Silver 1991]
Silver, M. S. 1991. "Decisional Guidance for Computer-Based Decision Support," *MIS Quarterly* (15:1), pp. 105-122.

[Simon 1957]
Simon, H. A. 1957. *Models of Man*. 1st Edition. New York , USA: Wiley.

[Simon 1969]
Simon, H. A. 1969. *The sciences of the artificial*. 1st Edition. Cambridge, USA: M.I.T. Press.

[Sommerville 2010]
Sommerville, I. 2010. *Software Engineering*. 9th Edition. Boston, USA: Addison-Wesley.

[Staab et al. 2001]
Staab, S., Studer, R., Schnurr, H.-P., and Sure, Y. 2001. "Knowledge Processes and Ontologies," *IEEE Intelligent Systems* (16:1), pp. 26-34.

[Standish 2009]
Standish. 2009. http://www.standishgroup.com. Accessed on 03/03/2012.

[Takeda and Veerkamp 1990]
Takeda, H., and Veerkamp, P. 1990. "Modeling Design Processes," *AI Magazine* (11:4), pp. 37-48.

[Tam et al. 1998]
Tam, R. C., Maulsby, D., and Puerta, A. R. 1998. "U-TEL: A Tool for Eliciting User Task Models from Domain Experts," in *Proceedings of the 3rd international conference on Intelligent user interfaces*, pp. 77-80.

[Tamir et al. 2008]
Tamir, D., Marcos, S., and Mueller, C. J. 2008. "An Effort and Time Based Measure of Usability," in *Proceedings of the 6th international workshop on Software Quality*, pp. 47-52.

[Tichy and Koerner 2010]
Tichy, W. F., and Koerner, S. J. 2010. "Text to software: developing tools to close the gaps in software engineering," in *Proceedings of the FSE/SDP workshop on Future of software engineering research (FoSER '10)*, pp. 379-384.

[Todd and Benbasat 1991]
Todd, P., and Benbasat, I. 1991. "An Experimental Investigation of the Impact of Computer Based Decision Aids on Decision Making Strategies," *Information Systems Research* (2:2), pp. 87-115.

[Todd and Benbasat 1999]
Todd, P., and Benbasat, I. 1999. "Evaluating the Impact of DSS , Cognitive Effort , and Incentives on Strategy Selection," *Information Systems Research* (10:4), pp. 356-374.

[Tuunanen 2003]
Tuunanen, T. 2003. "A new perspective on Requirements Elicitation Methods," *Journal of Information Technology Theory and Application* (5:3), pp. 45-72.

[Vaishnavi and Kuechler 2007]
Vaishnavi, V. K., and Kuechler, W. 2007. *Design Science Research Methods and Patterns: Innovating Information and Communication Technology*. New York, USA: Auerbach.

[Valusek and Fryback 1985]
Valusek, J., and Fryback, D. 1985. "Information requirements determination: obstacles within, among and between participants," in *Proceedings of the twenty-first annual conference on Computer Personnel Research*, pp. 103-111.

[van de Weerd et al. 2006]
van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., and Bijlsma, L. 2006. "Towards a Reference Framework for Software Product Management," in *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pp. 319-322.

[Vasey and Thayer 1987]
Vasey, M. W., and Thayer, J. F. 1987. "The Continuing Problem of False Positives in Repeated Measures ANOVA in Psychophysiology: A Multivariate Solution," *Psychophysiology* (24:4), pp. 479-486.

[Vlas and Robinson 2012]
Vlas, R. E., and Robinson, W. N. 2012. "Two Rule-Based Natural Language Strategies for Requirements Discovery and Classification in Open Source Software Development Projects," *Journal of Management Information Systems* (28:4), pp. 11-38.

[Voutilainen 2003]
Voutilainen, A. 2003. "Part-of-speech tagging," in *The Oxford handbook of computational linguistics*. Oxford University Press, pp. 219-232.

[Walls et al. 1992]
Walls, J. G., Widmeyer, G. R., and El Sawy, O. A. 1992. "Building an information system design theory for vigilant EIS," *Information Systems Research* (3:1), pp. 36–59.

[Wang and Benbasat 2009]
Wang, W., and Benbasat, I. 2009. "Interactive decision aids for consumer decision making in e-commerce: The influence of perceived strategy restrictiveness," *MIS Quarterly* (33:2), pp. 293-320.

[Weber 2004]
Weber, R. 2004. "The Rhetoric of Positivism versus Interpretivism : A Personal View," *MIS Quarterly* (28:1), pp. iii-xii.

[Wermter and Hahn 2006]
Wermter, J., and Hahn, U. 2006. "You can't beat frequency (unless you use linguistic knowledge)," in *Proceedings of the 21st International Conference on Computational Linguistics*, pp. 785-792.

[Wilson et al. 1997]
Wilson, W. M., Rosenberg, L. H., and Hyatt, L. E. 1997. "Automated analysis of requirement specifications," in *Proceedings of the 19th international conference on Software engineering (ICSE '97)*, pp. 161-171.

[Wixon et al. 1990]
Wixon, D., Holtzblatt, K., and Knox, S. 1990. "Contextual design: an emergent view of system design," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 329-336.

[Wu et al. 2006]
Wu, H., Zubair, M., and Maly, K. 2006. "Harvesting social knowledge from folksonomies," in *Proceedings of the seventeenth conference on Hypertext and hypermedia*, pp. 111-114.

[Yaniv 2004]
Yaniv, I. 2004. "The Benefit of Additional Opinions," *Current Directions in Psychological Science* (13:2), pp. 75-78.

[Zowghi and Gervasi 2003]
Zowghi, D., and Gervasi, V. 2003. "On the interplay between consistency, completeness, and correctness in requirements evolution," *Information and Software Technology* (45:14), pp. 993-1009.

Lebenslauf

Hendrik Meth

1995 - 2001	Universität Mannheim Diplomstudiengang Wirtschaftsinformatik, Abschluss: 11/2001 als Diplom-Wirtschaftsinformatiker
2002 - 2010	Robert Bosch GmbH und Icon GmbH Projektleiter, Produktmanager und Berater im Bereich Business Intelligence
2010 - heute	Universität Mannheim Doktorand am Lehrstuhl für Wirtschaftsinformatik IV, Prof. Mädche