Journal of the Association for Information Systems

Volume 16 | Issue 9 Article 2

9-25-2015

Designing a Requirement Mining System

Hendrik Meth
University of Mannheim, meth@es.uni-mannheim.de

Benjamin Mueller
University of Groningen, and Institute for Enterprise Systems at the University of Mannheim, b.mueller@rug.nl

Alexander Maedche
University of Mannheim, maedche@es.uni-mannheim.de

Recommended Citation
Meth, Hendrik; Mueller, Benjamin; and Maedche, Alexander (2015) "Designing a Requirement Mining System," *Journal of the Association for Information Systems*, 16(9), .
DOI: 10.17705/1jais.00408
Available at: https://aisel.aisnet.org/jais/vol16/iss9/2

Journal of the Association for Information Systems
JAIS

Research Article

Designing a Requirement Mining System

Hendrik Meth
University of Mannheim
meth@es.uni-mannheim.de

Benjamin Mueller
University of Groningen, and
Institute for Enterprise Systems at the
University of Mannheim
b.mueller@rug.nl

Alexander Maedche
University of Mannheim
maedche@es.uni-mannheim.de

Abstract

The success of information systems (IS) development strongly depends on the accuracy of the requirements gathered from users and other stakeholders. When developing a new IS, about 80 percent of these requirements are recorded in informal requirements documents (e.g., interview transcripts or discussion forums) using natural language. However, processing the resultant natural language requirements resources is inherently complex and often error prone due to ambiguity, inconsistency, and incompleteness. Thus, even highly qualified requirements engineers often struggle to process large amounts of natural language requirements resources efficiently and effectively. In this paper, we propose a design theory for requirement mining systems (RMSs) based on two design principles: (1) semi-automatic requirement mining and (2) usage of imported and retrieved knowledge. As part of an extensive design project, which led to these principles, we also implemented a prototype based on this design theory (REMINER). It supports requirements engineers in identifying and classifying requirements documented in natural language and allows us to evaluate the artifact's viability and the conceptual soundness of our design. The results of our evaluation suggest that an RMS based on our proposed design principles can significantly improve recall while maintaining precision levels.

Keywords: *Requirements Engineering, Requirements Mining Systems, Requirements Mining Productivity, Design Science Research, Design Theory, Advice Taking.*

* Jan Pries-Heje was the accepting senior editor. This article was submitted on 30th September 2014 and went through one revision.

Volume 16, Issue 9, pp. 799-837, September 2015

1. Introduction

The success of IS development is strongly dependent on the accuracy of the requirements gathered from users and other stakeholders (Appan & Browne, 2012; Hickey & Davis, 2004). Requirements that have been overlooked, misinterpreted, or incompletely specified can lead to high costs. Boehm and Basili (2001) estimate that a software problem's detection and removal after delivery is a hundred times more expensive than its correction during the requirements or design phases. Determining complete and correct software requirements is, therefore, extremely important. In practice, approximately 80 percent of requirements are recorded in informal, natural language requirements documents such as interview transcripts, discussion forums, and narrative scenarios (Mich, Franch, & Novi, 2004; Neill & Laplante 2003). Natural language is inherently powerful and expressive and, thus, appropriate for communication between a broad range of stakeholders and users (Casamayor, Godoy, & Campo, 2011). Nevertheless, although it appears to be appropriate to articulate and discuss requirements, severe problems can emerge when using natural language in specification documents because these documents might be ambiguous, inconsistent, and incomplete (Wilson, Rosenberg, & Hyatt, 1997). Moreover, it is almost impossible for subsequent development tools to directly interpret these documents. Accordingly, natural language requirements are usually transformed from such informal statements into more consistent, formal, and unambiguous representations (Tichy & Koerner, 2010). This translation requires identifying and classifying individual requirements, a process we refer to as requirement mining in this paper.

To understand why requirement mining requires our attention, note the trend in the IT industry to shift from custom software development toward product-based development (Xu & Brinkkemper, 2007). Product-based development often leads to software development being more remote from actual usage. Consequently, software is often designed and coded in the absence of a specific organizational context and vendors have to consider the "common denominator" in their potential future customers. We argue that this situation leads to the requirements mining process having an overall increased relevance because vendors need to ensure that their product meets as many of their potential customers' demands as possible for it to be marketable.

At the same time, and contrary to custom development, specialized software vendors generally design and develop software standardized around this "common denominator". More organizations and stakeholders are involved, particularly if compared to any one specific in-house development. Accordingly, the number of potentially relevant natural-language-based sources in the development process is growing, which makes requirement mining an even more critical task, particularly for software product vendors.

The fact that software vendors developing product software receive requirements in various forms, through multiple channels, and from different stakeholders helps illustrate this issue. In a B2B context (e.g., ERP software), for example, a product manager responsible for identifying and prioritizing requirements obtains these from internal (e.g., sales and marketing, service departments, or other development departments) and external stakeholders (e.g., different current and potential customers, or key user focus groups). Most of these requirements are embedded in unstructured texts, such as emails, support requests, and user research activities' transcripts. Similarly, in a B2C context (e.g., mobile apps), product managers receive new requirements for their software either directly through unstructured feedback and comments on the software (e.g., using app store's feedback function or support cases) or indirectly by tracking the discourse on their product on social media or in developer forums. In these cases, mining the requirements manually can be time consuming, error prone, and monotonous, especially if repeated multiple times when updates on previously existing unstructured sources become available (Ambriola & Gervasi, 2006; Huffman, Dekhtyar, & Sundaram, 2005). These problems lead to a low individual performance and, specifically, to a lower productivity of product managers doing requirement engineering. Consequently, when Mich et al. (2004) asked software engineers to name the two issues in their job they would like to do more efficiently, they found that "identify user requirements" topped the list (46%). When asked for solutions, the majority (69%)

chose “automation” as potentially most useful to improve their general day-to-day efficiency. This leads us to ask if software development tools could support requirement mining and, if so, how?

So far, multiple tools have been suggested to support requirement mining by means of technology (Casamayor, Godoy, & Campo, 2010; Cleland-Huang, Settimi, Zou, & Solc, 2007; Vlas & Robinson, 2012). Although previous studies on requirement mining systems (RMSs) have made great strides in technically developing such systems, few efforts have been made to systematically capture the prescriptive knowledge gained during the design process. The conceptual design’s codification and abstraction could significantly extend the requirement mining knowledge base, guide future research in this area, and support the development of relevant tools to support practitioners. Furthermore, existing RMSs have been mainly evaluated through simulations comparing the presented system’s results with a previously defined benchmark (e.g., theoretically defined key performance indicators or performance levels achieved by using previous solutions). Even though these evaluations allow precise measurements of absolute quality and performance criteria, they do not compare the presented system’s results with those achieved through manual discovery. Thus, we suggest that the question of whether an RMS improves a requirements engineer’s productivity has not yet been satisfactorily answered. Accordingly, we specifically address the following research question:

RQ: What design theory should guide the development of RMSs that make requirement mining more productive than manual requirement mining?

To answer this question, we 1) derive a conceptual RMS design based on knowledge drawn from theoretical and non-theoretical sources, 2) develop an artifact according to this design, and 3) test the design by evaluating the artifact to compare a requirements engineer’s system-supported mining productivity with manual discovery.

As a result, our work provides a conceptual RMS design that contributes to the IS literature because RMSs are an important design class that existing works have not adequately described. From a practical point of view, the study can help commercial providers of RMSs in designing their applications and help vendors with self-made solutions to further optimize their practices. Applied to commercial software development, the derived design prescriptions 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.

Following a design science approach, this paper proceeds as follows: in Section 2, we summarize the study’s and related work’s foundations. In Section 3, we introduce the project’s overall methodology and our design process. In Section 4, we discuss our main results and identify design requirements on the basis of both an iterative interplay of general knowledge and core theories and on expert opinion. Thereafter, we conceptualize generic design principles addressing these design requirements and map them to design features. These are then instantiated in a prototype. In Section 5, we use this prototype to measure its effects on requirement mining productivity. In Section 6, we integrate our findings into a design theory and discuss our contributions. In Section 7, we summarize the paper, reflect on its limitations, and provide an outlook on future research.

2. Foundations

2.1. Requirement Mining

In general, one can document software requirements in natural language (e.g., a narrative scenario), in models (e.g., UML models), or even figures (e.g., a drawn user interface mockup) (Pohl, 2010). In this paper, we focus on natural language requirements (NLRs). One can express NLRs in documents (e.g., informal requirement specifications, interview transcripts, workshop memos, and narrative scenarios) and in other resources (e.g., entries in issue tracking or test case management systems, support databases, and discussion forums) (Vlas & Robinson, 2012). Therefore, we use the term natural language requirements resource (NLRR) instead of the less comprehensive natural language document.

As we discuss in Section 1, NLRs are usually transformed from initially informal statements into a more consistent and unambiguous representation and often contain additional information about a requirement's category, or its interrelation with other requirements. In requirements engineering (RE) research, different terms describe this process as requirements elicitation (Castro-Herrera, Duan, Cleland-Huang, & Mobasher, 2009), or requirements analysis (Ambriola & Gervasi, 2006). Whatever the case, requirement mining is a sub-process thereof that comprises the individual requirements' identification and classification. We focus on these two steps in the following paragraphs.

In an NLRR, anything from single words (e.g., a particular data field) or an entire sentence (e.g., the description of a function) to a sequence of sentences (e.g., to specify a non-functional requirement) may represent a requirement. By mining the actual requirements contained in these items, requirement identification serves two main purposes: First, it separates text that describes requirements from that which is not relevant. Second, it delimits each requirement in the document, which results in multiple, individual requirement statements (Vlas & Robinson, 2012). Texts' differing degree of structure and preprocessing allows the amount of irrelevant content to vary widely. In open source software development, for example, requirements often need to be identified from forums containing thousands of lines of social communications, code segments, and slang, which are often not written in order to explicate requirements (Cleland-Huang et al., 2007). At the other end of the spectrum, requirements may be identified in already pre-processed, semi-structured use case descriptions containing requirements in a very condensed form. By ignoring or even eliminating non-relevant passages in an NLRR containing requirements, requirements identification concisely summarizes the source's relevant information.

Figure 1 illustrates this comprehensively by depicting two major requirement mining application scenarios and the subsequent need for requirements identification in IS development focusing specifically on software products. For example, product managers in a B2B environment face heterogeneous stakeholder groups and the corresponding challenge of their heterogeneous inputs for the development process. In a B2C environment, individual users' unstructured mass feedback challenges product managers. Independently of the actual application scenario, product managers are primarily faced with huge amounts of NLRs from different sources.

To further process individual requirements, these requirements need to be enriched with more information. Classifying requirements into distinct categories is a widely accepted way of enriching them with additional semantics (Casamayor et al., 2010; Cleland-Huang et al., 2007; Vlas & Robinson, 2012). By using requirements templates (e.g., the Volere requirements template[1]), one classifies requirements 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). A corresponding classification can simplify subsequent modeling activities or even be a prerequisite for them. Classified requirements can be grouped together to derive specific model types (e.g., a data model). In addition, a classification structure envisioned in a template can prevent certain software aspects (e.g., usability requirements) from being omitted.

[1] http://www.volere.co.uk/template.htm

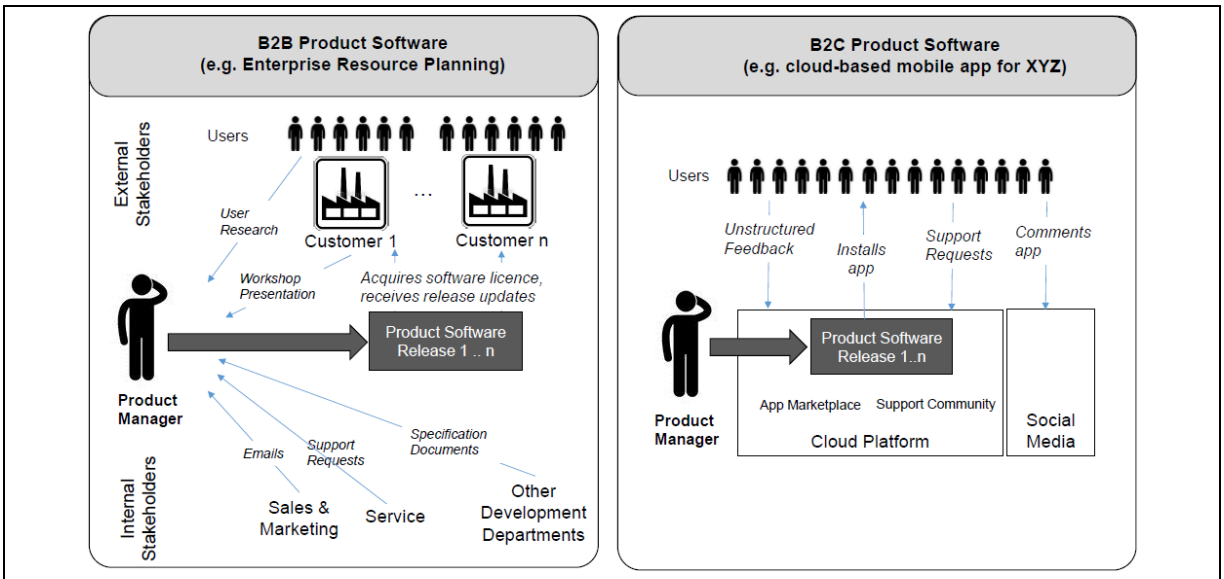


Figure 1. Requirement Sources in Product Software Development

2.2. System-Supported Requirement Mining

During the requirement mining process, a requirements engineer (or a system) needs to scan through the provided NLRR to identify and classify requirements. Two questions are repeatedly asked in respect of the processed texts: is this text passage, sentence, or word a requirement? And, if so, what kind of requirement?

Figure 2 depicts the basic steps of system-supported requirement mining. Once an NLRR is provided, the RMS pre-processes the NLRR so that requirements are identified and classified in a background process. This processing is based on a knowledge base on which the system draws. It results in proposed requirements. After this pre-processing, the requirements engineer drives an interactive approval process in which the engineer either approves or rejects the requirements proposed by the RMS. On a more abstract level, we suggest that this process can be seen as a series of consecutive decision tasks in which the RMS acts as an advice giver and the requirements engineer as the advice taker. In this analogy, assigning a text passage to a specific requirements category can be seen as a single decision task repeatedly performed throughout an NLRR. Decision making theory characterizes decision tasks according to multiple characteristics, such as 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 the advisor and the judge (low vs. high interaction) (Bonaccio & Dalal, 2006). Reflecting on the characteristics introduced above, RMSs' support of requirement mining can be characterized as a decision process comprising choice tasks (the assignment of distinct requirements categories) that a single advisor provides (the RMS) by following a solicited, but low, interaction.

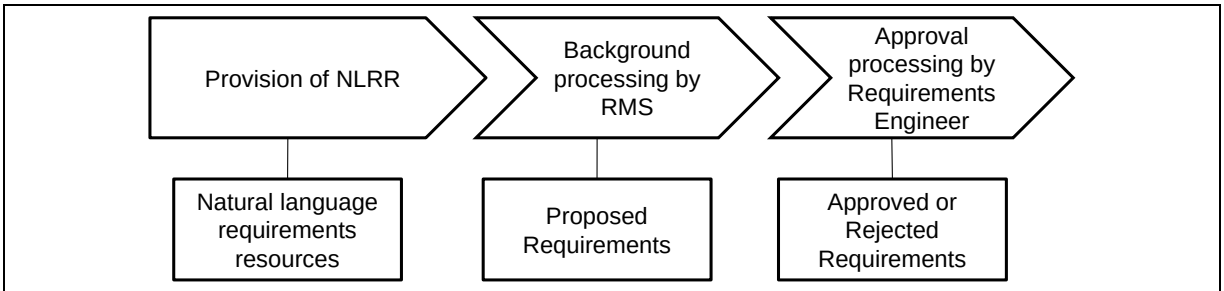


Figure 2. RMS-supported Requirement Mining Process

2.3. Existing Research on RMS and their Limitations

Most current RMSs identify requirements and classify them according to an existing taxonomy. To date, the literature has proposed different RMS designs. Cleland-Huang et al. (2007) focus on non-functional requirements (NFRs) (e.g., security, performance, and usability requirements). Based on the notion that each NFR sub-group has a unique set of keywords, the system uses different knowledge base items to find and classify NFRs from each subgroup. Casamayor et al. (2010) similarly aim at detecting NFRs and employ a semi-supervised categorization approach that only needs a small set of manually classified requirements to initially train the classifier. In their system, the classification model is iteratively enhanced on the basis of the users' feedback on the artifact's output. Rago, Marcos, and Diaz-Pace (2011) present QAMiner, a system that also aims at discovering NFRs. This system, however, analyzes use case specifications and relates requirements to pre-defined quality attributes (e.g., modifiability, performance, availability) to prevent these non-functional aspects from being understated in the resulting requirements specifications. Finally, Vlas and Robinson (2012) present an automated approach for identifying and classifying both functional and non-functional requirements in open source software projects' natural language feature requests.

Analyzing the described works in more detail reveals that researchers have investigated many different design choices. As we indicate earlier, and in contrast to this diversity in design, evaluations across these studies focus on simulations comparing the requirements that the system proposes with a previously defined benchmark. However, as we mention in Section 2.2, proposed requirements are not a requirement mining processes' final result but rather a preliminary step towards approved requirements. Owing to an NLRR's ambiguity and inconsistency, automated requirement mining results mostly require manual rework to correct the automatism's mistakes, to adapt its findings, or to add overlooked requirements (Cleland-Huang et al., 2007). Therefore, even automated approaches resulting in high-quality proposed requirements could require more overall effort than manual requirement mining if the rework effort is also considered. As a consequence, automated requirement mining does not necessarily outperform manual requirement mining even if it works efficiently. A study investigating whether using a respective system actually improves individual performance by comparing it to a manual approach could, therefore, complement current RMS work.

Furthermore, while the analyzed works include detailed descriptions of their specific prototypes, they lack the codifications and the abstractions of the demands that the system needs to fulfill and the concepts addressing each of these demands. A corresponding conceptualization has been intensively discussed in the design science research (DSR) literature (e.g., Baskerville & Pries-Heje, 2010; Gregor & Jones, 2007) and enables design approaches to be generalized beyond a description of a specific solution to a specific problem. Applying this approach to RMS, we suggest that the theoretical contributions drawn from previous works can be extended substantially.

Finally, the suggested systems lack clear theoretical grounds. 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 for a conceptualization with significant reach.

We address these gaps by 1) deriving a conceptual RMS design based on knowledge drawn from theoretical and non-theoretical sources, 2) developing a prototypical artifact according to this design, and 3) testing the design by evaluating said artifact in which we compare a requirements engineer's system-supported mining productivity with manual discovery.

3. Methodology

3.1. Overall Research Design

To contribute to closing these gaps, we conducted a research project following DSR's principles (Hevner, March, Park & Ram, 2004; March & Smith, 1995). Design is an established research approach in the IS field (compare, e.g., Gregor & Hevner, 2013; March & Smith, 1995; Nunamaker,

Chen, & Purdin, 1990; Simon, 1969, Walls, Widmeyer, & El Sawy, 1992; Winter 2008). Its key characteristic is that it "seeks to extend the boundaries of human and organizational capabilities by creating new and innovative artifacts" (Hevner et al., 2004, p. 75). DSR subsequently views these artifacts as something artificial (i.e., skillfully constructed by humans for a specific purpose) that intervenes in "complex social processes…by introducing changes into these processes" (Baskerville, 1999, p. 4). Hevner et al. (2004) identify actual software prototypes (among other things) as artifacts that incorporate some form of conceptual understanding to help solve a problem in practice. In the context of our work, one of DSR's key characteristics is its proposed interplay between abstract theoretical knowledge and context-specific practical knowledge. Since Hevner et al. (2004) re-emphasized DSR in the IS field, intense scholarly discussions on this issue have produced several influential contributions. In the context of our work, we build on these discussions in two ways: the role of theory in the design process and the design of the general research process.

With regard to the role of theory, we draw on Gregor's and Jones' (2007) suggestion that abstract theoretical knowledge serves a dual purpose in DSR. First, it can serve as an input for a design cycle and represent a form of kernel theory, which corresponds to the idea that the scientific knowledge base informs design in general (Hevner et al., 2004). Second, an abstract understanding of what has been learnt from designing an artifact can serve as a blueprint for building similar artifacts in the future (Piirainen & Briggs, 2011). Like a blueprint, this design theory documents how an artifact should be built to achieve the desired interventions and outcomes. Both these observations resonate strongly with gaps 1 and 3 as discussed above.

With regard to the general research process' design, DSR literature suggests that artifacts should be designed iteratively to enable the design results' continuous reflection and incremental refinement (Hevner et al., 2004; Takeda & Veerkamp, 1990). Building on and refining this general guideline, several influential scholars have suggested models for conducting design-oriented research (e.g., Nunamaker et al., 1990; Hevner 2007; Peffers, Tuunanen, Rothenberger, & Chatterjee, 2008; Vaishnavi & Kuechler, 2007). Our research design is similar to that of Vaishnavi and Kuechler (2007). Their framework supports our efforts to address the gaps identified above in three respects. First, the "awareness of problem" phase enables a tight integration of theoretical (i.e., literature) and non-theoretical (i.e., expert opinion) input. Second, the framework supports close collaboration with potential users to ensure viability and to improve the positive impacts on their productivity. While these respects apply throughout the process, the evaluation and conclusion phases Vaishnavi and Kuechler (2007) propose are specific examples of such impacts. In this regard, we also extend the framework that Vaishnavi and Kuechler (2007) present by drawing on Peffers et al. (2008) through specifically distinguishing between demonstration and evaluation. In the former, the design scientist demonstrates the artifact to subject-matter experts from the problem's domain (i.e., requirement engineers in our case) to capture their feedback. Then, in the latter, the artifact is evaluated in a suitable context to measure its effectiveness and efficiency. Third, the multiple, iterative cycles that Vaishnavi and Kuechler (2007) and Peffers et al. (2008) suggest support constant reflection and abstraction, which we believe are an essential prerequisite to develop an abstract theoretical design—a blueprint of sorts—rather than just a specific solution.

3.2. Design Cycles

Building on these theoretical and procedural considerations, two design cycles comprising a sequence of six phases each characterize our research design. Figure 3 depicts the resultant iteration between conceptualization, development, and evaluation.

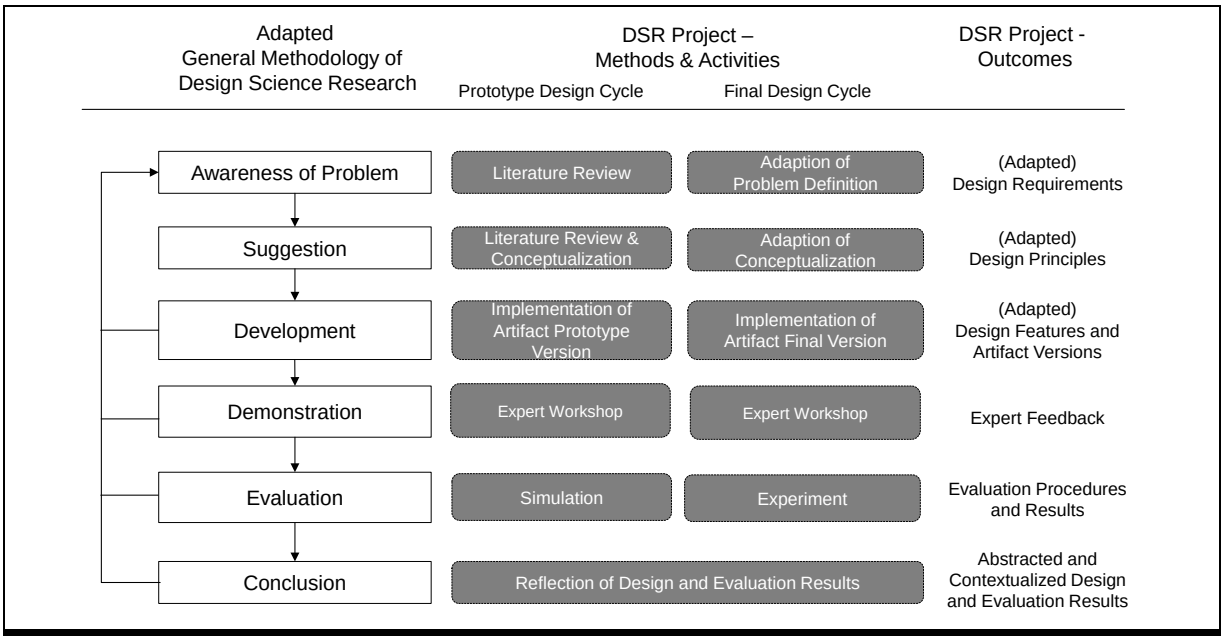


Figure 3. Our Design Science Research Approach

An intensive literature review to create problem awareness initiated the first of these two cycles, the prototype design cycle, which resulted in tentative design requirements for the artifact to be built. A study of the usability concept in software engineering, particularly for product software (Scheiber et al., 2012), triggered our investigation of this topic. In this study, the need to build advanced requirements engineering capabilities surfaced as one of the main challenges, specifically for small and medium-sized software companies. Guided by this initial problem awareness, the first literature review helped us derive tentative requirements, which we used to specify usage scenarios and personas for the envisioned artifact (Meth et al., 2013a). Based on these design requirements, we conducted a second literature review to identify general knowledge and theories that we could apply to address the identified problems. Using this knowledge, we conceptualized preliminary design principles in the suggestion phase. We then mapped these design principles to design features that we implemented in REMINER—a prototype version of the artifact—during the development phase. To collect feedback on the artifact's usefulness, we presented it to requirements engineering experts in various demonstration sessions. We traced the feedback they provided back to the related design decision and design principle to adapt the preliminary design principles. Subsequently, we analyzed the prototype in a quantitative simulation that focused on the interplay of the two main design principles. In the conclusion phase, we presented this evaluation's results (i.e., the preliminary design, the artifact's prototype version, and the simulations' results) at a leading software engineering conference to gather feedback, check for (theoretical) viability, and ascertain whether the attending practitioners and academics thought the artifact addresses and actually solves important issues in practice (Meth et al., 2013b).

In the second cycle—the final design cycle—we adapted the initial problem definition and conceptualization based on the first cycle's design, demonstration, and evaluation results. This led to our adjusting the initial design requirements and design principles. We again mapped the adapted design principles to design features, which resulted in our modifying the artifact we used to instantiate our design (REMINER). To improve the artifact's user-friendliness, we presented it to usability experts in various demonstration sessions (comparable to those in the prototype design cycle), which resulted in multiple small adaptations. Subsequently, we evaluated the final artifact version in a lab experiment conducted with students and in a replication of the experiment in a field environment involving experts. In these experiments, we measured the effects of each design principle on individual requirements engineers' productivity. Finally, we again abstracted and synthesized the design and evaluation results into a design theory for RMS.

We based our conceptualization on three design steps applied consecutively in both iterations (Hevner et al., 2004; March & Smith, 1995). First, based on the input from the awareness and suggestion phases, we identified RMS design requirements. Our understanding of design requirements is closely associated with the meta-requirements concept, which Walls et al. (1992) describe, and with general requirements, which Baskerville and Pries-Heje (2010) depict. In this sense, the design requirements comprise generic requirements that any artifact instantiated from this design should meet. In the second step, we derived generic design principles. We did so by, for example, drawing on decision support theory. Such design principles are closely related to the meta-design as Walls et al. (1992) describe and general components as Baskerville and Pries-Heje (2010) describe. Consequently, we regard design principles as an artifact's generic capabilities corresponding to the proposed design through which it addresses its requirements. Finally, we mapped design principles to specific design features, which we implemented in an expository instantiation. In the context of this paper, design features are specific ways to implement a design principle in an actual artifact. While design principles abstract from technical specifics, design features close this last step of conceptualization. In our work, we implemented these design features in the prototype that we used in the demonstration and evaluation phases.

This mapping of design principles to design features also supports the evaluation of the resultant prototype. As Kuechler and Vaischnavi (2012) highlight, the design principles for RMSs we suggest can be interpreted as explanatory statements. These statements, in turn, help explain why a prescribed action (i.e., a design feature instantiating one of these design principles) leads to a specific goal. This will not only guide our evaluation (e.g., by identifying relevant dependent variables) but also help us to generalize and abstract the evaluation's results into a better understanding of the conceptual underpinning of the design we propose.

3.3. Ensuring Grounding and Viability

As we discussed earlier (see Figure 3), our research design contains multiple instances during which we can ensure the proposed design's grounding and viability. In accordance with the processes that Vaishnavi and Kuechler (2007) and Peffers et al. (2008) propose, we used the fourth and fifth phases of each iteration (i.e., demonstration and evaluation) to ensure our design's grounding and viability. In terms of the demonstrations, we conducted a series of workshops with experts who focused on issues such as of design's viability and appropriateness or the prototype's usability. In terms of the evaluation, we conducted two distinct evaluations: an interim evaluation at the end of the prototype design cycle and an ex post evaluation (Pries-Heje, Baskerville, & Venable, 2008) at the end of the final design cycle. We briefly introduce the demonstrations and evaluations here but emphasize the ex post evaluation later because it evaluates the tentative end product of our research efforts to date and the design we propose in this paper.

To ensure the design's grounding and relevance in practice, we organized a total of seven workshops in each iteration, each comprising one to four subject-matter experts (i.e., requirement mining and usability) and two researchers (an overall number of 11 evaluators participated). All of the participants had extensive experience (avg. 9.7 years). The workshops lasted for about 1.5 hours and comprised three parts: a pre-questionnaire, the presentation of the prototype, and its evaluation. We asked the participating experts for specific feedback and traced each of the feedback items they provided back to the related design decision and design principle to adapt the preliminary design principles. The demonstration sessions in the prototype design cycle focused on the artifact's functional evaluation. In the final design cycle, the demonstration focused on usability[2].

To test the viability of the design's functionality, we conducted a series of simulations at the end of the prototype design cycle. In these simulations, the prototype processed several predefined NLRRs. We then compared the prototype's initial assessment to a sample solution (benchmark) that we derived from experts' assessment of the same NLRRs. Thus, we focused the interim evaluation on the automatism's performance and the initial results it provided[3].

[2] Meth et al. (2012a) provide details on the expert workshops.
[3] For detailed results of the simulation study, please see Meth et al. (2012b).

As we describe in Section 2.3, previous research on RMS evaluations focus solely on simulations and comparing the corresponding systems' results with a previously defined benchmark. Although simulations allow precise measurements of dependent variables in a controlled setting, they do not incorporate human interaction. Real requirements engineers are supposed to use an RMS to solve real problems in the real world. Consequently, the evaluation of a design should also involve actual interaction with the artifact to compare the system-supported requirement mining outcomes with the as-is situation of manual discovery. Therefore, we decided to add an experimental evaluation at the end of the final design cycle as Hevner and Chatterjee (2010) suggest. In an experiment, design principles can be accurately adjusted and their impacts on requirement mining productivity can be measured while controlling for potential influential factors (e.g., requirement mining knowledge or motivation).

4. Designing a Requirement Mining System

4.1. Developing Design Requirements

A design scientist needs to understand the general goals associated with the requirement mining process to derive specific RMS design requirements. The process's generalization and abstraction to a series of decision making tasks, which we introduce above, provide an approach with which 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 have been proposed to optimize decision quality (Wang & Benbasat, 2009). However, in addition to decision quality, the idea that cognitive effort considerations also influence decision making has been discussed since Simon's (1957) work. While he discusses cognitive efforts as a limitation leading to suboptimal decision results, cognitive efforts have been found to also influence the decision strategy choice. Researchers have often explained such decision strategy selection by using contingency models in which a cost and benefit tradeoff determines strategy choice (Beach & Mitchell, 1978; Payne, 1982). According to these models, decision makers follow the dual goal of maximizing decision quality and, simultaneously, minimizing their cognitive effort.

To optimize this tradeoff, researchers have proposed different types of decision support systems (DSSs) (Silver, 1991) and investigated DSS usage's effects on decision behavior (e.g., Todd & Benbasat, 1991, 1999). One uses DSS to improve decision results via its advice[4], which builds on the idea that high-quality advice will result in high-quality decisions (Gardner & Berry, 1995; Yaniv, 2004). Ideally, the cognitive effort required will simultaneously decrease because the DSS prepares the decision and the relevant information for the decision maker. However, while a DSS can improve decision quality and reduce the cognitive effort required, the system may also restrict users' decision behavior, which is termed system restrictiveness (Silver, 1988). System restrictiveness is the extent to which the DSS pre-selects decision strategies and, therefore, offers decision makers only a limited choice of strategies, which may not include their preferred ones (Silver, 1988). Therefore, when implementing decision aids, designers need to ensure that its benefits (e.g., the reduced cognitive effort) are not invalidated by its restrictions.

Table 1 summarizes human decision makers' goals and the DSS design requirements that address them. Most importantly, the perceived advice quality, perceived cognitive effort, and perceived restrictiveness are important features of any DSS. For example, Wang and Benbasat (2009) conducted an experiment with decision aids as components of e-commerce platforms and found that all three factors had significant effects on the intention to use a decision aid.

[4] Most studies define advice as a type of recommendation from an advisor that favors a particular option (Bonaccio & Dalal, 2006).

Table 1. Human Decision Makers' Goals and DSS Design Requirements	
Human decision makers' goals	**DSS design requirements**
Maximize decision quality	Increase decision quality by providing advice with high advice quality
Minimize cognitive effort	Reduce human decision maker's cognitive effort by providing decision support
Maintain control over decision strategy selection	Minimize system restrictiveness by allowing users to control the strategy selection

4.2. Applying Theoretical Concepts to Requirement Mining

We transfer the above insights to developing design requirements for RMS. In our case, we expect the requirements quality that an RMS proposes to determine the requirements quality that the requirements engineer finally approves. As we introduce earlier, RMSs require a knowledge base to identify and categorize proposed requirements. In general, the quality of requirements that an RMS propose mainly depends on the contents of the knowledge base used for the background mining process (Casamayor et al., 2010; Cleland-Huang et al., 2007). Researchers have found an extensive knowledge base with correctly classified requirements to result in high-quality proposed requirements (Casamayor et al., 2010; Cleland-Huang et al., 2007). Beyond a focus on high-quality proposed requirements implemented in earlier RMSs (Gacitua, Sawyer & Gervasi, 2011; Goldin & Berry, 1997; Kiyavitskaya & Zannone, 2008), we suggest that only an increase in the approved requirements' quality will address requirements engineers' goal of achieving a high-decision quality. Consequently, we derive the following design requirement:

> **DR1:** ***Increase the quality of approved requirements.*** *The requirement mining process should be supported by systems that improve the quality of approved requirements.*

To reduce requirements engineers' cognitive effort during this requirement mining process, we need to understand which phases of this process depend on human cognition. Most RMSs 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 effort of transforming the proposed requirements into approved requirements is the only determinant of the requirements engineer's cognitive effort. In some cases, this might still involve intensive reflection. However, in most cases, the cognitive efforts will be reduced from an active consideration of all the decision options to a rather reactive approval of the given advice.

Taking a holistic view of the requirements engineers' cognitive effort, besides the actual decision making process, the manual efforts required to create and maintain the knowledge base should also be considered and be minimized. Consequently, we derive the following design requirement:

> **DR2:** ***Decrease the cognitive efforts required to execute and prepare requirement mining.*** *The requirement mining process should be supported by systems that decrease the cognitive effort needed to transform the proposed requirements into approved requirements and the cognitive efforts needed to create and maintain the underlying knowledge base.*

Finally, to minimize restrictiveness, RMSs seem to offer two main dimensions to allow for strategy selection. First, they 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 using the system in a fully automated mode (Gacitua et al., 2011; Goldin & Berry, 1997; Kiyavitskaya & Zannone, 2008). To limit system restrictiveness, RMSs should allow requirement engineers enough flexibility to choose an appropriate type of support.

Furthermore, system restrictiveness should also be limited in regards to the knowledge to be used during requirement mining. RMSs can use different types of knowledge (e.g., imported knowledge vs. dynamically retrieved knowledge). To limit system restrictiveness, different types of knowledge should be usable during requirement mining. Consequently, we derive the following design requirement:

> **DR3:** ***Limit system restrictiveness during requirement mining.*** *The requirement mining process should be supported by systems that support minimal processing restrictions for conducting requirement mining.*

4.3. Translating Design Requirements into Design Principles

To address the design requirements formulated in Section 4.2, we rely on our earlier abstraction of the requirement mining process to a general decision making process, with which we drew an analogy between RMSs and DSSs. This analogy can also be used to introduce types of decisional guidance implemented in RMSs. In doing so, we draw on Silver (1991) and his description of decisional guidance (DG) as the way any DSS (i.e., also RMS) informs or influences decision makers in structuring and executing decision tasks. He defines a typology of DG based on three different characteristics: targets, forms, and modes of guidance.

4.3.1. Targets of RMSs' Guidance

Silver (1991) identifies two kinds of DG. First, DG to structure the decision making process helps decision makers select the right approach, method, or strategy to make a decision. For example, structural guidance could support one in choosing an existing decision strategy, such as "additive compensation" or "elimination by aspects"[5]. Second, executional guidance can help decision makers conduct the decision task operationally. For example, the system could prompt the user to enter values or calculate an alternative's overall value.

When contemplating the targets of RMSs' guidance, revisiting the actual process to be conducted is worthwhile. Requirement mining, as previously explained, can be regarded as a series of consecutive decision tasks in which assigning a text passage to a specific requirements category represents a single decision task performed repeatedly. Although this task requires substantial knowledge of requirements engineering and the corresponding business domain, it is a standardized procedure and executed rather similarly each time it is performed.

This seems to resonate with DR2 and, ultimately, DR1. While requirement mining hardly requires support to structure the decision task, it does require execution support to reduce requirements engineers' cognitive efforts while maintaining a high level of quality. This is particularly true in light of the large number of decisions to be made.

4.3.2. Forms of RMSs' Guidance

Silver (1991) suggests that DG might be implemented in a suggestive or informative way. Suggestive guidance advises decision makers on which strategy to choose or which values to enter. Conversely, informative guidance provides decision makers only 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.

An empirical study by Parikh, Fazlollah, and Verma (2001) provides important insights to determine appropriate forms of guidance. In an experimental study involving 141 participants, the authors investigate how different forms of DG influence decision quality and decision efficiency. They asked participants to examine a historical data set and identify its key characteristics. Based on the identified characteristics, they had to assign a suitable forecasting model to process this data. This decision task's basic constituents (identification of decision-relevant information and its subsequent

[5] 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. Eliminating aspects is a strategy based on comparing attribute values with threshold values. Alternatives are eliminated if one of the attributes does not meet a threshold

classification) resemble the decisions involved in the requirement mining process. Parikh et al. (2001) find that suggestive guidance outperforms informative guidance in respect of the decision quality and the decision efficiency.

These two variables (decision quality and decision efficiency) can be associated with the design requirements DR1 and DR2 we propose above. Revisiting our DSS-RMS analogy, increased decision quality is associated with approved requirements' increased quality and increased decision efficiency can be associated with a decrease in mining efforts. Therefore, we expect suggestive guidance to be an appropriate means of addressing DR1 and DR2.

4.3.3. Modes of RMSs' Guidance

Finally, Silver (1991) distinguishes different modes of guidance that describe the ways DG is generated. DG can be predefined, dynamic, or participative. Predefined guidance comprises context-specific information or recommendations that experts or regular users define upfront and import into a then-static knowledge base. In contrast, dynamic guidance is an adaptive mechanism that generates information and recommendations based on actual usage. DG (similarly to RMSs) usually uses knowledge bases to generate advice. Dynamic guidance builds up additional knowledge base content iteratively. Finally, participative guidance places a stronger emphasis on users' participation in determining guidance-specific content. For instance, in a decision task based on a decision table with different alternatives, participative guidance could be implemented by adding a functionality to manipulate the table by means of ordering or summation. In the next steps of our design, the presented types of guidance are associated with the requirement mining process and the identified design requirements. To this end, Parikh et al. (2001) also analyzes how different modes of guidance affect decision quality and decision efficiency. Their central finding is that dynamic guidance outperforms predefined guidance in terms of decision quality and decision efficiency.

Analogous to the argumentation regarding the form of guidance in which we associated decision quality and decision efficiency with DR1 and DR2, we expect dynamic guidance to result in the approved requirements' increased quality and in a decrease in mining efforts. Parikh et al. (2001) investigate different modes of guidance as exclusive alternatives. However, dynamic, predefined, and participative guidance can also be combined to improve results. We suggest that, when applied complementary to dynamic guidance, predefined and participatory guidance can provide additional advice and, thereby, further increase decision quality and decision efficiency.

Furthermore, revisiting the design requirement DR3, additionally applied participative guidance can provide final decision makers with a higher degree of freedom, which might reduce their perceived system restrictiveness. Therefore, in the context of requirement mining, we propose the complementary use of different modes of guidance. Figure 4 summarizes which forms of DG can help address which design requirements.

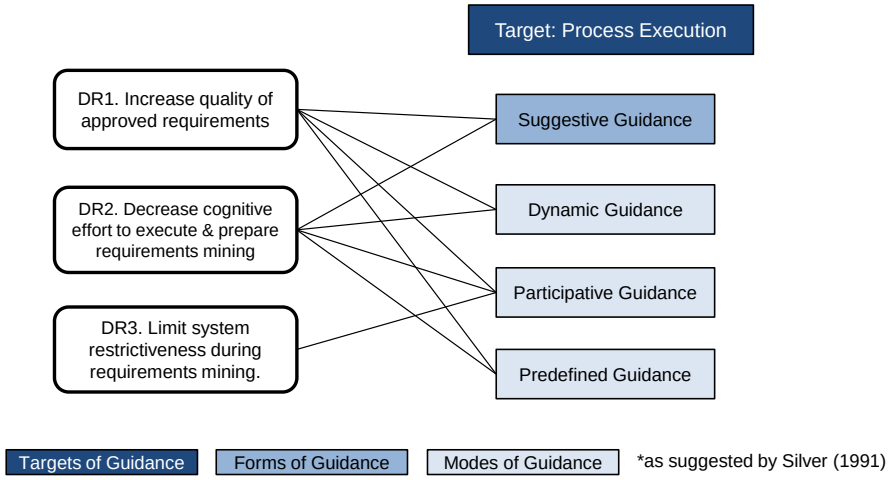


Figure 4. Associating Design Requirements with Types of Decisional Guidance

4.4. Deriving Design Principles of Requirement Mining Systems

The analogy built above helped us develop design principles that will enable RMSs to meet the design requirements we developed. In the context of requirement mining, suggestive guidance can be accomplished via automation, which results in the automation algorithm proposing a set of requirements. When mining requirements from NLRRs, a requirements engineer analyzes a text to identify relevant words and assign them to requirements categories. This process can be decomposed into single steps that are performed repeatedly and follow specific rules (Ambriola & Gervasi, 2006). Consequently, they can be translated into algorithms, which a computer can execute automatically. Automation addresses the first two design requirements that we identify in Section 4.2. First, automation can increase the approved requirements' quality. Reflecting the analogy to decision making, the proposed requirements' quality is expected to positively affect the approved requirements' quality. A carefully developed algorithm can identify a significant percentage of the requirements in a natural language document and those requirements that may have been overlooked in a purely manual discovery process (Berry, Gacitua, Sawyer & Tjong, 2012). Moreover, because the algorithm will not suffer fatigue or decreasing motivation as a human might, each part of a document will be given equal attention. This can additionally contribute to a more complete set of requirements. Second, automation should lead to a decrease in cognitive efforts because the requirements engineer does not need to manually identify and categorize each automatically classified requirement.

During the proposed requirements' manual approval, the requirements engineer decides whether to follow the RMS's advice or not. In the case of requirement mining, the NLRR's ambiguity and inconsistency often require a third option: requirements need to be adapted or added. In these cases, a functionality supporting manual discovery needs to complement the automatism (Berry et al., 2012; Kiyavitskaya & Zannne, 2008). However, any manual adaptation of automatically identified requirements is an 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. Beside the effects on DR1 and DR2, capabilities that support a manual identification and classification of requirements are also a way to enable participative guidance. Allowing the requirements engineer further freedom in the mining process can, thus, also minimize the system restrictiveness (DR3). Consequently, we propose the following design principle:

DP1: *Semi-automatic requirement mining:* *RMSs should support efficient automatic and manual requirement mining in NLRR.*

As we illustrated in Section 2.2, automated requirement mining requires an underlying knowledge base containing terms and a categorization of these terms. Relating the design requirements we developed in Section 4.2 to knowledge creation, a corresponding design principle should provide answers to the following questions: 1) how can knowledge quality be increased and 2) how can the requirements engineer's (cognitive) efforts to create knowledge be decreased?

Starting with the first question, the completeness and correctness of the knowledge base are aspects that can be used to assess the knowledge base's quality. A more extensive knowledge base will only lead to better mining results if a sufficient level of correctness is maintained. Supplementing domain-specific knowledge is an approach to augment the knowledge base with relevant knowledge. Documents that originate from the same domain share specific requirements elements not included in general knowledge (Lemaigre, García, & Vanderdonckt, 2008) (e.g., the data field "frequent flyer number" in the domain "traveling"). Similarly, single project's or entire organization's specific writing styles or standards can result in the need to extend imported knowledge (Cleland-Huang et al., 2007). In the context of software development (and the development of standardized software products in particular), another opportunity to provide domain-specific knowledge is drawing on existing documentation of a given product's previous versions or releases.

Beyond such predefined, imported knowledge, there are also additional ways to generate domain-specific knowledge. Addressing the design requirement behind the second question, the proposed design should support knowledge generation so that it minimizes the requirements engineer's effort. Therefore, besides predefined guidance, a mechanism is needed to support dynamic guidance. This can be realized by feeding back previous requirement mining activities' results into the knowledge base and, thereby, creating and using retrieved knowledge in addition to imported knowledge. Although this process requires some supervision to maintain the quality, this type of knowledge supplementation should be a lot more efficient than manually creating domain-specific knowledge. Consequently, we consequently propose the following design principle:

DP2: *Use imported and retrieved knowledge:* *RMSs should use manually imported and automatically retrieved knowledge during automatic mining.*

Figure 5 overviews the conceptualization process from the design requirements via the types of DG to the design principles. The figure shows how different types of DG can address the identified RMS design requirements. Furthermore, it outlines which RMS design principle is associated with which type of DG.

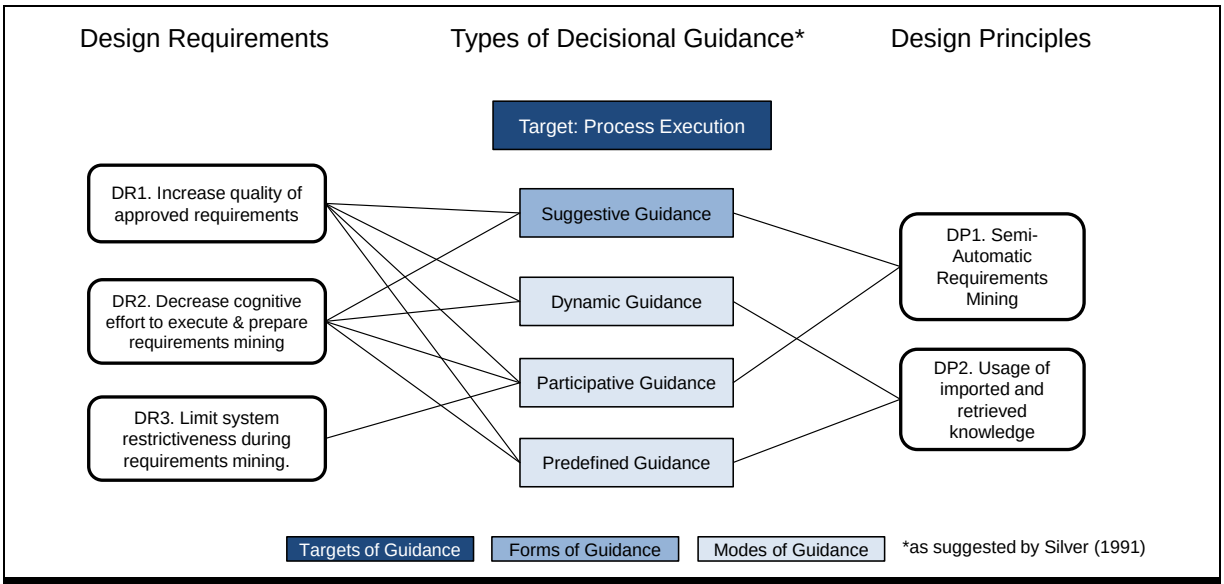


Figure 5. Deriving Design Principles from Design Requirements

4.5. Mapping Design Principles to Design Features

In the final step of the conceptualization, we map the identified design principles to design features. Design features are specific artifact capabilities to satisfy design principles (e.g., the algorithm chosen for automatic mining). Figure 6 introduces the design of the artifact features used to develop a prototype—REMINER. These features are based on the design requirements and design principles we develop above. One requires two types of functionalities to implement the first design principle (semi-automatic requirement mining): features for automatic and manual mining. Similarly to former approaches, automatic mining (DF2) has been implemented using common information retrieval techniques (Casamayor et al., 2010; Cleland-Huang et al. 2007; Vlas & Robinson, 2012). However, information retrieval requires individual words (instead of entire NLRRs) and further knowledge of each word (e.g., its word class). Therefore, one needs to preprocess NLRRs (DF1) via natural language processing. To improve the approved requirements' quality beyond the automatic mining, we include a manual mining functionality (DF3). To keep the additional manual mining efforts as low as possible, ease of use was an important requirement for us in designing this feature.

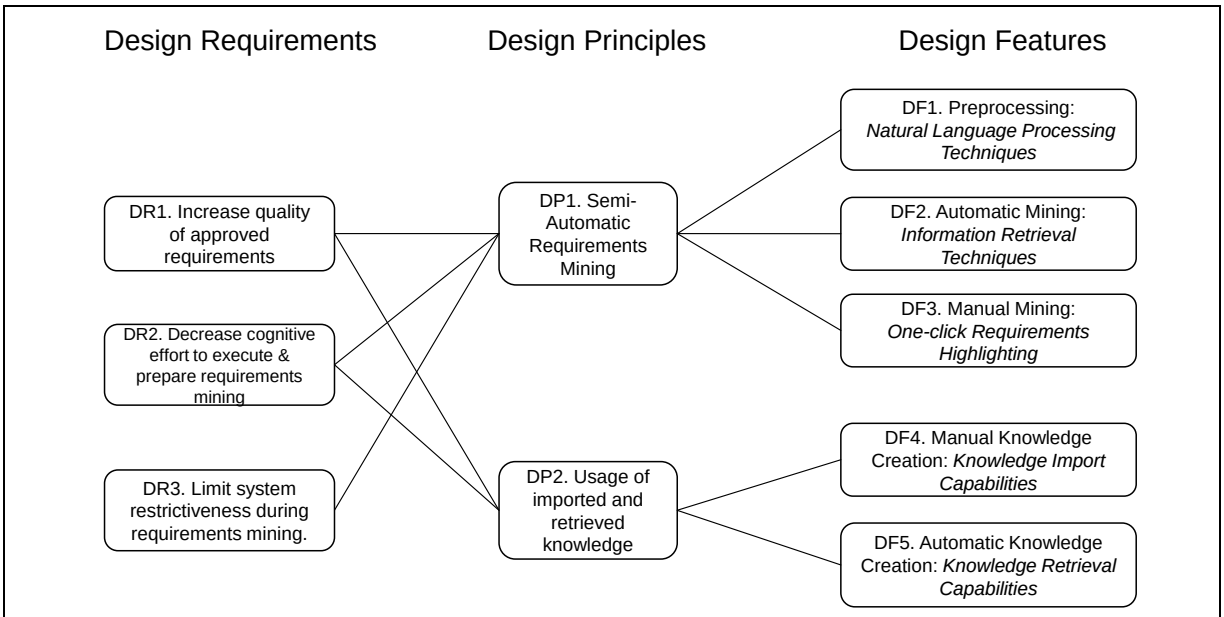


Figure 6. Mapping Design Principles to Design Requirements and Features

The second design principle (use imported and retrieved knowledge) requires implementing two further design features. First, a functionality to manually upload knowledge provides the content to initially execute automatic mining (DF4). Thereafter, the knowledge base's initial content can be iteratively extended by means of automatic knowledge creation using knowledge retrieval mechanisms (DF5).

In summary, overview the design features along a typical RE process (Figure 7). In practice, variations of this process are possible; for example, providing imported knowledge (step 1) could be a one-time activity before processing the very first NLRR.

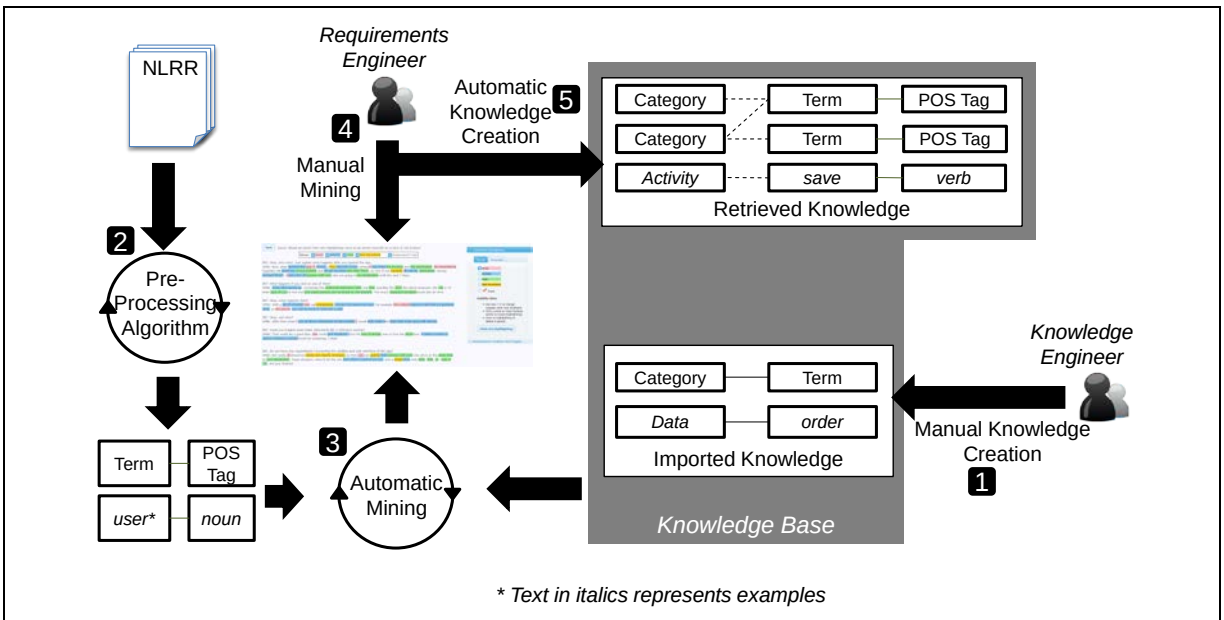


Figure 7. Requirement Mining Process Supported by our Proposed Design

Elaborating on this overview, a knowledge engineer can manually upload imported knowledge to the knowledge base as a first step of the requirement mining process (step 1 in Figure 7). Imported knowledge comprises terms associated with a specific requirements category (e.g., "credit card number" with the category "data requirement"). Then, during preprocessing (step 2), NLRs are transformed into single terms, which serve as an input for the automatic mining algorithm.

Similarly to former approaches (Casamayor et al., 2010; Cleland-Huang et al., 2007; Vlas & Robinson, 2012), our prototype uses NLP to implement automatic requirement mining (step 3). In it, techniques such as token detection, part of speech (POS) tagging, stop word elimination, and word lemmatizing are used. The result of this process is a set of tuples (term or POS tag; e.g., "supplier" and "noun"). Furthermore, automatic mining is enabled by an IR module comprising various algorithms based on the vector space model that Baeza-Yates and Ribeiro-Neto (1999) suggest.

The algorithms measure the similarity between the terms extracted from the documents and the terms from the knowledge base and assign the extracted terms to requirements categories. To do so, the categories are indexed like documents and the terms are the search queries. Consequently, a term's similarity to one of the categories is interpreted as the probability of the term belonging to this category.

Extending the above, we focus on automatic mining (step 3) in more detail. In it, there probabilities are calculated for all terms in the NLRR: based on retrieved knowledge, according to POS, and based on imported knowledge (steps "a" through "c" in Figure 8 respectively. On the basis of a term's POS, for example, a noun is more likely to be assigned to the category "data" than to "activity"[6].

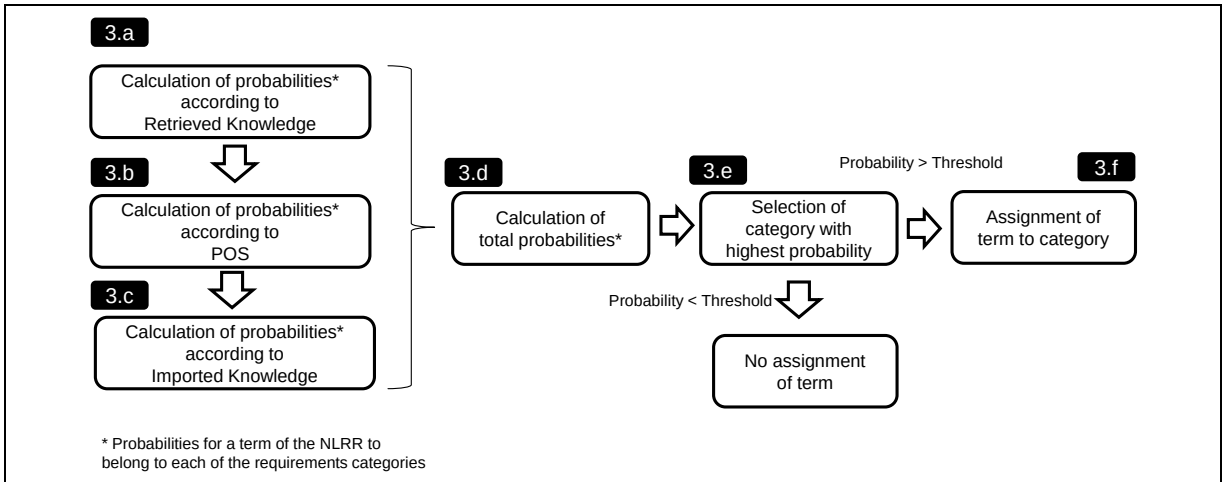


Figure 8. Individual Processing Steps During Automatic Mining

These three probability values (steps "a" through "c") are then integrated into a single, total probability value for each category (step "d"). The total value of the category with the highest probability is then chosen as a tentative classification of the term (step "e") and compared to a threshold (which can be customized to a value between 0 and 1). If the total probability were to exceed the threshold, the term should be assigned to the corresponding category (step "f").

Additionally, our prototype contains a functionality to enable one to manually identify and classify requirements (step 4 in Figure 7). During manual mining, the requirements engineer approves the results of the automated pre-processing. In that, the engineer can change or even delete requirements that the algorithm has suggested. Figure 9 shows a screenshot of the user interface for manual mining.

[6] The POS probability values have to be defined before running the algorithm. For example, the POS probabilities can be defined by using the percentage of assignments in the knowledge base: If any given term is assigned to the category "verb" 60 percent of the time, the weighting factor for this category is 0.6

Figure 9. User Interface for Manual Mining

The requirements are highlighted in the NLRR and different highlighter colors represent different requirements categories, which incorporates the metaphor of using text markers in physical documents. The initial list of requirements categories and the associated highlighting colors are based on the main requirements types that Robertson and Robertson (2006) describe in their Volere requirements process. Accordingly, functional requirements and non-functional requirements are distinguished. The former is further split into the categories “data” (for text passages describing data fields or objects) and “activity” (for text passages describing either the user or the system’s behavior). One can also use the category “actor” to indicate if a requirement is associated more with a user activity or a system activity. The text in the figure (in this case an interview transcript) contains highlights marking single words or entire text passages with a specific category’s highlighter color. Users can choose a highlighter color and highlight words with a single click. Another click deletes the highlighting to correct a false classification.

In addition, one can add further requirements. The finally approved requirements are then used for the automatic knowledge creation of retrieved knowledge (step 5 in Figure 7). Retrieved knowledge comprises terms, their associated requirements categories, and POS tags. To allow the RMS to learn from previous assignments, we combined imported knowledge (Ambriola & Gervasi, 2006; Kiyavitskaya & Zannone, 2008) with dynamically built-up knowledge gained from processing NLRRs (Gacitua et al., 2011; Goldin & Berry, 1997; Kof, 2004; Rayson, Garside, & Sawyer, 2000) to reduce the knowledge-creation efforts.

We designed our prototype (REMINER) embodying these features as a Web-based client-server system based on a three-tier architecture comprising a data tier, an application tier, and a presentation tier. For REMINER, we use components that are publicly available as open source alongside components we implemented ourselves as part of our research project (more details on the technical implementation can be found in Meth, 2013). Figure 10 overviews the system architecture.

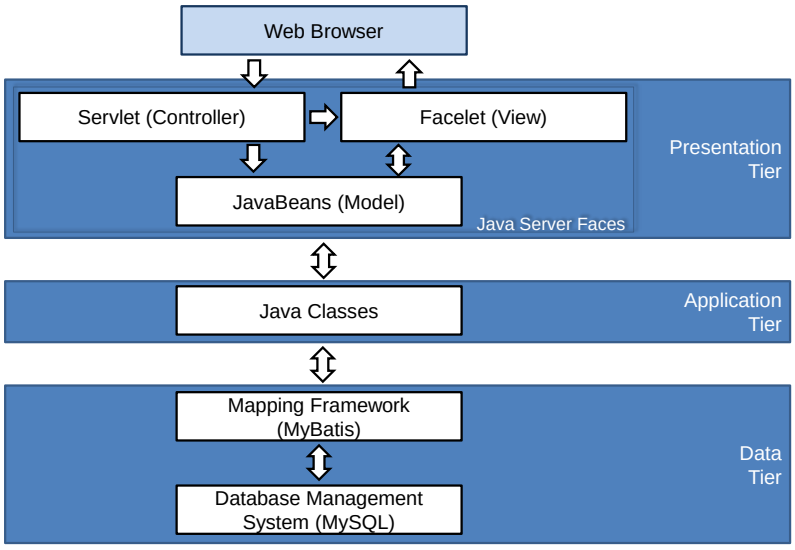


Figure 10. Prototype System Architecture

5. Ex-Post Evaluation

5.1. Evaluation Design

Our ex post evaluation comprised two parts. First, we conducted an artificial ex post evaluation of REMINER based on a laboratory experiment with a student sample. We used student participants because, with them, one can obtain a relatively large sample size with reasonable effort and achieve an adequate statistical power (Gallupe & McKeen, 1990). In this experiment, we compared the results achieved through system-supported requirement mining to manual discovery to address the research gap we describe above. Second, we added a naturalistic ex post evaluation by drawing on requirement mining experts in the field (Pries-Heje et al., 2008; Venable, 2006). While such a field setting cannot be as controlled as our lab setting (i.e., focused on internal validity), we suggest that naturalistic settings (real contexts) are more likely to actually reflect the proposed design's interplay with practitioners' praxis (real people and real problems; i.e., external validity) (Pries-Heje et al., 2008).

To evaluate our design principles, one can switch the REMINER's DFs associated with both design principles on and off, which results in different RMS configurations that one can evaluate separately. For example, semi-automatic requirement mining (DP1) could be switched on, while the usage of retrieved knowledge (DP2) could be switched off. While DP1 can be switched on independently from DP2, DP2 can only be activated when DP1 is, too. Through the separate activation of the design principles, the effects of each can be measured individually. Table 2 depicts the resulting three RMS configurations[7]. Based on these different configurations, our ex post evaluation used a single-factor, within-subject design to increase the statistical power and reduce the error variance that individual differences introduce (Hill & Lewicki, 2006). The within-subject factor is the RMS's configuration (manual requirement mining, semi-automatic requirement mining with imported knowledge, and semi-automatic requirement mining with imported and retrieved knowledge).

[7] See appendix B for exemplary proposed requirements after pre-processing.

Table 2. RMS Configurations

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

We conducted a pilot test to estimate the necessary sample size and appropriate length of the NLRRs used in the artificial experimental task. We applied the same single factor within-subject design that we applied in the main experiment in the pilot test. As a result, we recruited 40 participants for the main experiment. The participants were graduate students enrolled in a master-level IS course at a public university in Germany. Table 3 overviews our sample's characteristics.

Table 3. Participants' Descriptive Data (Average Values)

Age	Gender	Majors	Computer experience	Requirement mining experience
25.4 years (SD = 2.07)	Male: 32 Female: 8	MSc business informatics: 36 MSc of management: 4	4.75 out of 5	1.79 out of 5

We randomly assigned these participants to six time slots on three experimental days, with 6 or 7 participants per time slot. We performed the experiment in a multimedia classroom at the university. A lecturer of the IS course in which the students were enrolled introduced the experiment as an exercise for a course-related assignment with the objectives of understanding the different requirements categories relevant for business intelligence and learning how to use a Web application to perform requirement mining from text documents. None of the participants had access to the RMS before the experiment, nor were they aware of the experiment's purpose. Figure 11 depicts the experiment's procedure.

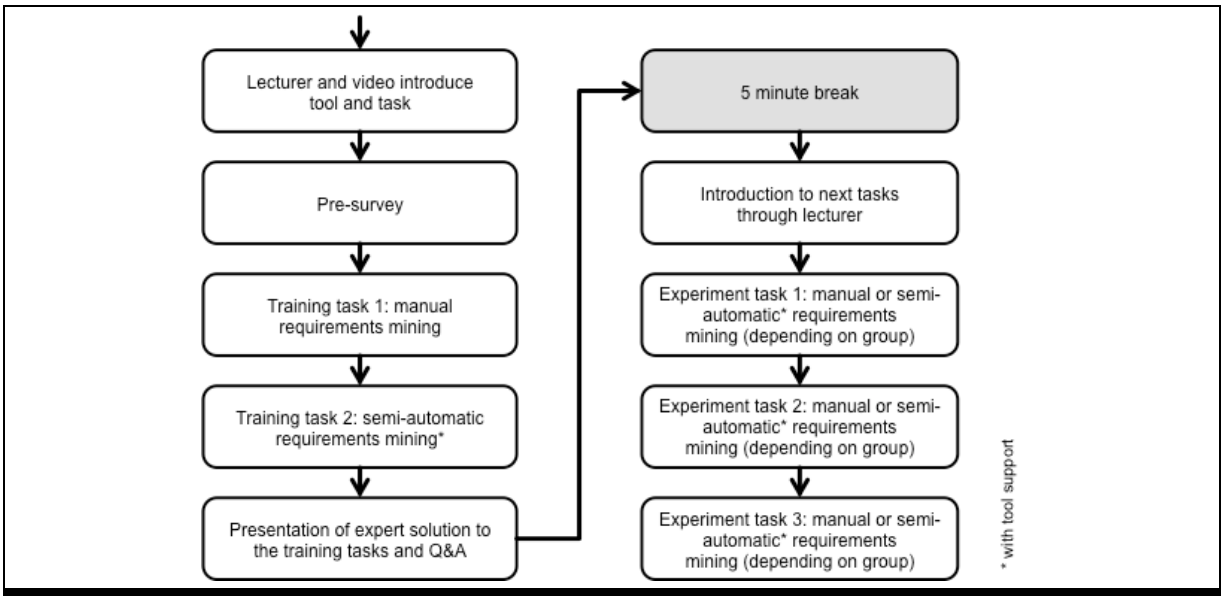


Figure 11. Experimental Procedure

The lecturer presented a tutorial video to teach the participants how to perform requirement mining and how to use REMINER. Thereafter, the lecturer asked the participants to fill in a brief

questionnaire about their demographic information, computer experience, and requirement mining experience. Next, the lecturer guided the participants through a training session to familiarize them with requirement mining. The lecturer asked the participants to perform requirement mining using the transcript of an interview on requirements for a train-reservation application for smartphones.

We chose the transcript from a series of previously conducted and transcribed interviews. In the first five minutes, the participants undertook the requirement mining manually. In the next five minutes, they performed requirement mining from the same transcript but with a few automatically mined requirements present at the start. Subsequently, the lecturer presented an expert's sample solution and answered the participants' questions if any. This was followed by a five-minute break.

After the break, the lecturer asked the participants to practice their requirement mining skills with a different set of interview transcripts comprising three transcripts of interviews on requirements for a car-sharing application for smartphones. The requirement mining within the three interview transcripts was supported by three different RMS configurations. To compensate for learning and fatigue effects in the within-subject design, we fully counterbalanced the three RMS configurations' presentation order across the participants, which yielded a total of six orders. The lecturer randomly assigned the participants to one of the six RMS configuration orders. The lecturer gave the participants five minutes to perform the requirement mining from each interview transcript. The lecturer then instructed them to switch to the next interview transcript and start requirement mining from it.

The naturalistic ex post evaluation principally followed the same design. In the field experiment, the participants were five requirements engineers (targeted users of the RMS) recruited from a large high-tech company. Table 4 overviews the sample's characteristics.

Table 4. Experts' Descriptive Data (Average Results)

Age	Gender	Job areas	Requirements engineering experience	Requirement mining experience
34.8 years (SD = 3.56)	Male: 3 Female: 2	• Product owners / managers • New product development managers	5.0 years (SD = 5.83)	3.6 years (SD = 1.14)

In the field experiment, we had to allow for a few modifications that were not present in the lab setting. First, we randomly assigned the participants to one of the RMS configuration orders. Since only five participants were involved in the field experiment and each participant randomly received a different RMS configuration order, we covered only five of the six possible RMS configuration orders in the field experiment. Secondly, we introduced the participants to the study's purpose as "obtaining experts' opinions on future RMS designs". None of the participant had access to REMINER before the experimental tasks, nor were they aware of its real purpose. We told the participants to work on the different tasks at their normal working pace. All the other procedures in the field experiment were the same as in the laboratory experiment.

Beyond our research's general limitations, which we discuss in Section 7, we reflect on the naturalistic ex post evaluation to correctly interpret its results and to later draw conclusions. While both the artificial and naturalistic ex post evaluations are similar in their design and procedure, they have two important differences. First, in contrast to the controlled lab setting, more aspects of the field setting were beyond our control, which reduced the naturalistic evaluation to a quasi-experimental one that could not easily capture and explore all the potentially relevant controls. Second, when drawing on real experts, this resource was not as readily available as the students. As a result, we relied on experts' voluntary participation and cannot present a sample that is sufficiently powerful to allow for the same kind of statistical analyses as in the artificial sample.

For both of these reasons, we caution against simply comparing the results let alone integrating their measurements into a single sample. We do, however, believe that we can compare our experts' recognized behavioral patterns with the artificial setting's results. Prior studies encourage us in this belief because they suggest that causal relationships are more generalizable across populations than specific characteristics are (Pedhazur & Schmelkin, 1991). In our case, this may indicate that the causal relationships between the RMS's design principles and the improved requirement mining productivity may remain true across different samples. To support this, we searched for systematic differences between the two experiments' results. We also checked whether experts had any material concerns either about the tool itself or its impact on their work when using our prototype.

5.2. Expected Productivity Effects of Design Principles

Altogether, the ex post evaluation focused on testing the two design principles' (DP1, DP2) effect on requirement mining productivity. Thus, our evaluation measures the effects that alternative combinations of the design principles have on multiple dependent variables. We conceptualize requirement mining productivity as an input-output ratio wherein the identified and classified requirements' quality serves as the output part (numerator of the ratio) and the invested mining effort as the input part (denominator). We used this ratio as this study's dependent variable.

Precision and recall are common measures to evaluate automatically identified requirements' quality and we similarly employ them here (Casamayor et al., 2010; Cleland-Huang et al., 2007; Gacitua et al., 2011). They are calculated by comparing participants' requirement mining outputs with expert solutions (Kiyavitskaya & Zannone, 2008; Vlas & Robinson, 2012). We can see recall as measuring completeness: it compares the number of correctly identified requirements with the total number of requirements in an NLRR. Precision measures correctness and is calculated as the proportion of correctly identified requirements in comparison to the number of identified requirements in an NLRR (see Table 5).

Table 5. Measurement of Recall and Precision in the RMS Context

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

One can measure requirement mining effort by the time a requirements engineer requires to conduct the mining task; that is, transforming an unstructured input document into a set of classified requirements. Because our experiment was based on a fixed time schedule, we also fixed the requirement mining effort and measured only the differences in the recall and precision (i.e., the quality of the identified requirements). Consequently, as in Diehl and Stroebe (1991) and Gallupe and McKeen (1990), the evaluation measured productivity in a fixed time period.

In terms of productivity effects related to recall, process automation (covered in our DP1) is a well-known mechanism to improve productivity for IT supported and non-IT supported processes (Atkinson & Kuhne, 2003; Jämsä-Jounela, 2007). In the case of automated requirement mining, we expect that productivity (measured by recall in a fixed time period) will similarly improve because an algorithm can automatically identify a large percentage of requirements without wasting the requirements engineer's time during the analysis (Cleland-Huang et al., 2007; Kiyavitskaya & Zannone, 2008; Vlas & Robinson, 2012).

Investigating this assumption from a theoretical point of view, we revisit the analogy to decision making we develop in Section 2.2. Automatically proposed requirements are a suggestive form of DG. In their experimental study, Parikh et al. (2001) showed that participants with DG outperformed those

without DG concerning the achieved decision quality and efficiency. Furthermore, suggestive guidance resulted in an increase in decision quality and efficiency in comparison to informative guidance. Since requirement mining is a specific instance of a decision making process, the application of automation mechanisms should similarly result in improvements in the requirements quality and efficiency in comparison to manual requirement mining.

Investigating this assumption from a process point of view, we can see the recall using a semi-automatic RMS as a sum of the automatism's initial recall and the recall resulting from subsequent manual adaptations 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 them to requirements categories. Therefore, we expect no significant recall difference between semi-automatic and manual RMS regarding these manual activities. In contrast, the initial recall that the automatism provided will remain and, thus, we expect it to have a significant effect. Consequently, we hypothesize:

> **H1**: *In a fixed time period, using a RMS that supports semi-automatic requirement mining with imported knowledge will result in higher recall than using a RMS that supports only manual requirement mining.*

Along similar lines, automated requirement mining requires a knowledge base containing requirements and a categorization of these elements (addressed by DP2). One can trace each automatically identified requirement back to a specific entry in this knowledge base. Accordingly, we can expect a more elaborate and extensive knowledge base to result in a higher percentage of identified requirements and, therefore, in reduced manual efforts (Cleland-Huang et al., 2007). In their empirical study on DG's effects, Parikh et al. (2001) observed similar effects concerning dynamic guidance. Based on knowledge that is dynamically created throughout the usage process, dynamic guidance outperformed predefined guidance concerning decision quality and efficiency. This finding is also in line with observations made during the interim evaluation.

Beside the size of the knowledge base, the domain specificity of the knowledge plays an important role in the requirement mining process (Casamayor et al., 2010). Generally, we can expect a higher degree of domain specificity to deliver better mining results (Lemaigre et al., 2008) by, for example, including domain-specific requirements (such as "conductor" or "attendant") beside domain-independent ones (like "manager" or "worker"). As we depict in Section 4.2, we propose two sources of knowledge to fill the knowledge base: in addition to manually imported knowledge, which is commonly used in existing RMSs (Kiyavitskaya & Zannone, 2008; Vlas & Robinson, 2012), the content of the knowledge base can be extended with automatically retrieved knowledge originating from previously processed NLRRs. As we describe when conceptualizing DP2, this should increase the size and domain specificity of the knowledge base. Further, encouraged by the interim evaluation's findings, we hypothesize that:

> **H2**: *In a fixed time period, using a RMS that supports semi-automatic requirement mining with imported and retrieved knowledge will result in higher recall than using a RMS that only supports semi-automatic requirement mining with imported knowledge.*

Because both recall and precision determine requirements quality, any impact on precision also has to be evaluated. However, in automated requirement mining from NLRR, recall is significantly more important than precision because 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 overlooked ones (Cleland-Huang et al., 2007). Berry et al. (2012) make the same argument by stating that requirement 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). Consequently, the artifact's design principles primarily address an improvement in the recall rate and do not target precision improvements.

Moreover, while the automatism's ability to find as many relevant words and text passages as possible predominantly determines the recall rate, the precision rate is strongly linked to the quality of the judgments following the identification of a word/text passage. A significant precision improvement can, therefore, only be realized if the algorithm provides better judgments than a human. However, since the RMS's proposed requirements are based on knowledge that humans created, this cannot be expected. In turn, any automatism used should also not negatively impact requirements determination productivity. Consequently, we hypothesize that:

> **H3**: *In a fixed time period, using a manual RMS, a RMS that supports semi-automatic requirement mining with imported knowledge, or a RMS that supports semi-automatic requirement mining with imported and retrieved knowledge does not result in significant differences in precision.*

Summarizing this section, Figure 12 overviews the hypotheses in a comprehensive evaluation model. Relating these considerations back to Kuechler and Vaischnavi (2012), the DPs as instantiated by the respective DFs function as prescribed actions that are intended to lead to the hypothesized improvements in recall (while keeping precision stable). Measuring these hypothesized effects will, in turn, allow us to conclude that the conceptual cause and effect links between the DPs and the design requirements we developed in Section 4 actually help understand how and why an RMS following our DPs improves the productivity of individual requirement engineers.

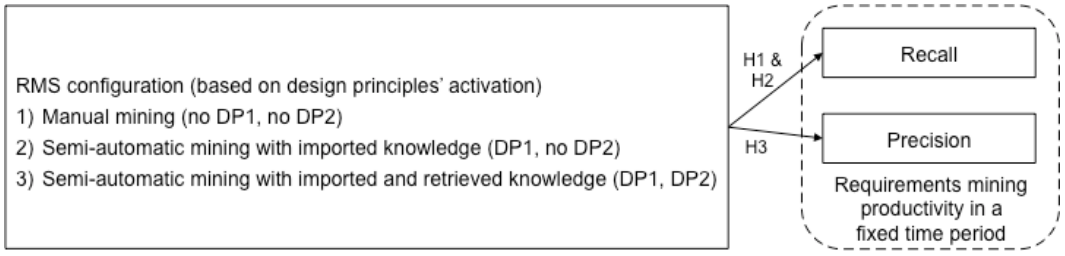


Figure 12. Evaluation Model

5.3. Evaluation Results

To test our hypotheses, we used a RMANOVA to examine the design principles' impacts on requirement mining recall and on precision during the lab experiment[8]. We performed pair-wise comparisons on RMS configurations' main effects to test Hypotheses 1 and 2. We applied a Bonferroni correction to control for the family-wise error rate (Vasey & Thayer, 1987). All pair-wise comparisons were significant at the 0.05 level, which means that the student participants using semi-automatic requirement mining with imported knowledge achieved significantly higher recall than those using manual requirement mining. Moreover, those using semi-automatic requirement mining with imported and retrieved knowledge achieved significantly higher recall than those using only semi-automatic requirement mining with imported knowledge. Thus, Hypotheses 1 and 2 were supported.

Hypothesis 3 regarding precision was also supported by the RMANOVA: we detected no significant difference in precision between the three RMS configurations ($p = .263$).

In the evaluation, we measured the effects of the two design principles on requirement mining productivity compared to the productivity of manual requirement mining. More specifically, we analyzed how semi-automatic requirement mining (DP1) and the combined use of imported and retrieved knowledge (DP2) affect requirement mining recall and precision in a fixed time period.

Concerning DP1, we found that the use of semi-automatic requirement mining significantly improved requirement mining recall. Starting with an initial recall of 54.0 percent (which the automatism

[8] For details, please see Appendix A and Meth et al. (2012b).

provided), the participants using semi-automatic requirement mining (DP1) achieved an average recall of 69.8 percent. This result is significantly better than that from manual requirement mining (50.7%). Analyzing the results of DP2, we found that the additional use of extracted knowledge caused further improved requirement mining recall. By applying DP2, the automatism provided an initial recall of 75.0 percent. Starting from this, the participants improved the recall to an average of 79.5 percent, which is significantly higher than the recall achieved through just DP1's activation (69.8%). These results (see Table 6) clarify that the automatism provided an initial recall that was already higher than manual mining's final recall. To correctly interpret the relative differences, though, keep in mind that the subjects had to manually check and approve the identifications and classifications suggested by DP1. Only after this check was done (i.e., the advice that the system gives is actually taken) did these initial suggestions translate into approved requirements, which were then counted for the final precision and recall values.

Table 6. Recall Values

Artifact configuration	Initial recall (%)	Final recall (%)
Manual only	**0**[9]	**50.7**
DP1	**54.0**	**69.8**
DP1 + DP2	**75.0**	**79.5**

Examining the evaluation results, we believe that our subjects relied on the RMS's initial identification and classification to allow them to use at least a part of their time to increase the recall through manually mining additional requirements (rather than using the entire time to correct potential automatism mistakes). Nonetheless, the approval process generally requires time and cognitive effort, which helps explain why the relative increase between the initial and the final recall in our lab experiment was lower when using the RMS than when doing manual mining. While this analysis' effort would initially be large, we believe that the assessment time per requirement decreases with an increase in trust in the proposed identifications and classifications. Consequently, the cognitive effort required to approve a requirement decreases over time, which explains why the participants could process more proposed, or even totally new, requirements, resulting in an increase in the recall.

As expected, DP1 did not significantly affect the precision. In comparison to the precision achieved during manual requirement mining (71.0%), DP1 resulted in a comparable precision value of 72.0 percent (non-significant improvement). Similarly, adding DP2 also did not significantly affect the precision. DP2 resulted in a precision value of 73.2 percent, which is comparable to that of manual requirement mining (71.0%) and is also a non-significant improvement. Reflecting on the evaluation up to this point, we observe that both design principles we propose had positive effects on requirement mining productivity. When comparing a configuration incorporating DP1 with manual mining as a baseline, there was a significant increase in recall. When switching on DP2 in addition to DP1, this initial effect was surpassed. This further increases the recall and, ultimately, the productivity, while the precision levels remain stable across all configurations[10].

While these results allowed us to be optimistic about our design's potential to improve RMSs, we extended our initial evaluation. Given that we conducted the above experiment with students only, we also reconnected with the experts whose knowledge and experience we had leveraged in the design project's earlier phases to involve them in a naturalistic evaluation. Table 7 overviews and compares the results.

[9] We did not measure the initial recall for manual mining. However, because subjects started with an NLRR that was not pre-processes, the initial recall logically was 0.

[10] This remains true if effort (the input factor / our dependent variable) is measured by the frequency of keystrokes and mouse clicks—often termed physical effort (Tamir et al., 2008)—and not kept constant. Using a screen capture tool, we found that the DPs had similarly significant and strong effects on productivity.

Table 7. Overview and Comparison of Ex-post Evaluations

	1) Manual		2) Semi-automatic with imported knowledge		3) 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%

We performed a RMANOVA on recall to compare the effects of different RMS configurations on the naturalistic ex post evaluation. The results show a significant difference in the participants' recall as the RMS configuration varied ($F(2, 8) = 31.74$, $p < .001$, $\eta 2 = .89$, $f = 2.82$). The pair-wise comparisons with Bonferroni corrections indicate that semi-automatic requirement mining with imported knowledge outperformed manual requirement mining regarding recall (mean difference = 31.0, $p = .007$, 95% CI [13.4%, 48.7%]), but no significant difference existed between semi-automatic requirement mining with imported knowledge and semi-automatic requirement 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%], $d = 0.95$). According to Cohen (1988), the observed effect size was a large effect. Thus, the insignificant result might stem from the very small sample size used in the field experiment. We conducted a post-hoc power analysis with G*Power 3 (Faul, Erdfelder, Lang, & Buchner, 2007). It showed that, to detect this effect size ($d = 0.95$) with a paired t-test, a sufficient power (e.g., 0.80) can only be achieved with samples larger than 12—a constraint we were unable to overcome.

As we expected, we detected no significant difference regarding precision in the practitioner sample analyzed by RMANOVA ($F(2, 8) = 0.34$, $p = .723$, $\eta 2 = .08$, $f = 0.29$).

The above analyses did not reveal evidence that the practitioner sample, compared with the student sample, demonstrates a different behavioral pattern regarding recall and precision when using the different RMS configurations. This finding is also in line with our observations during the naturalistic ex-post evaluation. We found no evidence 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 utmost caution.

6. Discussion

As the results of the experimental evaluations suggest that our proposed design principles actually increase requirement mining productivity, we integrate our findings to propose a basis for developing future RMSs. Building on the design developed in Section 4, we use the six key components of a design theory that Gregor and Jones (2007) introduce to guide this synthesis of our work. As we integrate our findings, we also take another look at the related literature and previous RMSs to illustrate our design's contributions to the requirement mining literature.

The first dimension that Gregor and Jones (2007) propose is purpose and scope; that is, identifying what the system is for. In response to the design requirements developed above, our design helps interested parties to develop RMSs that support requirements engineers in mining requirements from NLRRs. In doing so, systems built on our design improve a requirement engineer's overall

productivity as compared to a manual mining process. While our design is currently bound to the general software development domain, we suggest that the proposed class of systems could be applied to a wide range of NLRRs and to the context of various software and requirements engineering methodologies inside that scope. This highlights that our design deviates from previous literature on RMS design in that we do not bound our RMS to any specific type of requirement. Additionally, added evaluation cycles could see if the design's scope can be extended to incorporate general development processes for new products and services also outside the software domain.

Second, we mainly derived the justificatory knowledge we used for our design (see Sections 2 and 4) from literature on decisional guidance and on an abstract conceptualization of the interaction between a requirement engineer and an RMS as a sequence of advice seeking, giving, and taking. This, in close interplay with the grounding in practice, allowed us to identify design requirements and corresponding DPs, which we suggest are meaningful interventions to increase requirement mining productivity in practice. Compared to previous literature, this conceptualization of the problem is a novel approach to understanding tool support of requirement mining.

Third, we also identified relevant constructs that capture the artifact's impact. To this end, we suggest that the RMS be understood as an intervention into the process of manually mining requirements from any NLRR to improve its outcomes. Such improvement can be understood as an increase in requirements productivity, the key construct we propose. We conceptualized it with requirements quality and mining effort. We conceptualized the former by drawing on recall and precision as established output measures for requirement mining. Complementarily, we used time or the effort used to process a given NLRR as input measures conceptualizing the latter. Looking at our design requirements, the ability to maintain control over the selection of the decision strategy is an important side-condition that introduces perceived system restrictiveness as another important construct.

While these conceptualizations capture the dependent constructs (i.e., the effect of an RMS instantiating our design), the design itself incorporates several conceptual entities around which we developed it (see Section 4, especially Figure 4). For example, the RMS acts as an advice giver. Consequently, understanding the advice quality given by the system is an important independent construct directly linking to requirements quality. Following Silver (1991), we can describe the interaction between the requirement engineer and the RMS by the target, form, and mode of guidance as a final set of relevant constructs underpinning our proposed RMS design.

Logically, because these constructs are rooted in the conceptual interplay between design requirements and DPs, they link to the design's principles of form and function (fourth aspect of a design theory). These principles provide an abstract blueprint for constructing an RMS in the form of generic DPs and specific DFs (compare Figure 6). Following our goal to improve requirements engineers' productivity, these are means to support requirements mining by using an RMS.

Fifth, the artifact mutability is perhaps best reflected in the dynamic "evolution" of the knowledge base, which is linked to retrieved knowledge. Because an artifact based on the design we proposed is applied to an increasing number of NLRRs in a given context, its ability to identify and classify requirements correctly is likely to grow. As a consequence, the positive impact on productivity will increase accordingly. However, when changing context, this implies that any first analysis of NLRRs in a new domain critically relies on the quality and appropriateness of the important knowledge used. Additionally, we suggest that repeatedly using an RMS based on our design will increase a requirements engineer's trust in the system's initial recall, which will make the manual processing of this first advice less effortful. However, this positive effect might be counteracted by blind reliance on the system's recommendation and the corresponding effects on requirements quality. However, a more detailed exploration of these effects and their impact on artifact mutability has to be left to future studies at this point.

Finally, we also formulated a set of testable propositions regarding our design (H1 through 3) that we also used to evaluate the prototype we use as an expository instantiation. For future instantiations,

these might be helpful in developing test cases. The design, in turn, will also have to prove that such future instantiations also contribute to the productivity increases evident in our ex post evaluations.

In developing these hypotheses, and similarly to previous RMS proposed in the literature, our design integrates automatic mining and manual rework of the proposed requirements. Other than previous solutions, however, our design pays particular attention to the manual aspect in two regards. First, we explicitly use the quality of mining an NLRR manually as a baseline to compare the design to. This has shown that our design actually constitutes an improvement over processing NLRRs manually and is not just a solution superior to some predefined benchmark based on other RMS. Second, the design proposed here also considers the manual rework often required to transform proposed into approved requirements. Beyond the design, we also account for these considerations in the design of the evaluations we have carried out.

Summarizing the above discussion, Table 8 overviews our work's theoretical integration into a design theory for RMSs.

Table 8. RMS Design Theory

1	**Purpose and scope**	Explicit prescriptions for developing RMSs that support requirement mining from NLRRs to improve requirement mining productivity.
2	**Justificatory knowledge**	We derive design requirements and design principles from decision making theory and existing prescriptive knowledge from the requirements mining literature.

3	**Key constructs**	• Requirements productivity
		  o Requirements quality (precision, recall)
		  o Mining effort (time, effort)
		• Perceived system restrictiveness
		• Advice quality
		• Mode of guidance[11]

4	**Principles of form and function**	We derive design principles (DP 1 and 2) to support the requirement-mining process and knowledge-creation and maintenance processes and suggest corresponding design features (DF 1 through 5).
5	**Artifact mutability**	The contents of the knowledge base used for automatic mining and the underlying scheme for requirements categorization depend on the use context and will evolve as more and more NLRRs are processed in a given context. Also, a requirement engineer's experience with and trust in the system are likely to change as the system is used more, which will further reduce mining effort. In both cases, though, context (i.e., domain) changes will impact the artifact's design mutability.
6	**Testable propositions**	We formulate three hypotheses (H1-H3) to test the effects of different configurations of design principles on requirement quality as measured by recall and precision.

On top of this, and beyond earlier RMS designs, we synthesize our design into the design theory presented in Table 8 to explicitly present an abstract, conceptual solution rather than only a particular artifact. Owing to the design's abstraction and codification to generic design requirements and design principles, these are generalizable from the specific artifact to RMSs as a class of systems. In providing this conceptual solution, we hope to contribute to the requirements engineering literature in general and

[11] Because we kept both target and form of guidance are constant in our design (i.e., executional guidance and suggestive guidance respectively), we focus on the three modes of guidance (i.e., predefined, dynamic, and participative).

the RMS-related literature in particular by providing a general design for RMS—a common denominator for different systems instantiating this design. Such a specific differentiation of DPs and DFs is, to the best of our knowledge, a novel contribution to the respective literature. Future instantiations of our design can then serve as both tests to the underlying DPs and as creative variations and extensions of the prototype we suggest. Testing these competing designs will allow researchers to further develop the underlying conceptual design and, consequently, the conceptual understanding of tool-supported requirement mining we present here. Beyond such potential for future research, however, proposing a conceptual RMS design is the key theoretical contribution of our work.

Aside from this, RMSs should improve requirements engineers' productivity and, thus, provide added value, which manual requirement mining lacks. Our study complements existing RMS research by investigating if this expected productivity improvement can actually be observed.

Beyond the topical aspects of our research, we also contribute to the ongoing methodological discussion in the design science context. Reflecting on our research process, the effort we invested into maintaining a logical chain of evidence from design requirements to design principles to design features has greatly helped us in our design research project in three respects. First, our detailed analysis of the requirements we identified through our experts, own experiences, and extant literature helped us to identify a sound theoretical underpinning. Second, the traceability of design features all the way back to design requirements via design features strongly guided our implementing our prototype and, in its own right, improved our conceptual understanding and the relevance of the solution we propose. Third, the design principles' conceptualization allowed us to develop an experimental evaluation that is capable of quantifying each principle's effect on a dependent variable. This way, we can advance beyond "black box" testing in that our evaluation does not only show that there is an effect but also allows conclusions about why the effect changes occurred once we provided our prototype to the experiments' participants. Going beyond assessing the artifact's overall effect alone, this procedure allows precise inference from the evaluation back to the design process and the conceptual understanding of the problem. This approach could inform other design researchers' evaluation of their artifacts and the underlying design principles.

With regard to practical contributions, and beyond the market-driven forces we discuss in Section 1, we think that current software industry trends are likely to intensify the need for automated support of the requirement mining process. Among these, agile development and its repeated interaction with and involvement of customers (or users) are likely to make requirement mining a constant task throughout the development process because requirements are determined iteratively (Ramesh, Cao, & Baskerville, 2007). Building on imported and dynamically retrieved knowledge in requirement mining might reduce the added burden for requirements engineers. This goes hand in hand with the further emphasis placed on user-centered design (compare, e.g., ISO 9241-210). In this increased interaction with the user, one compiles a plethora of unstructured and semi-structured documents and materials that needs to be analyzed during requirement mining. A tool based on the design principles we propose is likely to support the requirements engineer through automation (DP1) and additional knowledge resources (DP2), which help improve productivity. The RMS's ability to support is especially true when accounting for the increased relevance of distributed requirements discovery (Agerfalk, Fitzgerald, & Slaughter, 2009; Pries-Heje & Pries-Heje, 2011). In this process, multiple types of information and communication support have been established to support distributed requirement mining. Using, for example, wikis (Geisser, Heinzl, Hildenbrand, & Rothlauf, 2007) or issue tracking systems (Scacchi, 2002), an early documentation of requirements—often in natural language—can also be achieved in distributed settings. In turn, requirements engineers need to process additional sources of NLRRs to ensure a development project's success.

In addition, the experiment's results show the potential benefits of integrating requirements and knowledge engineering activities. The evaluations provide evidence that reusing knowledge across different software development projects in the same or similar domains can result in better requirements specifications. Accordingly, software vendors could benefit from reusing knowledge across different products of the same product group. Similarly, customer companies could share knowledge across different applications in the same line of business. Apart from using an RMS, other

technologies (e.g., domain-specific wikis), directories (e.g., glossaries), and organizational means (e.g., lessons learned sessions or specific roles in the development team) can also foster knowledge reuse in requirements engineering. Finally, our study helps software vendors of requirements engineering tools to improve their software packages with regard to automated requirement mining capabilities. While support for manual requirement mining has been incorporated into selected commercial software packages (e.g., IBM Rational Doors), automated mining support is still scarce. The depicted conceptual design can inspire and guide future commercial RMSs by constraining their solution space and, thereby, improve design outcomes. When implemented in commercially available software and applied in a requirement mining process, the presented design principles can help increase requirements engineers' productivity and, thus, address a considerable requirements engineering practice problem.

7. Conclusion

As we discuss in Section 1, to answer our research question, we focused on 1) deriving a theoretically grounded RMS design, 2) developing an artifact based on this design, and 3) evaluating whether the requirement mining that this artifact supports actually results in an increased productivity (when compared to manual discovery). Addressing these three items, we introduce a set of theoretically founded design principles that can support future RMSs' development and instantiation. Our evaluation of a corresponding prototype (REMINER) suggests that an RMS instantiated based on these DPs actually helps increase individual requirement engineers' productivity because it positively impacts recall (all other things being equal) through high-quality advice and a mixture of predefined, dynamic, and participative guidance.

To adequately interpret the implications of these findings, readers should consider the following limitations of our research. In the RMS design's conceptualization, we applied decision making theory—a specific theoretical viewpoint—to underpin the design requirements and principles. Choosing alternative theoretical viewpoints could result in additional or different design requirements and principles. However, the results of the various steps we take to evaluate our design confirm that 1) both DPs positively affect the approved requirements' quality and 2) the quality of proposed requirements (the given advice) strongly determines the approved requirements' quality (the decision taken). These results are in accordance with decision making theory. Therefore, evidence exists that the conceptualization provides an appropriate and relevant basis for RMS design and the derivation of meaningful design requirements and principles. In respect to the experimental evaluation's conduct, a further limitation is that we analyzed the experiment text data based on manual document analysis. Although we conducted this analysis thoroughly to provide our sample solution, manual analysis is error-prone and can reduce reliability. However, two researchers independently analyzed the results and achieved a high inter-rater reliability (98.97% in the documents coded twice), which provide evidence that this limitation did not have a major impact. Another limitation is that our ex post analysis relied primarily on a student sample. However, Pedhazur and Schmelkin (1991) point out, causal relationships tend to be generalizable across populations. Although the experts' naturalistic evaluation was not as powerful as the artificial evaluation and needs to be interpreted with the respective caution, it did not reveal any behavioral patterns that contradict the results based on our student sample, which further increases our confidence in our evaluation's results. Nonetheless, the student sample may exhibit a high degree of learning. While we fully counterbalanced the various manipulations, this learning could pertain to the task of mining requirements as a whole. By studying this effect, future studies could include more experimental groups to better control for potentially confounding factors.

Beyond such improvements to the current design, there are many possible extensions to our work. In keeping with Hevner et al.'s (2004) statement that DSR is inherently iterative, future research could extend the presented conceptual design with additional design cycles. During these cycles, alternative theoretical lenses could be applied or a more intensive observation made of the artifact's usage in an actual software development project (e.g., in the form of an action design research study). Both extensions promise interesting adaptations and enrichments of the identified design components. From an evaluation point of view, a replication of the experiment study in a different domain could also add interesting insights. If 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 then need to be manually established and might not be reused in further requirement mining. Future research could use a more sophisticated domain and differentiate the participants according to their domain knowledge and specifically examine the moderating effects of the participants' domain knowledge on the relationships between design principles and productivity.

Adopting a perspective that goes beyond the support of the requirements engineers when faced with NLRRs is another opportunity for future studies. Complementary to our mainly cognitive focus, added support for the interaction between requirements engineers and future users could also help increase productivity. While beyond the scope of our research, such research might also more specifically account for different software development approaches and processes (e.g., agile software development). Another idea is to go beyond tool support altogether. In this vein, software developers' and business domain experts' joint custody of the knowledge base (related to artifact mutability above) is likely to facilitate the identification of a comprehensive set of requirements and its incorporation into new software.

Finally, the design we propose specifically builds on the idea to automation through automated pre-processing of an NLRR with human intelligence and cognition. Beyond responding to an immediate need in the software development domain (Mich et al., 2004), the design's underlying metaphor of advice giving and advise taking will likely hold for other decision support systems outside this domain, too. Further exploring this based on the design suggested might open up additional avenues for contributions to both practice and science.

Acknowledgements

We thank the editor and the anonymous reviewers of the paper for their valuable guidance and developmental comments in revising this manuscript. We also express our gratitude towards the participants at conferences and workshops where earlier versions of this paper were discussed for their helpful thoughts and suggestions—especially those of the *JAIS* Theory Development Workshop 2012. Our special thanks go to Ye Li for her support in designing and conducting the artifact's evaluations and our industry partners for their participation in and invaluable support of this research.

References

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

Ambriola, V., & Gervasi, V. (2006). On the systematic analysis of natural language requirements with CIRCE. *Automated Software Engineering*, *13*(1), 107-167.

Appan, R., & Browne, G. J. (2012). The impact of analyst-induced misinformation on the requirements elicitation process. *MIS Quarterly*, *36*(1), 85-106.

Agerfalk, P. J., Fitzgerald, B., & 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), 317-328.

Atkinson, C., & Kuhne, T. (2003). Model-driven development: A metamodeling foundation. *IEEE Software*, *20*(5), 36-41.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. Boston, MA: Addison Wesley.

Baskerville, R., & Pries-Heje, J. (2010). Explanatory design theory. Busine*ss & Information Systems Engineering*, *2*(5), 271-282.

Baskerville, R. L. (1999). Investigating Information Systems with Action Research. *Communications of the Association for Information Systems*, 2, 2-32.

Beach, L., & Mitchell, T. (1978). A contingency model for the selection of decision strategies. *The Academy of Management Rev*iew, *3*(3), 439-449.

Berry, D., Gacitua, R., Sawyer, P., & Tjong, S. F. (2012). The case for dumb requirements engineering tools. In B. Regnell & D. Damian (Eds.), *Requirements engineering: Foundation for software quality* (pp. 211-217). Berlin, Germany: Springer.

Boehm, B., & Basili, V. (2001). Software defect reduction top 10 list. *IEEE Computer*, *34*(1), 135-137.

Bonaccio, S., & 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), 127-151.

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

Casamayor, A., Godoy, D., & Campo, M. (2011). Mining textual requirements to assist architectural software design: A state of the art review. *Artificial Intelligence Review*, *38*(3), 173-191.

Castro-Herrera, C., Duan, C., Cleland-Huang, J., & Mobasher, B. (2009). A recommender system for requirements elicitation in large-scale software projects. In *Proceedings of the 2009 ACM symposium on Applied Computing* (pp. 1419-1426). New York, NY: ACM Press.

Cohen, J. (1988). *Statistical power for the behavioral sciences* (2nd ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. *Requirements Engineering*, *12*(2), 103-120.

Diehl, M., & Stroebe, W. (1991). Productivity loss in idea-generating groups: Tracking down the blocking effect. *Journal of Personality and Social Psychology*, *61*(3), 392-403.

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

Gacitua, R., Sawyer, P., & Gervasi, V. (2011). Relevance-based abstraction identification: Technique and evaluation. *Requirements Engineering*, *16(*3), 251-265.

Gallupe, R. B., & 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), 1-13.

Gardner, P. H., & Berry, D. C. (1995). The effect of different forms of advice on the control of a simulated complex system. *Applied Cognitive Psychology*, *9*(7), 55-79.

Geisser, M., Heinzl, A., Hildenbrand, T., & Rothlauf, F. (2007). Verteiltes, internetbasiertes requirements-engineering. *Wirtschaftsinformatik*, *49*(3), 199-207.

Goldin, L., & Berry, D. M. (1997). AbstFinder: A prototype natural language text abstraction finder for use in requirements elicitation. *Automated Software Engineering*, *4*(4), 375-412.
Gregor, S., & Hevner, A.R. (2013). Positioning and presenting design science research for maximum impact. *MIS Quartely,* 37(2), 337-A336.
Gregor, S., & Jones, D. (2007). The anatomy of a design theory. *Journal of the Association for Information Systems*, *8*(5), 312–335.
Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, *28*(1), 75-105.
Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian Journal of Information Systems*, *19*(2), 87-92.
Hevner, A., & Chatterjee, S. (2010). *Design research in information systems: Theory and practice*. Boston, MA: Springer.
Hickey, A. M., & Davis, A. M. (2004). A unified model of requirements elicitation. *Journal of Management Information Systems*, *20*(4), 65-84.
Hill, T., & Lewicki, P. (2006). Statistics: Methods and Applications, Tulsa, OK: StatSoft.
Huffman, H., J., Dekhtyar, A., & 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* (pp. 1-5).
Jämsä-Jounela, S.-L. (2007). Future trends in process automation. Annual Reviews in Control, 31(2), 211-220.
Kiyavitskaya, N., & Zannone, N. (2008). Requirements model generation to support requirements elicitation: the Secure Tropos experience. Automated Software Engineering, 15(2), 149-173.
Kof, L. (2004). An application of natural language processing to domain modeling—two case studies. *International Journal on Computer Systems Science Engineering*, *20*(1), 37-52.
Kuechler, W., & Vaishnavi, V. (2012). A framework for theory development in design science research: Multiple perspectives. *Journal of the Association for Information Systems*, *13*(6), 395-423.
Lemaigre, C., García, J. G., & Vanderdonckt, J. (2008). Interface model elicitation from textual scenarios. In *Proceedings of the Human-Computer Interaction Symposium* (Vol. 272, pp. 53-66).
March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision Support Systems*, *15*(4), 251-266.
Meth, H. (2013). *A design theory for requirements mining systems*. Mannheim: University of Mannheim.
Meth, H., Brhel, M., & Maedche, A. (2013a). The state of the art in automated requirements elicitation. *Information and Software Technology*, *55*(10), 1695-1709.
Meth, H., Maedche, A. & Einoeder, M. (2013b). Is knowledge power? The role of knowledge in automated requirements elicitation. In *Proceedings of the 25th International Conference on Advanced Information Systems Engineering* (pp. 578-593).
Meth, H., Maedche, A., & Einoeder, M. (2012a). 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).
Meth, H., Li, Y., Maedche, A., & Mueller, B. (2012b). Advancing task elicitation systems—an experimental evaluation of design principles. In *Proceedings of the 33rd International Conference on Information Systems*.
Mich, L., Franch, M., & Novi, I. P. (2004). Market research for requirements analysis using linguistic tools. *Requirements Engineering*, *9*(1), 40-56.
Neill, C. J., & Laplante, P. A. (2003). Requirements engineering: The state of the practice. *IEEE Software*, *20*(6), 40-45.
Nunamaker, J. F., Chen, M., & Purdin, T. (1990). Systems development in information systems research. In *Proceedings of the 23rd Annual Hawaii International Conference on System Sciences* (pp. 631-640).
Ossher, H., Amid, D., Anaby-Tavor, A., Bellamy, R., Callery, M., Desmond, M., De Vries, J., Fisher, A., Krasikov, S., Simmonds, I., & 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).
Parikh, M., Fazlollah, B., & Verma, S. (2001). The effectiveness of decisional guidance: An empirical evaluation. *Decision Sciences*, *32*(2), 303-331.
Payne, J. W. (1982). Contingent decision behavior. *Psychological Bulletin*, *92*(2), 382-402.

Pedhazur, E., & Schmelkin, L. (1991). *Measurement, design, and analysis: An integrated approach*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2008). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45-77.

Piirainen, K., & Briggs, R. (2011). Design theory in practice—making design science research more transparent. In H. Jain, A. Sinha, & P. Vitharana (Eds.), *Service-oriented perspectives in design science research* (pp. 47-61). Berlin, Germany: Springer

Pohl, K. (2010). *Requirements engineering: Fundamentals, principles, and techniques*. Berlin, Germany: Springer.

Pries-Heje, J., Baskerville, R., & Venable, J. R. (2008). Strategies for design science research evaluation. In *Proceedings of the European Conference on Information Systems*.

Pries-Heje, L., & Pries-Heje, J. (2011). Agile & distributed project management: A case study revealing why SCRUM is useful. In *Proceedings of the European Conference on Information System.*

Rago, A., Marcos, C., & Diaz-Pace, J. A. (2011). Uncovering quality-attribute concerns in use case specifications via early aspect mining. *Requirements Engineering*, *18*(1), 67-84.

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

Rayson, P., Garside, R., & Sawyer, P. (2000). Assisting requirements engineering with semantic document analysis. In *Proceedings of the RIAO* (pp. 1363-1371).

Robertson, S., & Robertson, J. (2006). *Mastering the requirements process*. Upper Saddle River, NJ: Addison-Wesley.

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

Scheiber, F., Wruk, D., Oberg, A., Britsch, J., Woywode, M., Maedche, A., Kahrau, F., Meth, H., Wallach, D., Plach, M. (2012). Software usability in small and mediuzm sized enterprises in Germany: An empirical study. In A. Maedche, A. Botzenhardt, & L. Neer (Eds.), *Software for people* (pp. 39-52). Berlin, Germany: Springer.

Silver, M. S. (1988). User perceptions of DSS restrictiveness: An experiment. In *Proceedings of the 21st Annual Hawaii International Conference on System Sciences* (pp. 116-124).

Silver, M. S. (1991). Decisional guidance for computer-based decision support. *MIS Quarterly*, *15*(1), 105-122.

Simon, H. A. (1957). *Models of man*. New York, NY: Wiley.

Simon, H. A. (1969). *The sciences of the artificial*. Cambridge, MA: MIT Press.

Takeda, H., & Veerkamp, P. (1990). Modeling design processes. *AI Magazine*, 11(4), 37-48.

Tamir, D., Marcos, S., & 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, W. F., & 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* (pp. 379-384).

Todd, P., & Benbasat, I. (1991). An experimental investigation of the impact of computer based decision aids on decision making strategies. *Information Systems Research*, *2*(2), 87-115.

Todd, P., & Benbasat, I. (1999). Evaluating the impact of DSS, cognitive effort, and incentives on strategy selection. *Information Systems Research*, *10*(4), 356-374.

Vaishnavi, V. K., & Kuechler, W. (2007). *Design science research methods and patterns: Innovating information and communication technology*. New York, NY: Auerbach.

Vasey, M. W., & Thayer, J. F. (1987). The continuing problem of false positives in repeated measures ANOVA in psychophysiology: A multivariate solution. *Psychophysiology*, *24*(4), 479-486.

Venable, J. (2006). A framework for design science research activities. In *Proceedings of the 2006 Information Resource Management Association Conference* (pp. 21-24).

Vlas, R. E., & 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), 11-38.

Walls, J. G., Widmeyer, G. R., & El Sawy, O. A. (1992). Building an information system design theory for vigilant EIS. *Information Systems Research*, *3*(1), 36–59.

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

Wilson, W. M., Rosenberg, L. H., & Hyatt, L. E. (1997). Automated analysis of requirement specifications. In *Proceedings of the 19th International Conference on Software Engineering* (pp. 161-171).

Winter, R. (2008). Design science research in Europe. *European Journal of Information Systems*, *17*(5), 470-475.

Xu, L., & Brinkkemper, S. (2007). Concepts of product software. *European Journal of Information Systems*, *16*(5), 531–541.

Yaniv, I. (2004). The benefit of additional opinions. *Current Directions in Psychological Science*, *13*(2), 75-78.

Appendices

Appendix A: Ex-Post Evaluation Results Overview[12]

Table A-1. Results of RMANOVA for Artificial Ex Post Evaluation

DV	Source	*DF*	*F*	*p*	η^2	Cohen's *f*
Recall	RMS config.	2	129.76	< .001	.77	1.82
	Error	78	-	-	-	-
Precision	RMS config.	2	1.36	.263	.03	0.19
	Error	78	-	-	-	-

Table A-2. Results* of Pairwise Comparison for Artificial Ex Post Evaluation

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%

*Recall only

Table A-3. Results of RMANOVA for Naturalistic Ex Post Evaluation

DV	Source	*DF*	*F*	*p*	η^2	Cohen's *f*
Recall	RMS config.	2	31.74	< .001	.89	2.82
	Error	8	-	-	-	-
Precision	RMS config.	2	.34	.723	.08	.29
	Error	8	-	-	-	-

Table A-4. Results* of Pairwise Comparison for Naturalistic Ex Post Evaluation

Pair comparison		Mean difference	*p**	95% confidence interval*	
				Lower	Upper
Semi-automatic with imported knowledge	Manual	31.0%	.007	13.4%	48.7%
Semi-automatic with imported and retrieved knowledge	Semi-automatic with imported knowledge	9.1%	.301	-7.8%	26.1%

*Recall only

[12] Further details on the results can be obtained from Meth et al. (2012b).

Appendix B: Screenshots of Different Pre-Processing Results

Configuration 2 (DP1 enabled, DP2 disabled):

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 a 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 account doesnt exist, I have the opportunity to create a new one. The app needs special information like first name and surname (only real names are accpeted!), nickname, age, my hometown (maybe with real adress-information to check if the person is real). Also car information (seats, size,.. maybe for the girls the colour). My email adress and very important, my cell phone number.

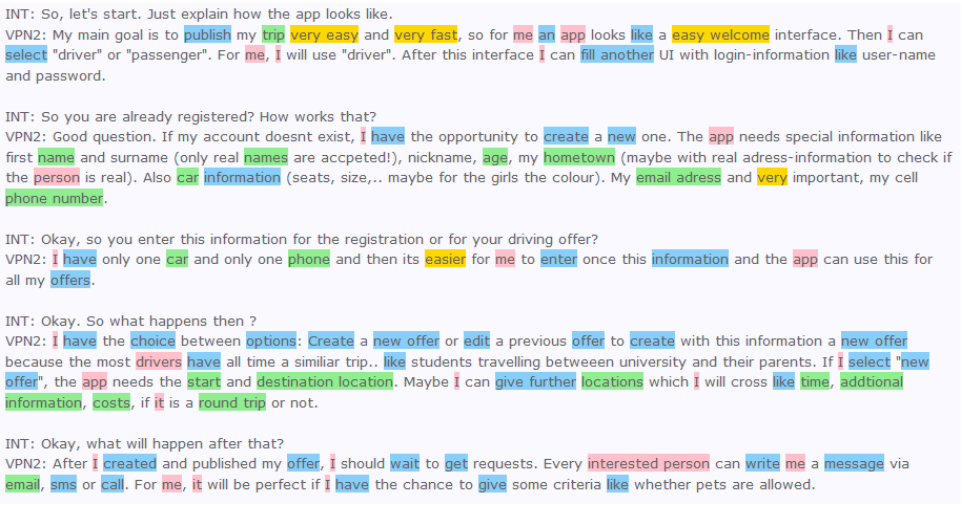
INT: Okay, so you enter this information for the registration or for your driving offer?
VPN2: I have only one car and only one phone and then its easier for me to enter once this information and the app can use this for all my offers.

INT: Okay. So what happens then ?
VPN2: I have the choice between options: Create a new offer or edit a previous offer to create with this information a new offer because the most drivers have all time a similiar trip.. like students travelling betweeen 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, addtional information, costs, if it is a round trip or not.

INT: Okay, what will happen after that?
VPN2: After I created and published my offer, I should wait to get requests. Every interested person can write me a message via email, sms or call. For me, it will be perfect if I have the chance to give some criteria like whether pets are allowed.

Figure B-1. Pre-processing Results with RMS Configuration 2

Configuration 3 (DP1 enabled, DP2 enabled):



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 a 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 account doesnt exist, I have the opportunity to create a new one. The app needs special information like first name and surname (only real names are accpeted!), nickname, age, my hometown (maybe with real adress-information to check if the person is real). Also car information (seats, size,.. maybe for the girls the colour). My email adress and very important, my cell phone number.

INT: Okay, so you enter this information for the registration or for your driving offer?
VPN2: I have only one car and only one phone and then its easier for me to enter once this information and the app can use this for all my offers.

INT: Okay. So what happens then ?
VPN2: I have the choice between options: Create a new offer or edit a previous offer to create with this information a new offer because the most drivers have all time a similiar trip.. like students travelling betweeen 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, addtional information, costs, if it is a round trip or not.

INT: Okay, what will happen after that?
VPN2: After I created and published my offer, I should wait to get requests. Every interested person can write me a message via email, sms or call. For me, it will be perfect if I have the chance to give some criteria like whether pets are allowed.

Figure B-2. Pre-processing Results with RMS Configuration 3

About the Authors

Hendrik METH works as a Manager for Business Intelligence at BorgWarner, Germany. He completed his diploma and PhD studies in Information Systems at the University of Mannheim, Germany. Before his current employment, he held various industry positions in the Business Intelligence area, as a project and product manager for Bosch and as a consultant for Icon SCM. His research focuses on Requirements Engineering, User-Centered Design and Business Intelligence/ Big Data topics. He published in several journals and conferences, such as *Information and Software Technology*, *IEEE Computing*, ICIS, and CAISE. He has served as a reviewer for conferences including ICIS and ECIS.

Benjamin MUELLER is an Assistant Professor at the University of Groningen in the Netherlands and an Associate Researcher at the Institute for Enterprise Systems at the University of Mannheim, Germany. He received his PhD from the EBS Business School in 2010 and holds graduate degrees in Management and Information Systems from EBS Business School and Georgia State University. Benjamin's research is focusing on the interplay of organizational and technological facets of Enterprise Systems. He is particularly interested in investigating how individual, organizational, and technological aspects of Enterprise Systems adapt to each other in times of organizational transformation. Complementary to that, he contributes the ongoing debate on theory and theorizing in the IS discipline. He has published over 40 peer-reviewed papers in various outlets such as, for example, the Journal of Management Information Systems or the International Conference on Information Systems. Benjamin serves the IS community in various editorial capacities.

Alexander MAEDCHE is Full Professor of Information Systems at the Business School and Managing Director of the Institute for Enterprise Systems (InES) at the University of Mannheim, Germany. He earned his doctorate from the Karlsruhe Institute of Technology (KIT), Germany. He has six years industry experience in information systems in positions as IT manager in the Bosch Group and vice president of product management at SAP SE. His research focuses on user-centered IS development and digital transformation as well as intelligent information systems. He has published more than 100 papers in journals and conferences, such as the *IEEE Intelligent Systems*, *AI Magazine*, *Information and Software Technology*, the International Conference on Information Systems (ICIS), and Conference on Advanced Information Systems Engineering (CAISE). He has served and serves as associated editor for ICIS and ECIS, is a PC member of CAISE, and *Business & Information Systems Engineering* (BISE) section editor.