Documenting Relations between Requirements and Design Decisions: A Case Study on Design Session Transcripts

Tom-Michael Hesse and Barbara Paech

Institute of Computer Science, Heidelberg University
Im Neuenheimer Feld 326, 69120 Heidelberg, Germany
{hesse,paech}@informatik.uni-heidelberg.de

Abstract. Context/Motivation: Developers make many important decisions as they address given requirements during system design. Each decision is explained and justified by decision-related knowledge. Typically, this knowledge is neither captured in a structured way, nor linked to the respective requirements in detail. Then, it is not obvious, how design decisions realize the given requirements and whether they further refine or shape them. Thus, the relations and alignment of requirements and design cannot be assessed properly. **Problem/Question:** While there are several studies on decision-making in general, there does not exist a study uncovering how decision-related knowledge emerges based on requirements. Such a study is important to understand the intertwined relations of requirements and design decisions as well as how requirement descriptions could be enhanced with feedback from design decision-making. **Principal Idea/Results:** We applied a flexible documentation approach for decision-related knowledge on discussion transcripts of two design sessions with professional designers. We analyzed the discussions for decision-related knowledge and documented it together with its relations to the given requirements. Several complex and incrementally growing knowledge structures for decisions were found to emerge in relation to the given requirements. Also, we uncovered that decision-related knowledge contained uncertainties about requirements and further refined them. **Contribution:** Our study uncovers detailed relations between requirements and design decisions and thereby improves the understanding of their mutual impact on each other. We also derive recommendations for the cooperation between requirements engineers and designers in practice. In addition, we demonstrate that our documentation approach for decision-related knowledge provides a comprehensive view on decisions and their relations to requirements.

Keywords: Decision documentation, decision-making, design decisions, requirements traceability, case study

1 Introduction

During software design many decisions are made. On the one hand, such design decisions significantly shape the structure of the developed system [11] with respect to the given requirements. On the other hand, design decisions also impact these requirements [16], as developers face uncertainties and need to clarify them. In addition, design decisions can potentially restrict or extend the given requirements for the system. Thus, knowledge about design decisions is crucial to assess the intertwined relations between requirements and design [8]. Typically, this decision-related knowledge is complex. For instance, it may consist of multiple issues and goals, alternatives for solving the decision problem, additional context information or rationales justifying the choice. We will refer to this knowledge as *decision knowledge* and call its belonging entities decision knowledge *elements*. All of these elements can be related to requirements and, therefore, can be important drivers of the system's design.

Whereas several studies exist on decision-making in design (cf. [5,22,25]), currently no study explicitly addresses relations between requirements and decision knowledge elements in detail. Such a study is hindered by the fact that comprehensive decision knowledge is often not accessible due to missing documentation. Even if decisions are documented, detailed knowledge structures are mostly not covered. Then, also relations to requirements are only captured coarse-grained for entire decisions. However, as requirements might be the origin or driver of particular decision knowledge elements [8], a fine-grained decision documentation of realistic design discussions is needed as a foundation for this study.

In this paper, we investigate the design discussion transcripts of professional software designers to identify any contained decision knowledge elements with their relations to requirements. Therefore, we have applied our incremental documentation approach for decision knowledge [8] on these transcripts. Then, the resulting knowledge structures and relations were analyzed. Our *overall goal* is *to better understand how given requirements are exploited by designers in their design decision knowledge*. The contribution of our study to this goal is to analyze the relations between requirements and decision knowledge elements stated by designers in their decisions. We identified and examined emerged structures of decision knowledge elements and their detailed interaction with requirements. This helps to understand how requirements and design decisions influence each other. Based on these findings, software designers can be supported in identifying and documenting the relevant decisions with respect to requirements. In addition, our findings provide insights on how requirements could be enhanced with feedback from the design process, for instance by clarifying potential uncertainties with the stakeholders. Moreover, we demonstrate the capability of our documentation approach to create a comprehensive view on decision knowledge and its relation to requirements for design decisions.

The remainder of this paper is structured as follows. In Section 2, we briefly describe our decision documentation approach and introduce the investigated design discussion transcripts with related studies. Section 3 presents our approach for coding and analyzing the transcripts with our research questions and the

resulting coding table. In Section 4, we describe our findings. Then, these results and the threats to validity are discussed in Section 5. Finally, we summarize our findings and present ideas for future work in Section 6.

2 Background and Related Work

In this section, we introduce our approach for *decision documentation*. Then, we briefly describe the *investigated design discussions* with the addressed requirements and present *related studies* for the transcripts and our research method.

Decision Documentation As already defined, *decision knowledge* is concerned with all information developers need to understand a given decision problem, its context and justifications for the decision. A decision problem at least comprises a set of alternatives, which can be compared by different criteria [15]. The context of decisions might consist of constraints brought up by requirements or assumptions on the environment of the developed system. So, the context might constitute or influence criteria within the decision problem. Justifications for the decision are typically given in form of arguments supporting or challenging alternatives. As we have pointed out in [17], different models address the documentation of decision knowledge during requirements engineering and software design. However, these models either represent the entire decision in a summarized way (e.g., in pre-defined textual templates) or they only focus on parts of decision knowledge. Then, not all structures and relations within decision knowledge can be captured. In addition, the existing approaches do not support an incremental documentation of decision knowledge.

Due to these shortcomings, we decided to apply our own decision documentation model as presented in [8]. Our model offers a variety of different knowledge elements and is depicted in Figure 1.

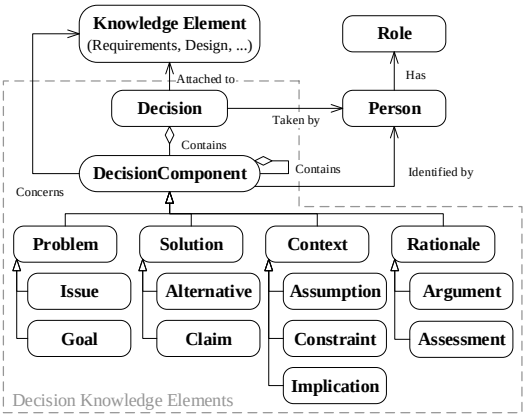

Fig. 1. Decision Documentation Model according to [8]

All knowledge elements concerned with particular aspects of decision knowledge are called *decision knowledge elements*. The basic element is *Decision*, which contains all related decision knowledge elements as *DecisionComponents*. Decision knowledge elements can be added incrementally over time by different *Persons*. Decisions and DecisionComponents can be linked to other knowledge elements, for instance to requirements or design artifacts. Moreover, different kinds of DecisionComponents are distinguished to describe the decision's *Problem* and *Solution*, its *Context* and *Rationale*. Problem elements contain details on the necessity to make a decision, for instance as *Issues* or *Goals* to be addressed by a decision. Solution elements contain options for the decision, like a *Claim* on how to solve a problem or different *Alternatives*. Context elements represent information on the environment of the decision and its knowledge elements. Such information can be given by *Assumptions* influencing the decision, *Constraints* restricting the decision, or *Implications* resulting from different alternatives. Rationale elements contain reasons related to other decision knowledge elements, such as *Arguments* for or against an alternative or their justification by an *Assessment* of criteria.

Investigated Data The investigated data are transcripts of three design sessions, which were initially distributed as material for the international workshop "Studying Professional Software Design" in 2010 [9]. In all sessions, the teams received a textual description of their task. They were given one hour and fifty minutes to create a high-level system design for a traffic simulation system. The task description contained a set of briefly described requirements for the simulation system. An overview of these requirements is given in Table 1. The requirements cover different aspects of the system model, such as the representation of intersections, lights, traffic sensors and traffic simulation. We will refer to these aspects as the *System* category. In addition, the interaction of the users with the system is described. For instance, the users shall control the traffic simulation or traffic density. We will refer to this as the *Interaction* category.

The designers were instructed to use a whiteboard for any drawings or notes, but no other instructions were given. Each session was held by two professional software designers and recorded on video as well as transcribed by the workshop organizers. We have investigated two transcripts with designers from Adobe and Amberpoint. The third transcript was not investigated, as the respective session was shorter than the others and deviated in conditions.

Related Studies In several studies on design decision-making (cf. [5,22,25]) complex decisions from real-world projects were investigated. These studies focus on the process of decision-making and the applied decision-making strategies, but they did not consider the related requirements extensively. Ko and Chilana [13] investigate decisions in issue reports of different large open-source projects. Although they evaluate design decisions with more fine-grained knowledge structures, they assess requirements only in a limited way by software qualities.

Table 1. Summary of Requirements Given to the Designers

No.	Content of Requirement
Functional Requirements	
R-I	Enable students to create a visual map with at least six intersections and roads of varying length as simulation area.
R-II	Enable students to describe the behavior and timing of traffic lights; the system shall allow for left-hand green arrow lights.
R-II.a	Combination of traffic lights, which result in crashes, are not allowed.
R-II.b	Every intersection on the map is a 4-way intersection and has traffic lights.
R-II.c	Enable students to choose for each intersection to have sensors, which trigger the traffic lights.
R-III	Enable students to simulate traffic flows on the map in real-time; the system shall depict the traffic flows and traffic light states.
R-IV	Enable students to change the density of traffic entering the simulation.
Non-functional Requirements	
R-V	The system shall be easy to use.
R-VI	The system shall motivate the students to explore the simulation.
R-VII	The system design shall be elegant.
R-VIII	The system design shall be clear.

Further related studies originate from approaches concerned with design decision documentation or requirements traceability using decision knowledge. Most approaches on design decision documentation (cf. [14,23,24]) only present small examples of how they can be applied. So, they do not offer realistic and complex data in their case studies. Some approaches on requirements traceability, for instance as described by Cleland-Huang at el. [4], use decisions to create trace links between requirements and other artifacts, like design diagrams or code. Thus, they do not represent fine-grained structures of decision knowledge for their trace links.

Several studies have been executed based on the introduced design session transcripts. They can be found in special issues of *Design Studies* in 2010 and *Software* in 2012 as well as in [18]. For instance, the studies of Jackson [10] and Shaw [20] analyze the design structures and the explored design space. The studies of Tang et al. [21] and Baker and van der Hoek [1] investigate the decision-making process. Mostly, the applied research method in these studies is similar to our study, as the transcripts were analyzed by coding relevant text parts according to given coding schemes. Only the study of Ball et al. [2] explicitly considers relations between requirements and decision knowledge. The given requirements are grouped according to their level of complexity and examined for relations to different design strategies. However, no study is investigating in detail how particular decision knowledge elements are related to requirements.

3 Research Method

In this section, we present our research method. First, we introduce our *research questions* for the study and then define the *coding table* for the text analysis of the transcripts. Finally, we briefly describe the *coding process*.

Research Questions According to our overall goal (cf. Section 1), we aim to investigate relations between requirements and design decisions at a fine-grained level. Consequently, the first research question *RQ1* is: *Which relationships exist between requirements and decision knowledge elements?* Such fine-grained relations are likely to influence the evolution of decision knowledge structures over time. For instance, constraints based on specific requirements might restrict solution alternatives, so that new implications for the decision arise. This leads to *RQ2*: *How do fine-grained knowledge structures emerge based on the given requirements?* As pointed out by Chen et al. [3], requirements with significant influence on a system's design often are difficult to define and tend to be vaguely described. They report, that designers then make assumptions about the missing details. Thus, we address uncertainties about requirements in *RQ3*: *How do decision knowledge elements address uncertainty about requirements?* Among other reasons, these uncertainties might impact the given requirements by triggering their extension or other refinements. Therefore, we also investigate the impact of decision knowledge elements on requirements by *RQ4*: *How do decision knowledge elements impact and refine the given requirements?*

Coding Table and Coding Process Based on the leaf entities and relations in our documentation approach, we derived a coding table to identify the different decision knowledge elements and their relations to requirements within the transcripts. All codes are given in Table 2. A general code *Context* was added to capture context knowledge, which could not be categorized in detail. As an argument may support or challenge other knowledge elements, two different codes were created for arguments. For each identified decision knowledge element a unique running ID and a name were created. Elements with a late position in the transcript got a higher ID. The ID as well as the numbers of the requirements presented in Table 1 were used to express relations. We used 41 decisions of the design space described by Shaw [20] as a high-level structure, with 20 decisions belonging to the Adobe transcript and 21 to the one for Amberpoint. All identified decision knowledge elements were either contained in such a decision or in another decision knowledge element.

The first author coded both transcripts completely. The first 10% of the data was also coded by the second author and both codings were compared. The authors discussed any deviations and further refined the coding table and the criteria for setting a code. Then, the first author coded the remaining data. A coding example is given in Table 3.

Table 2. Codes for Transcript Analysis

RQ	Code	Description
1	DKE.*concerns*(R-x)	Reference to a requirement; code was set according to keywords, like "traffic lights", "sensor" or "rate of traffic"
2	*Issue* / *Goal*	Concrete open question / Abstract, more general aim
2	*Alternative* / *Claim*	Solution proposal: Can be assessed by criteria / is based on personal experience, informal knowledge
2	*Context* / *Assumption* / *Constraint* / *Implication*	General information / Uncertain or approximated information / Limitation or restriction / Consequence
2	*pro-Argument* / *contra-Argument* / *Assessment*	Information supports / challenges / assesses another knowledge element
2	DKE.(Decision|DKE)	Element contained in decision or another element
3	*Uncertain*(Description)	Developers explicitly express uncertain or vague information about the given requirements
4	*Impact*(Description)	Developers explicitly express extensions to or limitations of given requirements

DKE: ID of decision knowledge element, R-x: requirement number

Table 3. Transcript Excerpt with Coding Example

Transcript	Yes, so it's got an infinite number of roads and intersections you can lay out. ID: 6
Code RQ1	6.*concerns*(R-I)
Code RQ2	*Assumption.*(Decision "Intersections implied by road crossings")
Code RQ3	-
Code RQ4	*Impact*(it's got an infinite number of roads and intersections)

4 Results

In this section, we present the results of our coding. The percentages of all decision knowledge elements per transcript are depicted in Figure 2. In total we found 182 decision knowledge elements with 55 relations to requirements in the Adobe transcript and 198 decision knowledge elements with 65 relations to requirements in the Amberpoint transcript. For the Adobe transcript, higher percentages of *Issues*, *Claims* and *Implications* were found than for the one of Amberpoint. In contrast, the Amberpoint team made more *Arguments* explicit in their discussions. Only one explicit *Assessment* of different alternatives was found in the Adobe transcript. It should be noted that both teams followed different solution approaches, as described by Shaw [20]. This difference is illustrated by the 5 decisions with the most decision knowledge elements for each team, as shown

in Table 4. Whereas the Adobe team focused on the system's functionality and technical architecture using a Model-View-Controller-approach, the Amberpoint team was mostly concerned with designing the user interface and interaction behavior of the system.

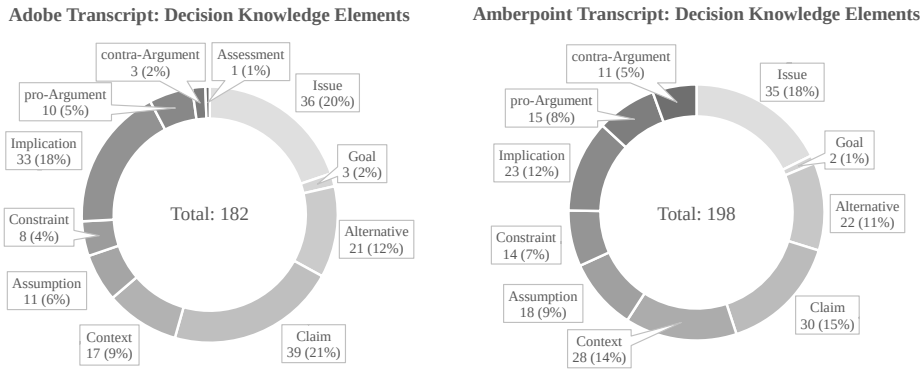

Fig. 2. Percentages of Decision Knowledge Elements for each Transcript

Table 4. Decisions with Most Decision Knowledge Elements (DKE) for each Transcript

Adobe Decisions	**#DKE**	**Amberpoint Decisions**	**#DKE**
Set of objects – traversed by a controller at each clock tick	25	Discrete cars – Cars with state, route, destination	26
Intersections – Have roads (with lights and cars)	22	Intersections – Signals and sensors in approaches	23
High-level organization – Network	17	Connection of roads to intersections – Lights and sensors in approaches	20
Place in hierarchy – Traffic signals belong to roads	16	Traffic Model – Master traffic object, discrete cars	20
Layout of visual map – Intersections implied by road crossings	12	System Concept – User Interface	16

Results for RQ1: Relations between Requirements and Decision Knowledge Elements The detailed percentages of relations to requirements for each kind of decision knowledge element are presented in Figure 3.

The Adobe team addressed all functional requirements in their decision knowledge, but they did not explicitly refer to any non-functional requirement. In contrast, for the Amberpoint transcript no relations to requirement II.b "Only 4-way

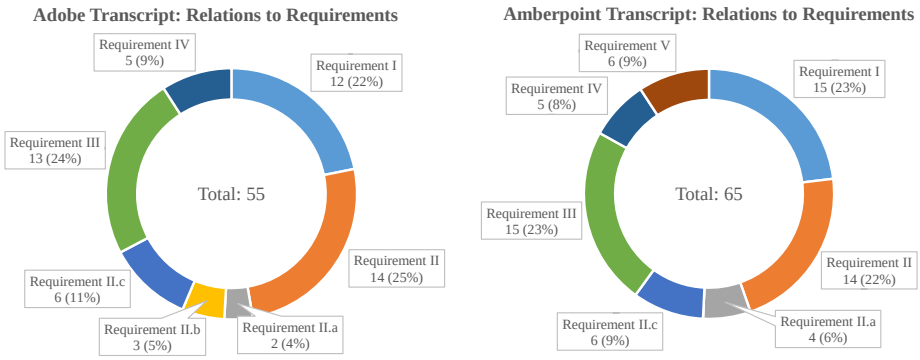

Fig. 3. Percentages of Relations to Requirements for each Transcript

intersections" could be identified, but the non-functional requirement V "Usability" was addressed. However, a pattern for both teams is that non-functional requirements were mostly not referenced in the investigated design decisions. When comparing both teams, several percentages for relations to requirements and decision knowledge elements are similar. Looking at the requirements addressed by the teams and the relationships of these requirements to decision elements, we found differences between the teams. This is depicted in Figure 4.

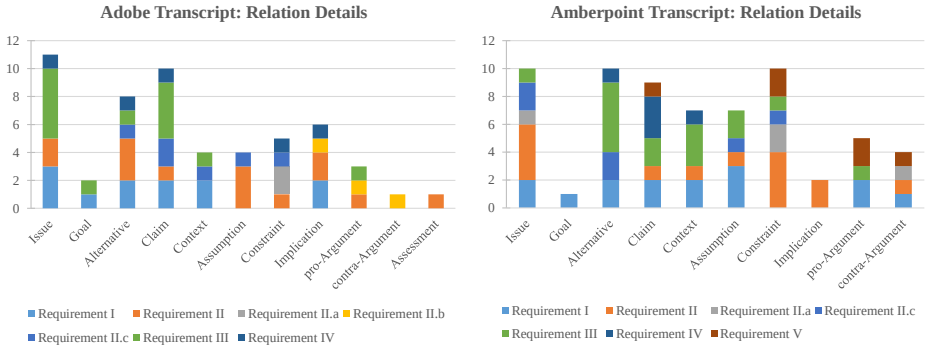

Fig. 4. Detailed Percentages of Relations to Requirements by Decision Knowledge Element

For instance, for the Adobe team we found three references of requirement II within *Alternatives*, but no *Assumptions* related to requirement I. In contrast, the Amberpoint transcript contained no references to requirement II for *Alternatives*, but several *Assumptions* were related to requirement I. Looking at the the relationships to requirements altogether, we found for both teams that requirements were mostly related to *Issues*, *Alternatives* and *Claims*. In contrast, links

between requirements and context elements seem to be specific for each team. Looking at the relationships aggregated for entire decisions, we also observed differences, as shown in Table 5. For instance, many references to requirements were found for the decision on the place for traffic lights in the system's hierarchy made by the Adobe team. However, the Amberpoint team did not consider any requirement explicitly for the same decision. In general, this indicates that references on requirements in the investigated design decisions depend not only on the actual content of the requirements, but also on the preferences and priorities of the team.

Table 5. Major Deviations for the Number of Relations to Requirements per Decision

Decision	**#Relations for Adobe**	**#Relations for Amberpoint**	Δ
Road System – Connection of roads to intersections	3	13	**10**
Traffic Lights – Place in hierarchy	10	0	**10**
Simulator	7	0	**7**
Road System – Intersections	3	9	**6**
System Concept	2	8	**6**

Results for RQ2: Emerged Decision Knowledge Structures We identified complex structures of decision knowledge elements in the transcripts, which emerged based on the given requirements. During the design discussions, statements of the developers jumped between multiple decisions. However, for sequences of decision knowledge elements they typically addressed the same requirement. Different examples of resulting knowledge structures are depicted in Figure 5. In most decisions problem (*Issues*, *Goals*) or solution elements (*Alternatives*, *Claims*) initially addressed requirements. Over time, they were accompanied by context knowledge. This reflects the further exploration of the decision and its environment by the designers. An example is shown in part (a) of Figure 5. More complex knowledge structures addressed multiple requirements within one decision due to context elements related to different requirements. A potential cause could be that designers aim to satisfy multiple requirements within one decision. Then, they start to make trade-offs between alternatives by adapting and extending the alternatives over time. An example is given as part (b) in Figure 5. Typically, *Solutions* were not formally assessed according to given criteria, as only one *Assessment* was found. Instead, one or more particular *Arguments* were stated by the designers to support or challenge a *Solution*. For the Amberpoint transcript, several of these *Arguments* also were explicitly related to requirements. A reason could be that often designers prefer sufficient solutions over optimal to reduce their effort [25]. Then, only the most important

arguments are considered. An example for this structure is depicted as part (c) in Figure 5.

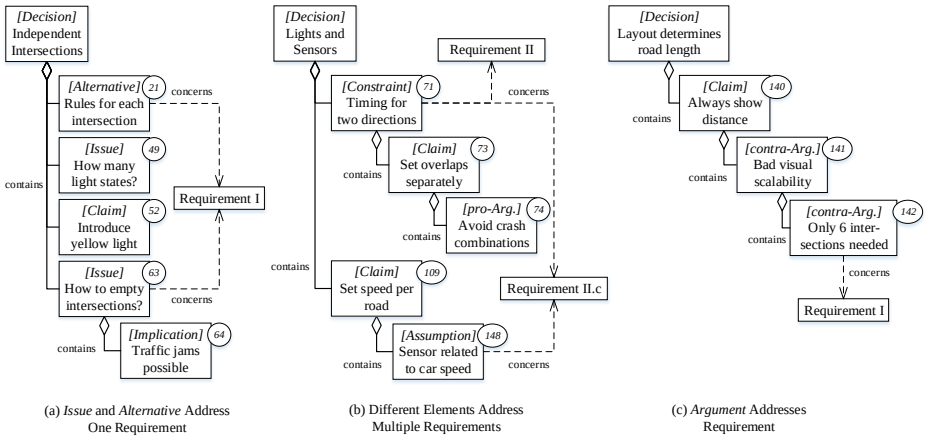

Fig. 5. Excerpts of Emerged Knowledge Structures (with Element IDs in Circles)

Results for RQ3: Uncertainty about Requirements in Decision Knowledge Elements In total, we identified 21 different uncertainties by the code *Uncertain*. We sorted them according to the affected category of requirements. The teams explicitly stated these uncertainties in their decision knowledge elements. An overview is given in Table 6.

Table 6. Uncertainty about the Given Requirements in Decision Knowledge Elements

Category	**Example of Uncertainty**
Interaction – Simulation	"[...] I don't know if there'd be two modes: an editing mode and a simulation mode." *[Assumption]*
Interaction – Traffic Density	"[...] I'd go back to the customer and try and figure out, how did they collect this [traffic] data [...]" *[Assumption]*
System – Intersections	"But we are assuming straight lines." *[Assumption]*
System – Lights	"The left-hand turns are protected, but does it have only left-hand [turns]?" *[Issue]*
System – Sensors	"[...] If you have [a] sensor, what does that mean?" *[Issue]*
System – Simulation	"How do you assess the success of the timing?" *[Issue]*

Typically, uncertainties were addressed in *Assumptions* and *Issues*. We identified multiple uncertainties the designers had about the user's interaction with

the simulation system. In addition, both teams discussed uncertainties about the capabilities and limitations of different entities implied by the requirements, such as intersections, lights and sensors. Moreover, the Amberpoint team explicitly stated several uncertainties about how parts of the simulation functionality should be addressed in their decisions. Overall, in the Amberpoint transcript we identified 15 explicitly addressed uncertainties about requirements in decision knowledge elements and 6 in the Adobe transcript.

Results for RQ4: Impact of Decision Knowledge Elements on Requirements In total, we identified 15 decision knowledge elements to impact the given requirements by the code *Impact*. Four elements were found in the Adobe transcript and 11 were contained in the Amberpoint transcript. An overview is given in Table 7.

Table 7. Impact of Decision Knowledge Elements on the Given Requirements

Category	**Example of Impact**
Interaction – Simulation Configuration	"[...] where you put these roads determines the maximum number of cars [...]" *[Implication]*
Interaction – Simulation Usage	"[...] it could be you can draw while you're simulating." *[Claim]*
Interaction – User Groups	"[...] the end users seem to be the students, and the professor." *[Assumption]*
System – Intersection Structure	"[...] it's got an infinite number of roads and intersections [...]" *[Assumption]*
System – Simulation Analysis	"[...] then you have an analytics piece looking in and assessing questions [...]" *[Issue]*

Eight out of 15 decision knowledge elements impacting the requirements were context elements. This indicates, that information from the decision context could be a trigger to refine and adapt the given requirements. Adaptions to requirements were made either by extension or restriction. Both teams discussed extensions to the given requirements in decision knowledge elements of different kinds. For instance, the Amberpoint team explicitly addressed professors as a potential user group for the system within an *Assumption*. However, this was not requested by the requirements in the prompt. In contrast, restrictions to the given requirements typically were expressed in *Solution* and *Implication* elements. For instance, the Adobe team reasoned that the road layout determines the maximum number of cars possible for the roads. This describes a potential limit of requirement IV, which requests the designers to let the users control the traffic density without limitations resulting from the physical capacity of the road.

5 Discussion

In the following paragraphs, the presented results are discussed with respect the given research questions. In addition, we describe how we have addressed potential threats to validity for our study. Our documentation approach for decision knowledge enables to uncover knowledge structures in given design decisions. The discovered differences between both transcripts were not expected. It should be noted that our study is not representative, as we have investigated the transcripts of two specific design sessions only. However, we gathered valuable insights that should be further investigated in replicated and large-scale studies.

Summary of Results for Our Research Questions The results for our research questions show a diverse picture of the relations between requirements and decision knowledge for the two given design session transcripts. On the one hand, for both teams many similar percentages of requirement relations and decision knowledge elements in total were found. Both teams mainly focused on the given functional requirements in their design decisions. Also the decision knowledge structures showed similarities, and both teams explicitly considered uncertainties and refinements of requirements in their decision knowledge elements. On the other hand, relations between requirements and particular decision as well as kinds of decision knowledge elements strongly differ. In addition, the teams chose different solution approaches within their design and expressed different amounts of requirement uncertainties and impacts in their decision knowledge elements. They also stated these uncertainties and impacts in different kinds of knowledge elements. Overall, the coarse-grained decision knowledge structures and relations to requirements appear to be similar for both teams, whereas the more fine-grained decision knowledge elements with their particular relation to requirements deviate.

Recommendations Derived from the Results Our findings show that for the investigated design discussions many relations were found between the given functional requirements and decision knowledge elements.

Our study also confirms the well-known fact that designers should more explicitly consider the non-functional requirements for their design within their decisions, as non-functional requirements were mostly not related to decision knowledge elements. In addition, we found relations to requirements to depend more on the preferences and priorities of the team, than on the actual content of the requirements. These two insights represent patterns for decision knowledge structures, which are likely to decrease the quality of design decisions. Therefore, these patterns should be avoided by designers.

Moreover, relations between requirements and decision knowledge elements should be made explicit, so that both requirements engineers and designers can assess the importance of particular requirements for the design. For instance, relations to requirements in problem and solution elements might indicate the significance of those requirements for the design outcomes. This would be in line

with the characteristics described by Chen et al. [3] for architecturally significant requirements. Thus, designers could use these relations to recognize architecturally significant requirements more easily.

Next, requirements engineers would benefit if designers clearly stated how they want to address uncertainties about requirements. Our findings show that different kinds of decision knowledge elements, like *Assumptions* or *Issues*, were expressed due to such uncertainties. If designers explicitly noted these uncertainties and marked them as a prerequisite for a decision, valuable feedback for the requirements engineering process could be derived. For instance, such uncertainties and their impact on the design could be discussed with stakeholders to avoid a misalignment between requirements and design.

Moreover, other approaches could be extended with our insights. Goal modeling techniques, like i* or GQM, are concerned with the exploration of different alternatives for implementing given requirements [12]. These approaches could be extended to explicitly cover decisions with their relations to requirements and design artifacts. For instance, description templates for goals could be extended with a decision section, representing design decisions made to achieve a goal.

Overall, we advocate to integrate developers more closely into the requirements engineering process. Requirements engineers and developers should enter a dialogue, which could be guided by documented decisions. Then, follow-up questions on requirements by developers during the implementation can be addressed by requirements engineers.

Insights for Our Documentation Approach From the results for RQ1 and RQ2 we conclude that our documentation approach for decision knowledge proved to be capable of capturing decision knowledge structures and their relations to requirements in a comprehensive and fine-grained way. In addition, the results for RQ3 and RQ4 indicate that our documentation approach helps to identify the mutual impact of requirements and design on each other. However, such detailed documentation is not realistic for all design decisions under real-world settings due to the required analysis effort and the differing importance of decisions. Thus, it is important to support developers, so that they can document relevant decisions with less effort. We propose to support requirements engineers and designers by semi-automatically documenting specific decisions for given requirements, such as decisions concerned with security [6]. In addition, developers could be supported by code annotations for decision knowledge to integrate decision documentation with implementation [7].

Threats to Validity According to Runeson et al. [19], we discuss four different types of threats to validity for our study.

Internal validity is concerned with the correlation between the investigated factors and other factors [19]. First, the decision knowledge expressed by the designers might have been influenced by missing further instructions on documentation or design reasoning. Thus, the designers might have worked less structured and did not articulate all decision-related thoughts. However, this corresponds to work conditions in practice. If the designers had been asked to apply specific

methods or structured processes, our results for the design decisions would depend on those methods or processes. Second, relations to requirements might have been impacted by the rather short design prompt. This might have caused additional uncertainties, which were not related to the content of a requirement, but to its description in the prompt. We addressed this threat by deriving core keywords for the content of each requirement, and used these keywords for coding relations to requirements.

External validity is concerned with the degree to which the results of our study can be generalized [19]. We only investigated transcripts of two design sessions. Therefore, our findings depend on the designers of the two investigated teams and might not be generalized for or comparable to other teams. We could have included the third transcript of the UCI design workshop in our analysis, but this would have resulted in more threats to internal validity due to the deviations in the session's settings. In addition, the involved designers were professionals from industry and had key roles in their respective companies. In consequence, the investigated data and our results are likely to represent typical design sessions and their decision knowledge.

Construct validity is concerned with any gaps between intended and actual observations of the researchers [19]. Our coding table could have identified something else than decision knowledge elements. We mitigated this threat by testing and refining the codes in previous coding experiments. Also the fit with regard to the decisions identified by Shaw [20] was very good. With our coding, we have covered 18 out of 20 decisions for the Adobe transcript, and 20 out of 21 decisions for the Amberpoint transcript. In addition, in another project we have investigated comments in issue reports within the Firefox project for decision-related knowledge. There, we applied the coding table presented in this paper successfully. As the transcripts of design discussions and discussions within issue comments are similar in structure and content, we reached a good fit of our documentation approach with the contents given in the transcripts. Moreover, our documentation approach is based on other fundamental approaches for documenting decision knowledge, as described in [8].

Reliability validity is concerned with the degree to which data and analyses of a study are dependent on specific researchers [19]. Only one coder coded all data from the transcripts, so that the codes set by this coder might not be reliable. We addressed this threat with checks and code alignments, as a second coder also coded data samples from the transcripts. Small parts of the design discussions were inaudible in the videos and, therefore, were marked and left out in the transcripts. Thus, relevant decision knowledge might have been missed in our analysis. We mitigated this threat by checking the surrounding text of any inaudible passage for hints on the missing content.

6 Conclusion and Future Work

In this paper, we have presented a study on discussion transcripts of two design sessions with professional software designers. We have investigated the tran-

scripts for any contained decision knowledge and its relations to the given requirements. Therefore, we have coded the transcript texts according to a defined coding scheme. Designers addressed the given functional requirements in their design decisions, so that complex structures of decision knowledge emerged. Moreover, decision knowledge elements also contained uncertainty about the given requirements and impacted them with extensions or restrictions. This shows the mutual impact of requirements and decision knowledge elements on each other. It also points out that designers might benefit from making relations between requirements and decision knowledge elements explicit. Then, these knowledge elements could provide valuable feedback for the requirements engineering process and help to clarify and further improve the requirements.

As future work, it should be further investigated how designers can be supported in making the most important decision knowledge elements explicit. This requires research in two directions. First, the current study should be repeated in larger scale. Additional design sessions could be analyzed to further refine our findings. Second, the results of this study could be used to improve the tool support for our decision documentation approach. For instance, the tool for code annotations could ask developers for relations to and uncertainties about the requirements when they are documenting decision knowledge elements.

Acknowledgment

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future — Managed Software Evolution. Results described in this paper are based upon videos and transcripts initially distributed for the 2010 international workshop "Studying Professional Software Design", as partially supported by NSF grant CCF-0845840.

References

1. Baker, A., van der Hoek, A.: Ideas, subjects, and cycles as lenses for understanding the software design process. Design Studies 31(6), 590–613 (2010)
2. Ball, L.J., Onarheim, B., Christensen, B.T.: Design requirements, epistemic uncertainty and solution development strategies in software design. Design Studies 31(6), 567–589 (2010)
3. Chen, L., Ali Babar, M., Nuseibeh, B.: Characterizing Architecturally Significant Requirements. Software 30(2), 38–45 (2013)
4. Cleland-Huang, J., Mirakhorli, M., Czauderna, A., Wieloch, M.: Decision-Centric Traceability of Architectural Concerns. In: International Workshop on Traceability in Emerging Forms of Software Engineering. pp. 5 – 11. IEEE (2013)
5. Falessi, D., Cantone, G., Kazman, R., Kruchten, P.: Decision-making techniques for software architecture design. ACM Computing Surveys 43(4), 1–28 (2011)
6. Hesse, T.M., Gaertner, S., Roehm, T., Paech, B., Schneider, K., Bruegge, B.: Semiautomatic Security Requirements Engineering and Evolution using Decision Documentation, Heuristics, and User Monitoring. In: First International Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE) at RE2014. pp. 1–6. IEEE (2014)

7. Hesse, T.M., Kuehlwein, A., Paech, B., Roehm, T., Bruegge, B.: Documenting Implementation Decisions with Code Annotations. In: 27th International Conference on Software Engineering and Knowledge Engineering. pp. 152–157. KSI Research Inc. (2015)
8. Hesse, T.M., Paech, B.: Supporting the Collaborative Development of Requirements and Architecture Documentation. In: 3rd Internatioal Workshop on the Twin Peaks of Requirements and Architecture (TwinPeaks) at RE2013. pp. 22 – 26. IEEE (2013)
9. van der Hoek, A., Petre, M., Baker, A.: Workshop "Studying Professional Software Design" at University of California, Irvine (2010), `http://www.ics.uci.edu/design-workshop/`, URL retrieved in 10-2015
10. Jackson, M.: Representing structure in a software system design. Design Studies 31(6), 545–566 (2010)
11. Jansen, A., Bosch, J.: Software Architecture as a Set of Architectural Design Decisions. In: 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05). pp. 109–120. IEEE (2005)
12. Kavakli, E., Loucopoulos, P.: Goal Modeling in Requirements Engineering: Analysis and Critique of Current Methods. Information Modeling Methods and Methodologies: Advanced Topics of Database Research pp. 102–124 (2005)
13. Ko, A.J., Chilana, P.K.: Design, discussion, and dissent in open bug reports. In: Proceedings of the 2011 iConference. pp. 106–113 (2011)
14. Kruchten, P., Lago, P., Vliet, H.V.: Building Up and Reasoning About Architectural Knowledge. In: Hofmeister, C., Crnkovic, I., Reussner, R. (eds.) Quality of Software Architectures, Lecture Notes in Computer Science, vol. 4214, pp. 43–58. Springer (2006)
15. Ngo, T., Ruhe, G.: Decision Support in Requirements Engineering. In: Engineering and Managing Software Requirements, pp. 267–286. Springer (2005)
16. Nuseibeh, B.: Weaving Together Requirements and Architectures. Computer 34(3), 115–119 (2001)
17. Paech, B., Delater, A., Hesse, T.M.: Integrating Project and System Knowledge Management. In: Ruhe, G., Wohlin, C. (eds.) Software Project Management in a Changing World, pp. 157–192. Springer (2014)
18. Petre, M., van der Hoek, A.: Software Designers in Action: A Human-Centric Look at Design Work. CRC Press (2013)
19. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering. Guidelines and Examples. Wiley (2012)
20. Shaw, M.: The role of design spaces. Software 29(1), 46–50 (2012)
21. Tang, A., Aleti, A., Burge, J., van Vliet, H.: What makes software design effective? Design Studies 31(6), 614–640 (2010)
22. Tang, A., Babar, M.A., Gorton, I., Han, J.: A Survey of Architecture Design Rationale. Journal of Systems and Software 79(12), 1792–1804 (2006)
23. Tang, A., Jin, Y., Han, J.: A rationale-based architecture model for design traceability and reasoning. Journal of Systems and Software 80(6), 918–934 (2007)
24. Tyree, J., Akerman, A.: Architecture Decisions: Demystifying Architecture. Software 22(2), 19–27 (2005)
25. Zannier, C., Chiasson, M., Maurer, F.: A model of design decision making based on empirical results of interviews with software designers. Information and Software Technology 49(6), 637–653 (2007)