Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2012 Proceedings

European Conference on Information Systems (ECIS)

5-15-2012

TWENTY SOFTWARE REQUIREMENT PATTERNS TO SPECIFY RECOMMENDER SYSTEMS THAT USERS WILL TRUST

Axel Hoffmann
Kassel University

Matthias Söllner
Kassel University

Holger Hoffmann
Kassel University

TWENTY SOFTWARE REQUIREMENT PATTERNS TO SPECIFY RECOMMENDER SYSTEMS THAT USERS WILL TRUST

Hoffmann, Axel, Information Systems, Kassel University, Pfannkuchstr. 1, 34121 Kassel, Germany, axel.hoffmann@uni-kassel.de

Söllner, Matthias, Information Systems, Kassel University, Pfannkuchstr. 1, 34121 Kassel, Germany, soellner@uni-kassel.de

Hoffmann, Holger, Information Systems, Kassel University, Pfannkuchstr. 1, 34121 Kassel, Germany, holger.hoffmann@uni-kassel.de

Abstract

Trust has been shown as a crucial factor for the adoption of new technologies. Surprisingly, trust literature offers very little guidance for systematically integrating the vast amount of insights from behavioral research on trust into the development of computing systems. The aim of this article is to translate results from behavioral sciences into software requirement patterns that address user trust in recommender systems. Software requirement patterns are used in requirements engineering to recognize important and recurring issues and reduce the effort of compiling a list of software requirements. We collected antecedents that build trust, and developed software requirement patterns that demand functionality to support these antecedents. This paper contributes by presenting software requirement patterns consisting of the name, the goal and the pre-defined requirement template that can be used to specify trust requirements in recommender system development projects.

Keywords: Software Requirement Patterns, Requirements Reuse, Trust Requirements

1 Introduction

Trust has been shown as an important factor for the adoption of new technologies (Gefen et al., 2003). As early as 30 years ago Luhmann stated: *"One should expect trust to be increasingly in demand as a means of enduring the complexity of the future which technology will generate"* (Luhmann, 1979). The technologies we are using are getting more and more automated and opaque (Lee and See, 2004), and thus we continue to lose our ability to know what exactly happens inside the system. Hence, we perceive us to be decreasingly able to control the systems we are using. On the other side we use more and more systems that recommend items (e.g., books, music, movies) to us. Therefore, recommender systems help us to reduce the number of alternatives given and to make a pleasant decision. In this research we use recommender system as an example for systems that do depend on trust.

To address the increasing demand for trust, numerous researchers have called for systematic ways to enhance users' trust in IT systems (Leimeister et al., 2005). Users' trust can be addressed throughout the whole lifecycle of an IT system. This paper shows how trust can be addressed from the very beginning of system development. The first step towards an IT system is the elicitation of the requirements. Therefore, requirements analysts talk to customers and stakeholders, review old system specifications, analyze business processes and so on (Berkovich et al., 2011, Sommerville, 2007). While this approach works well for wishes and expectations that can be made explicit by the user, or implicit requirements that can be made explicit by special means (e.g., prototyping), trust is a fuzzy concept and there are only few guidelines (e.g., Patrick et al., 2005) that help requirements analysts to consider the trustworthiness of IT systems. Diverse interests in trust have generated many definitions. Moreover, trust can be considered from various standpoints as well as different points in time – e.g., initial trust before the user used the system to trusting a system to make a change from a known system to a new one, etc. As a result, a deep and broad understanding of different concepts of trust is necessary to be able to use the various facets of trust to deduct requirements for system functionality that enhances the user trust in the recommender system.

An existing approach that requirements analysts use to reduce the effort of acquiring requirements are software requirement patterns. A pattern, in general, describes a common problem and the core of a solution to that problem (Alexander et al., 1977, Alexander, 1979). The problem we face is the enhancement of user trust in systems that depend on trust like recommender systems. Our proposed solutions are requirement templates that can be used in requirement specifications that should be considered in the following system development. The aim of this paper is to present software requirement patterns consisting of the name, the goal and the pre-defined requirement template that can be used in system development projects. Thus, the theoretical contribution type is Design and Action (Gregor, 2006).

The remainder of the paper is organized as follows. First of all, we give an overview of the related work in trust theory and software requirement patterns. Next, we briefly describe how trust requirements for IT systems are derived in trust engineering. After a description of the research design in section 4 we present twenty software requirement patterns to enhance user trust in recommender systems in section 5. This is followed by the discussion and conclusion.

2 Related Work

This section summarizes former work related to trust or software requirement patterns. We first describe trust and the challenges it raises in the development process. We briefly illustrate the trust engineering method that serves as the foundation for our research approach. Next, we elaborate on the use of software requirement patterns in requirements specification.

2.1 Trust

Since the late 1990s the interest in trust research has greatly increased. This is evident in publications of several special issues in major journals in: Human–computer Interaction (HCI) and Information Systems (IS) (Benbasat et al., 2008, Benbasat et al., 2010). The main value of trust is that it serves as a mechanism to reduce complexity (Luhmann, 1979). This becomes important for many disciplines because of the increasing complexity of organizations and technology (Lee and See, 2004). With various disciplines using trust in different contexts, trust is widely used, and the interpretations of trust become multifarious (Ebert, 2009), resulting in a plethora of definitions.

The most common approach is to define trust as an intention or willingness to act. This approach is also followed by most IS trust researchers, who rely on the most widely used and accepted definition of trust by Mayer et al. (1995): "*trust […] is the willingness of a party [trustor] to be vulnerable to the actions of another party [trustee] based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party*."

The definition by Mayer et al. (1995) focuses on trust between people, groups of people, or organizations. Thus, they are especially valuable for areas of IS research dealing with different kinds of computer-mediated relationships between people. Further, IT artifacts serve as a tool for users to achieve a desired goal. Therefore, a second stream of IS research studies trust relationships between people and IT artifacts (Wang and Benbasat, 2005). They argue that IT artifacts can be compared to humans, thus making the existing definitions of trust suitable for researching trust relationships between people and IT artifacts (Wang and Benbasat, 2005).

2.2 Trust Engineering

For developing the software requirement patterns, trust literature was reviewed that focused on the insights of how trust develops and when trust becomes important. There are only a few methods that systematically address trust in the development process of IT systems (trust engineering, e.g., Söllner et al., 2011a, Söllner et al., 2011b). Due to the fact that we use the foundation of trust engineering to formulate the software requirement pattern, we briefly describe the approach. Trust engineering emphasizes that trust is only important in situations of uncertainty. Based on these insights, antecedents of trust that counter these uncertainties need to be identified from theory. Trust antecedents are factors that build trust. The trust engineering method starts with a definition of the intended use of the information system. This step is required because it is necessary to identify the uncertainties the user faces during the interaction process, and these uncertainties depend upon the intended use of the information system. Afterwards, the uncertainties are identified – e.g., by the designers of the systems or using interviews with future users of the system. Additionally, the uncertainties are prioritized with regard to their negative impact on the possibility of achieving the intended goal of the application. After having identified and prioritized the uncertainties, the dimensions of trust are identified that can be used to address single uncertainties. The next step zooms deeper into the single dimensions and one or more antecedents that are suited to address if a single uncertainty should be found. After the uncertainties and single antecedents of trust are matched, trust supporting requirements are formulated. In the final step, detailed trust supporting components are derived, based on the trust supporting requirements.

The core of the trust engineering approach is that it is more effective to address the antecedents of trust in the development process than to address trust itself. Considering components and antecedents of trust Patrick et al. (2005) extracted a composite set of design guidelines (Table 1) from literature. These guidelines should make it "easier for designers to identify those elements capable of promoting trust"(Patrick et al., 2005). With our research we want to pick up this goal and combine it with results from trust engineering by providing concrete software requirement patterns that address trust antecedents.

1. Ensure good ease of use.
2. Use attractive design.
3. Create a professional image - avoid spelling mistakes and other simple errors.
4. Don't mix advertising and content - avoid sales pitches and banner advertisements.
5. Convey a "real-world" look and feel-for example, with the use of high-quality photographs of real places and people.
6. Maximize the consistency, familiarity, or predictability of an interaction, both in terms of process and visually.
7. Include seals of approval such as TRUSTe.
8. Provide explanations, justifying the advice or information given.
9. Include independent peer evaluation such as references from past and current users and independent message boards.
10. Provide clearly stated security and privacy statements, and also rights to compensation and returns.
11. Include alternative views, including good links to independent sites within the same business area.
12. Include background information such as indicators of expertise and patterns of past performance.
13. Clearly assign responsibilities (to the vendor and the customer).
14. Ensure that communication remains open and responsive, and offer order tracking or an alternative means of getting in touch.
15. Offer a personalized service that takes account of each client's needs and preferences and reflects its social identity.

Table 1. Trust design guidelines (Patrick et al., 2005)

2.3 Requirements Reuse and Software Requirement Patterns

Reuse is an established practice in software engineering. In requirements engineering , reuse can help requirements analysts to elicit and document software requirements (Robertson and Robertson, 2006). Software requirement patterns are a worthwhile approach to reuse requirements (Franch et al., 2010). A pattern, in general, describes a problem which occurs over and over again, and then describes the core of the solution to that problem in such a way that it can be used a million times over, without ever doing it the same way twice (Alexander, 1979). Software requirement patterns are used for the software analysis stage. There are different approaches that differ in scope, notation and application (Franch et al., 2010, Henninger and Corrêa, 2007). Recent approaches using software requirement patterns for writing software requirement specifications can be found in the work of Withall (2008) and in the Pattern-based Requirements Elicitation (PABRE) by Renault, Mendez-Bonilla, Franch, and Quer (Renault et al., 2009a, Renault et al., 2009b).

A pattern-based approach can reduce the effort of acquiring requirements for many development projects (Hoffmann et al., 2012). The possible benefits for requirements analysts are not only the reduction of time spent to perform the elicitation of the requirements, but also the improvement of the quality of the requirements book obtained (Renault et al., 2009b). For this reason, the reusability of software requirement patterns is the prerequisite for their applicability in practice.

3 Research Design

This section describes the research question, the unit of analysis and the research method. We seek to answer the research question if requirements to enhance user trust in recommender systems can be formulated as software requirement patterns. Therefore, we use a three step approach.

Trust engineering emphasizes that trust can be influenced in a more systematic and, thus, more effective way by influencing its antecedents. Therefore, we collected antecedents of trust in order to derive software requirement pattern from them. Due to the huge number of contributions on trust and many different proposed antecedents, we build on the results of a previous literature review collecting trust antecedents in leading journals that was conducted by Söllner and Leimeister (Söllner and Leimeister, 2010). We supplemented the list by the antecedents collected by Lee and See (2004) and

antecedents suggested by Muir (Muir, 1994) to have a good groundwork for the software requirements pattern.

Results from three requirements specifications, all which were archived from trust antecedents with the trust engineering method by Söllner et al. (2012), served as our source material. The documents were provided for our research. Given the documents containing trust requirements that address different antecedents, we followed the systematic approach of Withall (2008) to find candidates for requirement patterns. These documents contained four, seven and 24 trust requirements. We listed all requirements in a spreadsheet. If a requirement was similar to one we already had on the list, we noted that and moved on. For the identified requirements we formulated requirement patterns.

For antecedents from literature that we had no example requirements for in the requirements specifications we followed the opportunistic approach (Withall, 2008). Opportunistic means that we did not use given software requirements, but formulated software requirements on our own. Therefore, we reviewed the definitions of the antecedents given in the source literature and checked if it is possible to address this antecedent within system specification. We used the examples of antecedents and requirements pattern from the previous step and formulated analogously general requirement pattern.

The requirement patterns were reviewed by one requirements analyst and four software developers in a group discussion. They were asked to check if the requirements patterns were clear und applicable in the development process. This review was necessary, since both parties will use the patterns later on. The requirements analysts will use the patterns for deriving requirements. These requirements will be based on the templates as provided in the patterns. Consequently, software developers need to review whether the way the templates are formulated in a way they need requirements to be formulated. The requirement patterns were adapted at the end of the group discussion.

4 Results

Applying the research design from section 3 to our chosen example for recommender systems we derive software requirements pattern to specify recommender systems that users will trust. The requirements patterns address the antecedents of trust. Thus, trust can be influenced in a systematic and effective way.

Table 2 lists the trust antecedents we used for our research. Due to our research focus, we did not question the influence of single antecedents on trust. Further, the list mixes trust antecedents from two research streams. The first one has its roots in the management discipline and focuses on trust between people, groups of people, or organizations (Mayer et al., 1995). The second one focuses on trust relationships between people and IT artifacts (Lee and See, 2004, Wang and Benbasat, 2005). Due to the fact that HCI studies purport that people enter relationships with IT artifacts and respond to them in a way comparable to responding to other people (Reeves and Nass, 1996), we do not differentiate between interpersonal and system trust in this phase. Unsuitable trust antecedents will be detected in the next steps. There are only few antecedents we could finally use to derive software requirement patterns with the research method. These are written in italics.

Ability	Balanced Asset specificity (tangible and intangible)	Company tenure of a purchasing manager
Accessibility	*Benevolence*	Competence
Attitudinal predisposition towards peers	Business sense	Concern
Availability	Calculative-based beliefs	Confidence in legal system
Availability of competent human resources	Commitment-based HR practices	Confidentiality
	Communication	Congeniality

Table 2: Antecedents of trust (Söllner and Leimeister, 2010, Lee and See, 2004, Muir, 1994)

Consideration of team members' input	Inspirational leadership	Persistence
Consistency	Integrity	*Personalization*
Context-specific reliability	Intentions	Positive feedback profile
Control	Interaction between partner and stranger	*Predictability*
Dependability Willingness to reduce uncertainty	Interaction Frequency	Prior exchange history
Discreetness	Interdependence	Recruitment of own local managers
Distribution fairness	Interpersonal competence	Reliability
Ease of use	Judgment	Shared values
Executive communication	Leap of faith	Shared vision
Executive knowledge	Loyalty	Similarities in demographic attributes
Expectation of continuity	Methods for personal rapport	Sincerity
Expertise	Motivation to lie	Situational normality
Faith	*Motives*	Social interaction ties
Familiarity	Openness	Social presence
Fiduciary responsibility	Organization support	Structural assurance
Functional/specific competence	Organizational tenure	Tactfulness
Generalized value congruence	OSS beliefs	Task-oriented communication
Guanxi	OSS norms	Timeliness
Harmonious conflict resolution	OSS values	Transfer of own business practices
Hostages	Own information sharing	Trial and error experience
Image appeal	Partner similarity	*Understanding*
Information Accuracy	Peer affiliative citizenship behavior	*Visible organizational symbol*
Initial trust condition	Performance	Willingness to reduce uncertainty

Table 2(cont.): Antecedents of trust (Söllner and Leimeister, 2010, Lee and See, 2004, Muir, 1994)

The antecedents express what is perceived by the user. Therefore, there is a strong need for influencing the user perception. We checked the antecedents if the user's perception can be influenced by the system design and specified requirements pattern.

We developed twenty software requirement patterns. We have selected natural language to formulate requirement patterns. Non-technical experts prefer natural language requirements for reading, analysis and discussion. This is in line with recent approaches using software requirement patterns for writing software requirements specifications (Withall, 2008, Renault et al., 2009a).

To illustrate the requirement pattern, we use the following attributes that are components of the recommended structure of a requirement pattern in Franch et al. (2010):

- Goal: The goal has the role of the problem part of a pattern. It has an important role since it will help to decide whether the pattern is applicable to the software (Renault et al., 2009b). This is determined by the planned functionality of the software.
- (Fixed Part) Template: The fixed part template is the core of the solution, stating that the software has to achieve the goal of the requirement pattern, but not indicate how this goal can be achieved. Since the fixed part of a form is abstract, it is possible to provide extra-information or constraints in the extension part about how to achieve the goal of the requirement pattern (Renault et al., 2009b).
- Sources: The sources usually comprise the source documents. For our purposes, we provide the antecedent from which the requirements were derived, and cite the source which mentioned and explained the antecedent.

From trust engineering we had three requirement specifications for recommender systems (restaurants (Söllner et al., 2011a), events, care activities) containing trust requirements. One example requirement from the restaurant recommender is that the user should be able to explicitly rely on ratings of friends before a recommendation is generated. The requirement addresses the antecedent information accuracy (Söllner et al., 2011a). The goal of this requirement is that the user can choose the data which is used for the recommendation. The general requirement template we formulated is: The system shall offer possibilities to the user to select data sources.

For antecedents we had no formulated requirement pattern we used the definition of the antecedents that were provided in the source documents. For example, personalization is used as a trust antecedent by Komiak and Benbasat (2006) for recommendation agents (RA). They define perceived personalization as "a customer's perception of an RA's personalization (i.e., the extent to which the RA understands and represents his or her personal needs)" (Komiak and Benbasat, 2006). Further, they explain that a "RA represents a customer's personal needs as a set of preferred product attributes and/or weights; it then filters the product information, calculates the ranking of the recommended products, and presents its recommendations, ranking, and explanations to the customer. In this case, perceived personalization means that the product attribute preferences used by the RA for its recommendation generation will effectively articulate the customer's personal needs and that the RA's product filtering strategy and ranking calculations are consistent with the customer's personal shopping strategy" (Komiak and Benbasat, 2006). They showed that perceived personalization directly increases trust. This means, that users trust increases if they have setting options to adapt the system to their needs consistently. Therefore, the goal during the development of recommender systems should be: The users have the feeling that they can adapt the recommender systems to their personal needs. To offer the users the feeling that they can adapt the systems to their personal needs the system shall provide setting options. At this level it cannot be generalized which settings are useful, but the requirement suggest that more setting options have a positive influence on trust. Of course, other antecedents, e.g. ease of use, should not be influenced negatively.

The following are examples of software requirement patterns that address antecedents of trust, and thus can enhance user trust in recommender systems (Table 3).

1	**Setting options**	
	Goal	The users have the feeling that they can adapt the recommendations to their personal needs.
	Template	The recommender system shall provide setting options.
	Source	Personalization (Komiak and Benbasat, 2006)
2	**Select data sources**	
	Goal	The users can choose the data which is used for the recommendation.
	Template	The system shall offer possibilities to the user to select data sources.
	Source	Information Accuracy (Fox, 1996)
3	**Up to date**	
	Goal	The users know that the recommendation is up to date.
	Template	The system shall offer the date of used data to the user.
	Source	Information Accuracy (Fox, 1996)
4	**Source of Information**	
	Goal	The users know where the data comes from.
	Template	The system shall offer the source of used data to the user.
	Source	Information Accuracy (Fox, 1996)
5	**Used data**	
	Goal	The users comprehend which data the system uses to create recommendations.
	Template	The system shall present details to the user how the recommendation of the system was created.
	Source	Understanding (Zuboff, 1988)
6	**Data usage**	
	Goal	The users comprehend how the system uses the data to create recommendation.
	Template	The system shall present details which data determine the recommendation of the system.
	Source	Understanding (Zuboff, 1988)

Table 3: Software requirement pattern

7	**Reason for personal data**	
	Goal	The users know why they need to provide their personal data.
	Template	The system shall explain why personal data should be given by the user.
	Source	Understanding (Zuboff, 1988)
8	**Personal data usage**	
	Goal	The users know what happened with their personal data.
	Template	The system shall list the purpose for which the personal data of the user are used.
	Source	Understanding (Zuboff, 1988)
9	**Service selection**	
	Goal	The users can use known services.
	Template	The system shall offer the selection of different services for the same task (e.g., payment).
	Source	Control (Shankar et al., 2002)
10	**Undo input**	
	Goal	The users can undo their inputs of the application.
	Template	The system shall offer functions to the users to delete personal input.
	Source	Control (Shankar et al., 2002)
11	**Undo action**	
	Goal	The users can undo the actions of the application.
	Template	The system shall offer functions to the users to evoke system action.
	Source	Control (Shankar et al., 2002)
12	**Personal data usage II**	
	Goal	The users comprehend what happened with their personal data.
	Template	The system shall list for what the personal data of the user was used.
	Source	Control (Shankar et al., 2002)
13	**Feedback signal**	
	Goal	The users know that something happened.
	Template	The system shall confirm user interaction.
	Source	Control (Shankar et al., 2002)
14	**Self-explanatory button icon**	
	Goal	The users anticipate the future behavior of the IT artifact.
	Template	The icon of buttons shall describe the function it will initiate.
	Source	Predictability (Jennings, 1967)
15	**Self-explanatory button label**	
	Goal	The users anticipate the future behavior of the IT artifact.
	Template	The label of buttons shall describe the function it will initiate.
	Source	Predictability (Jennings, 1967)
16	**Security options**	
	Goal	The users perceive the system producer as being benevolent.
	Template	The system should enable all security options by default.
	Source	Benevolence (Mayer and Gavin, 2005)
17	**Personal data deletion**	
	Goal	The users perceive the system producer as being benevolent.
	Template	The system shall delete personal data that are not used anymore.
	Source	Benevolence (Mayer and Gavin, 2005)
18	**Know the producer**	
	Goal	The users know the positive orientation of the producer towards the user.
	Template	The users shall have the opportunity to get to know the producer (address fear of user).
	Source	Benevolence (Mayer and Gavin, 2005)

Table 3 (cont.): Software requirement pattern

19	**Motives of developers**	
	Goal	The users know why the designers developed the IT artifact (which problem should be solved).
	Template	The system shall describe to the users why it was created.
	Source	Motives (Gabarro, 1978)
20	**Organizational logo**	
	Goal	The users know the brand of the recommender system.
	Template	The system shall provide the organizational symbol.
	Source	Visible organization symbol (Rafaeli et al., 2008)

Table 3 (cont.): Software requirement pattern

When a pattern is to be used, the requirements analyst first has to examine whether this pattern is at all relevant for the design of the system (Renault et al., 2009a). If, for example, a recommender system does not gather, process or utilize personal data, a pattern which only purposes the handling of such data need not be adopted.

To create requirements from the pattern requirements, analysts need to adapt them to the specific software system. We explain the use of the pattern with the help of a restaurant recommender system. We demonstrate the use the software requirement pattern 3 (up to date). The goal of the pattern is that the user knows that the recommendation is up to date. The template says that the system shall offer the date of used or presented information to the user. The information used for the recommendations are, e.g., ratings of other users. Therefore, one requirement for the concrete recommender system could be: The system shall offer the dates of the used user ratings. With this information the user can be sure the recommendation is not outdated. If trust theory is right, this should have a positive influence on user trust.

After identifying all relevant requirement patterns and formulating the requirements for the current application, the requirements analyst should add the trust requirements to the requirements document (Renault et al., 2009a).

5 Discussion

Results from trust engineering show that trust can be enhanced systematically during the system development process (Söllner et al., 2011a). With the help of the presented software requirement pattern we want to give requirements analysts who want to specify recommender systems an easy-to-use approach for considering user trust. According to IS theory on technology acceptance, increased user trust enhances the chances that a specific system will be adopted by its intended users (Gefen et al., 2003). Thus using the presented patterns will help requirements analysts to specify requirements for a recommender system enhancing the chance of the system of being adopted by its intended users. To identify patterns, we examined different trust antecedents and searched for suitable requirements to address these antecedents. Thus, we found technical requirements which were important in different systems. From these technical requirements we formulated the software requirement patterns.

The patterns were developed in the context of recommender systems. Therefore requirements specifications from such systems were used. That is the context we expect the software requirements patterns work best. It should be possible to adapt the pattern to other software systems that provide a graphical user interface. The pattern can help requirements analysts to address trust on a basic level. Other approaches like trust engineering can help them to achieve more detailed and maybe more suitable trust requirements.

Due to the fact that HCI studies purport that people enter relationships with IT artifacts and respond to them in a way comparable to responding to other people (Reeves and Nass, 1996), we did not differentiate between antecedents from interpersonal or system trust. Our results show that with the research method we were only able to derive software requirement patterns from eight antecedents

(Table 2). This is also a limitation to this study. Future research should question the suitability of trust antecedents from personal trust and extend the antecedents of system trust.

Another limitation is that some sources the antecedents were mentioned in do not provide a definition for the antecedents. Therefore, it was hard to identify the purpose of the antecedents. If no other source of the antecedents was given nor it was defined by another publication, we could not include the antecedents in the further process.

We did not check if antecedents overlap each other because the consideration of overlapping antecedents would also enhance user trust. Further, we did not check if software requirement patterns address more than one antecedent. For trust enhancement this would not be a problem at all.

Trust antecedents like expertise (Moorman et al., 1993) or image appeal (Cyr et al., 2009) are characteristics of the producer that need to be built for a longer period of time. Also a positive feedback profile (Ba and Pavlou, 2002) or prior exchange history (Poppo et al., 2008), e.g., in an online store, cannot be specified in advance, but finally enhance user trust. Therefore, if the producer appears to be trustworthy from user experience with past systems, the user will probably trust the new system more easily.

Due to the characteristics of trust, there are overlaps with other system characteristics, especially usability. Perceived ease-of-use is also seen as an antecedent of trust (Gefen et al., 2003). Therefore, every effort to enhance usability can enhance user trust in the system. This goes in line with the trust design guidelines of Patrick et al. (2005).

If the trust design guidelines by Patrick et al. (2005) and the software requirements pattern are compared, it can be seen that there are guidelines and pattern with a similar advices. It shows that there is a broad common understanding in literature how to enhance trust during system development. With the software requirement patterns we try to make it easier for requirements analysts to use the results from trust research for their own requirements specifications.

For use in practice, it is important that the patterns are reusable. If this is ensured the effort to create patterns is worthwhile. To ensure the reusability, we developed patterns by means of technical requirements derived in different projects for different systems. A further challenge in the development of such patterns is that they implement the results of trust research, but should be used by requirements analysts. This assumes that the patterns are specified in a language that can be understood by engineers. For this reason, our patterns were formulated in technical language. Therefore, it could be ensured that there are no misunderstandings.

6 Conclusion

The enhancement of user trust in recommender systems cannot be reached by supplementing individual software components or modules to a system, as they affect the whole software. Therefore, requirements resulting from the trust theory must be considered in the early phases of requirements engineering in order that the trustworthy system design can be ensured in early stages of development. To speak from one's own experience, early consideration of systematic trust enhancement does not take place in most current development projects.

Software requirement patterns offer a solution for requirements analysts to factor trust requirements directly into the information system design. These patterns are generalizable, consequently leading to reusability. We created the software requirement patterns from existing trust requirements and trust antecedents from literature. We specified the patterns in a technical language to guarantee the applicability. With our patterns, requirements analysts have a lightweight approach to incorporate trust requirements into system specifications. It can improve the productivity of requirements analysts, as they can start from a set of predefined requirement patterns in a technical language. This easy-to-use approach can reduce the effort of compiling a list of software requirements and enhance the quality of

the trust requirements because the requirement patterns are created with the help of trust theory and trust experts.

In trust theory, trust is seen as a multifarious construct and many explanatory models of trust exist. Trust engineering emphasizes that trust can be influenced in a more systematic, and thus a more effective, way by influencing its antecedents. With the software requirement pattern we give explicit advice how this can be done while specifying recommender agents.

To enhance usability of the software requirement pattern we plan to integrate the requirement patterns within a requirement pattern catalog. Further, we want to parameterize some parts to allow more detailed choices by each analyst applying the pattern and make it easier to adapt the pattern for different kinds of recommender systems.

References

Alexander, C. (1979). The timeless way of building, Oxford University Press, USA.

Alexander, C., Ishikawa, S. and Silverstein, M. (1977). A pattern language: towns, buildings, construction, Oxford University Press, USA.

Ba, S. and Pavlou, P. A. (2002). Evidence of the Effect of Trust Building Technology in Electronic Markets: Price Premiums and Buyer Behavior. MIS Quarterly, 26 (3), 243-268.

Benbasat, I., Gefen, D. and Pavlou, P. A. (2008). Special Issue: Trust in Online Environments. Journal of Management Information Systems, 24 (4), 5-11.

Benbasat, I., Gefen, D. and Pavlou, P. A. (2010). Introduction to the Special Issue on Novel Perspectives on Trust in Information Systems. MIS Quarterly, 34 (2), 367-371.

Berkovich, M., Leimeister, J. and Krcmar, H. (2011). Requirements Engineering for Product Service Systems. Business & Information Systems Engineering, 3 (6), 369-380.

Cyr, D., Head, M., Larios, H. and Bing, P. (2009). Exploring Human Images in Website Design: A Multi-Method Approach. MIS Quarterly, 33 (3), 539-566.

Ebert, T. a. E. (2009). Facets of Trust in Relationships – A Literature Synthesis of Highly Ranked Trust Articles. Journal of Business Market Management, 3 (1), 65-84.

Fox, J. E. (1996). The effects of information accuracy on user trust and compliance. In Proceedings of the conference companion on Human factors in computing systems: common ground, ACM, Vancouver, British Columbia, Canada.

Franch, X., Palomares, C., Quer, C., Renault, S. and De Lazzer, F. (2010). A Metamodel for Software Requirement Patterns. In Proceedings of the 16th International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 85-90, Essen, Germany.

Gabarro, J. J. (1978). The development of trust, influence, and expectations. In Athos, A. G. & Gabarro, J. J. (Eds.), Interpersonal Behavior: Communication And Understanding In Relationships, pp. 290-303, Prentice-Hal, Englewood Cliffs, NJ.

Gefen, D., Karahanna, E. and Straub, D. W. (2003). Trust and TAM in Online Shopping: An Integrated Model. MIS Quarterly, 27 (1), 51-90.

Gregor, S. (2006). The Nature of Theory in Information Systems. MIS Quarterly, 30 (3), 611-642.

Henninger, S. and Corrêa, V. (2007). Software pattern communities: Current practices and challenges. In Proceedings of the 14th Conference on Pattern Languages of Programs, ACM, New York.

Hoffmann, A., Schulz, T., Hoffmann, H., Jandt, S., Roßnagel, A. and Leimeister, J. M. (2012). Towards the Use of Software Requirement Patterns for Legal Requirements. In Proocedings of the 2nd International Requirements Engineering Efficiency Workshop (REEW 2012) at REFSQ 2012 (Seyff, N. and Madhavji, N. H. Eds.), Essen, Germany.

Jennings, E. E. (1967). The mobile manager: A study of the new generation of top executives, Bureau of Industrial Relations, University of Michigan.

Komiak, S. Y. X. and Benbasat, I. (2006). The Effects of Personalization and Familiarity on Trust and Adoption of Recommendation Agents. MIS Quarterly, 30 (4), 941-960.

Lee, J. D. and See, K. A. (2004). Trust in Automation: Designing for Appropriate Reliance. Human Factors, 46 (1), 50-80.
Leimeister, J. M., Ebner, W. and Krcmar, H. (2005). Design, Implementation, and Evaluation of Trust-Supporting Components in Virtual Communities for Patients. Journal of Management Information Systems, 21 (4), 101-135.
Luhmann, N. (1979). Trust and power, Wiley, Chichester, UK.
Mayer, R. C., Davis, J. H. and Schoorman, F. D. (1995). An Integrative Model of Organizational Trust. Academy of Management Review, 20 (3), 709-734.
Mayer, R. C. and Gavin, M. B. (2005). Trust in management and performance: who minds the shop while the employees watch the boss? Academy of Management Journal, 48 (5), 874-888.
Moorman, C., Deshpande, R. and Zaltman, G. (1993). Factors affecting trust in market research relationships. The Journal of Marketing, 57(1), 81-101.
Muir, B. M. (1994). Trust in automation: Part I. Theoretical issues in the study of trust and human intervention in automated systems. Ergonomics, 37 (11), 1905 - 1922.
Patrick, A., Briggs, P. and Marsh, S. 2005. Designing systems that people will trust. In: Cranor, L. F. and Garfinkel, S. (Eds.), Security and Usability: Designing Secure Systems That People Can Use. Sebastopol, CA: O'Reilly Media.
Poppo, L., Zhou, K. Z. and Sungmin, R. (2008). Alternative Origins to Interorganizational Trust: An Interdependence Perspective on the Shadow of the Past and the Shadow of the Future. Organization Science, 19 (1), 39-55.
Rafaeli, A., Sagy, Y. and Derfler-Rozin, R. (2008). Logos and Initial Compliance: A Strong Case of Mindless Trust. Organization Science, 19 (6), 845-859.
Reeves, B. and Nass, C. (1996). The media equation: how people treat computers, television, and the new media like real people and places, Cambridge University Press, Stanford, CA.
Renault, S., Mendez-Bonilla, O., Franch, X. and Quer, C. (2009a). PABRE: Pattern-based Requirements Elicitation. In Proocedings of the Third International Conference on Research Challenges in Information Science, p. 81-92, Fez, Morocco.
Renault, S., Mendez-Bonilla, O., Franch, X. and Quer, C. (2009b). A Pattern-based Method for building Requirements Documents in Call-for-tender Processes. International Journal of Computer Science and Applications, 6 (5), 175 - 202.
Robertson, S. and Robertson, J. (2006). Mastering the requirements process, Addison-Wesley, Upper Saddle River, NJ, USA.
Shankar, V., Urban, G. L. and Sultan, F. (2002). Online trust: a stakeholder perspective, concepts, implications, and future directions. The Journal of Strategic Information Systems, 11 (3-4), 325-344.
Söllner, M., Hoffmann, A., Altmann, M., Hoffmann, H. and Leimeister, J. M. 2011a. Vertrauen als Designaspekt – Systematische Ableitung vertrauensunterstützender Komponenten am Beispiel einer mobilen Anwendung. In Proceedings of the VHB Jahrestagung. Kaiserslautern, Germany.
Söllner, M., Hoffmann, A., Hoffmann, H. and Leimeister, J. M. 2011b. Towards a Theory of Explanation and Prediction for the Formation of Trust in IT Artifacts. In Proceedings of SIGHCI 2011, Paper 6, Shanghai, China.
Söllner, M., Hoffmann, A., Hoffmann, H. and Leimeister, J. M. (2012). Vertrauensunterstützung für sozio-technische ubiquitäre Systeme. Zeitschrift für Betriebswirtschaft (to appear).
Söllner, M. and Leimeister, J. M. 2010. 15 years of measurement model misspecification in trust research? A theory based approach to solve this problem. In Proceedings of the European Academy of Management Annual Conference 2010, Rome, Italy.
Sommerville, I. (2007). Software Engineering, Addison-Wesley, Harlow, England.
Wang, W. and Benbasat, I. (2005). Trust in and Adoption of Online Recommendation Agents. Journal of the Association for Information Systems, 6 (3), 72-101.
Withall, S. (2008). Software Requirements Patterns, Microsoft Press, Redmont, Washington.
Zuboff, S. (1988). In the age of the smart machine: The future of work and power, Basic Books, New York.