Please quote as: Hoffmann, H. & Söllner, M. (2014): Incorporating Behavioral Trust Theory Into System Development for Ubiquitous Applications. In: Personal and Ubiquitous Computing, Ausgabe/Number: 1, Vol. 18, Erscheinungsjahr/Year: 2014. Seiten/Pages: 117-128.

Pers Ubiquit Comput (2014) 18:117–128
DOI 10.1007/s00779-012-0631-1

ORIGINAL ARTICLE

Incorporating behavioral trust theory into system development for ubiquitous applications

Holger Hoffmann · Matthias Söllner

Received: 22 February 2012 / Accepted: 17 October 2012 / Published online: 22 November 2012
© Springer-Verlag London 2012

Abstract Trust has been shown to be a key factor for technology adoption by users, that is, users prefer to use applications they trust. While existing literature on trust originating in computer science mostly revolves around aspects of information security, authentication, etc., research on trust in automation—originating from behavioral sciences—almost exclusively focuses on the socio-technical context in which applications are embedded. The behavioral theory of trust in automation aims at explaining the formation of trust, helping to identify countermeasures for users' uncertainties that lead to lessened trust in an application. We hence propose an approach to augment the system development process of ubiquitous systems with insights into behavioral trust theory. Our approach enables developers to derive design elements that help foster trust in their application by performing four key activities: identifying *users' uncertainties*, linking them to trust *antecedents from theory*, deducting *functional requirements* and finally *designing trust-supporting design elements* (TSDEs). Evaluating user feedback on two recommender system prototypes, gathered in a study with over 160 participants, we show that by following our process, we were able to derive four TSDEs that helped to significantly increase the users' trust in the system.

Keywords Trust support · Uncertainty · Antecedent · Design elements · Sociotechnical system · Evaluation

H. Hoffmann (✉) · M. Söllner
Kassel University, Kassel, Germany
e-mail: hhoffman@uni-kassel.de
URL: http://www.wi-kassel.de

M. Söllner
e-mail: soellner@uni-kassel.de
URL: http://www.wi-kassel.de

1 Introduction

Current research conducted in the disciplines of computer science and in the behavioral sciences has recognized trust as a factor of major importance in system design. The understanding of the trust concept itself, the formation of trust and insights into how to foster trust vary between and even within the disciplines though. In computer science, most of the research on trust focuses on aspects like information security, authentication and access control—specifically on how to model trust and define cooperative strategies in distributed systems based on this understanding [1]—and only little research effort is spent on developing an understanding of trust in human–computer interaction [2, 3]. In the behavioral sciences, the focus in trust research is on understanding trust in social situations, with extant works on *trust in automation* [4, 5] aiming to explain the trust relationship of a user with technology. One of the major factors in behavioral science is to work with the users' perception of a system in order to understand why they trust or distrust a system and what can be done to foster their trust.

Developing novel ubiquitous applications presents a major challenge when using traditional system development processes, as those systems are not isolated, but used in a social and organizational context and are thus subject to user needs and perceptions in this context [6]. Research on technology acceptance suggests that in order to create successful applications, these needs and perceptions have to be addressed, with trust as a key determinant of technology adoption and usage [4, 7]. This is why improving system security alone is not sufficient to raise user acceptance. In order to affect the users' perception of the system's trustworthiness, trust-supporting measures also have to be brought to the user's attention—for example, by

adding a graphical element like a padlock icon into the address bar of the web browser.

To improve trust in—and consequently foster usage of—ubiquitous applications, we present an approach to augment the traditional software development process with insights into behavioral sciences in order to create trusted ubiquitous applications. The approach is designed to specifically cope with user perception in the form of requirements and design elements and is hence related to human computer interaction (HCI) research. Our approach is designed to be usable for system developers alongside software development process models that follow a rigid structure (e.g., V-Model XT) as well as agile development processes (e.g., SCRUM). Consequently, the resulting functional requirements and tangible designs for trust-supporting design elements can be used in conjunction with approaches covering traditional trust aspects in computer science, like secure communications, authentication, etc.; developers are also able to conduct the steps of our approach ex-post, that is, our approach can be used to improve existing ubiquitous systems.

In the following sections, we briefly outline the different perspective of trust in behavioral research concerning technology acceptance compared to computer science research. Building on the research on trust in automation, we then present a process for employing the underlying trust theory as a way to define requirements and derive design elements during system development. We applied this approach to DinnerNow, a ubiquitous recommender system for finding the best restaurant for a spontaneous visit. In an evaluation with over 160 participants, we were able to show that the design elements we derived using our process significantly increased the subjects' trust in the application.

2 The different perspective on trust from behavioral sciences

In current literature on trust in computer science, the topic of trust appears in many different and rather diverse areas. Artz and Gil [1] made an attempt to classify the research on trust in computer science, showing that the understanding of trust and trust research in computer science revolves around topics like information security, access control and reliability in networks and distributed systems as well as trust in game theory or agent systems and finally trust in the decision-making process. The underlying concepts differ concerning the definitions, methods and tools. When considering the state of trust research in computer science, two things are especially noteworthy: Firstly, there are currently very few publications, for example, [8], covering software engineering methods to improve trust, that is, there are few rigorous methods to follow for improving trust. Secondly, while research on dependable and secure computing [9] already includes system "attributes" for trusted computing, HCI research—as a bridge between computer science and the behavioral sciences—hints at the importance of human perception and special design elements to foster trust. Sillence et al. [10] found that the look and feel of a website influences the users' judgment whether or not the site was trustworthy, and Stephens [2] proposes concrete design elements to foster trust on websites.

The understanding of trust in the behavioral sciences faces similar problems to those found in computer science, as there also is no consensus on the definition of trust. Nevertheless, Rousseau et al. [11] note that many definitions have a common core based on positive expectations and vulnerability. While early research on trust in behavioral sciences often focused on trust relationships between people or organizations—especially in the management community (e.g., [12, 13])—researchers within the IS and HCI community began studying trust relationships between people and technology (e.g., [4, 14]). Regarding the development of ubiquitous systems, the work on trust in automation by Lee and See [4] is the most promising theory, as they define automation as "technology that actively selects data, transforms information, makes decisions or controls processes." This definition is easily applicable to ubiquitous systems, since these systems in most cases will conduct several of the listed actions to fulfill their purpose and support their user. Consequently, we adapt the understanding of trust in automation by Lee and See [4] and define trust as the belief "that an agent will help achieve an individual's goal in a situation characterized by uncertainty and vulnerability." Following this definition, the ubiquitous system is the agent and the ubiquitous system's user is the individual. The bulk of behavioral literature interprets this trust as a multi-dimensional latent construct [15], which is formed by multiple antecedents representing factors for fostering trust.

Research on trust in automation shares this view of trust and identifies multiple antecedents leading to user's trust in an automated system, classified into one of three "summary dimensions": *performance*, *process* and *purpose* [4]. Antecedents in the *performance* dimension reflect the user's assessment of the capability of the system in helping him to achieve his goals, those in the *process* dimension reflect the user's perception regarding the system's functionality and the degree to which the system's algorithms are chosen and implemented appropriately, and finally, the antecedents in the *purpose* dimension are indicative of the user's assumptions of the system designer's intention when developing the system [4, 16].

An exemplary model, incorporating a selection of antecedents of trust in automation, is shown in Fig. 1. Four antecedents are classified in the *performance* dimension: *Competence* reflects the user's rating of the overall suitability of the system to solve his problem. *Information accuracy* stands for the user's assessment of how precise the information given by the system is. *Reliability over time* gauges the user's assessment how reliable the system will be in the future. *Responsibility* represents the opinion whether the IT artifact has all the functionality needed to achieve the user's goal.

Another four antecedents are classified as belonging to the *process* dimension: *Dependability* is the user's perception whether the system's reactions are consistent and reliable. *Understandability* indicates the user's ability to grasp the functionality of the system. *Control* stands for the user's impression of being in control of the system. *Predictability* represents the user's assumptions whether the system's behavior can be anticipated.

Finally, three antecedents are classified in the *purpose* dimension: *Motives* measure the issue of whether the user thinks that the developers truthfully communicated the system's intended use. *Benevolence* of designers is the indicator of the user's assessment if the developers have the users' interests in mind. *Faith* stands for the user's belief that he can rely on the system in the future.

Next to the very prominent theoretical model by Lee and See [4], a number of publications offer additional insights into trust formation in technical systems. Approaching the trust relationship between users and a technical system from various points of view, those theories propose various additional antecedents that play a role in fostering trust. Collections of possible antecedents, including a definition of the antecedents and/or references to the original theories, are presented in [4, 16, 17]. The following table shows an excerpt from those collections.

The antecedents in trust in automation theory are comparable to what is called a system attribute in the taxonomy by Avizienis et al. [9]. Even though they revolve around similar concepts, three aspects distinguish them. First, the point of view is reversed: Avizienis et al. consider system attributes to be system characteristics, which are fixed in software and therefore can be determined precisely, while trust in automation regards the antecedents as the users' perception—which depends on the individual user and can be biased or flawed. Second, even though both works share common names, the meaning is often different, for example, defined Avizienis et al. [9] dependability as an integrating concept that encompasses system attributes like availability and reliability, while Lee and See [4] consider it an antecedent covering the users' perception whether the system's reactions are consistent and reliable. Third, the taxonomy of dependable computing is considering six system attributes, covering the interaction between computer systems, while Lee and See [4] leave the total number of antecedents open. Using the theoretical framing by Lee and See as our foundation and augmenting it with antecedents from related theories, we argue that it is possible to identify software requirements and designs that support the antecedents and hence foster the formation of users' trust. Unlike approaches that attempt to find requirements and designs for the latent construct of "trust," this process helps to systematically establish trust based on theoretical insights into elements that have been shown to affect it.

3 Creating trust-supporting design elements for ubiquitous applications

While the potential of incorporating behavioral trust theory into the system design process has been acknowledged before, only technical flaws and weaknesses in a system's design, implementation or operation are covered by approaches like vulnerability analyses for fostering trust concepts found in computer science [20]. Until now existing research on how to transfer behavioral insights

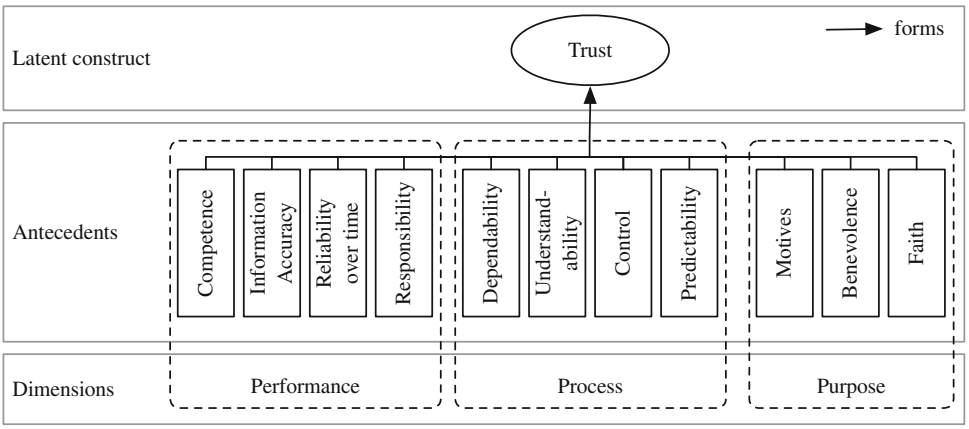

Fig. 1 Model of the formation of trust in automated systems derived from [5]

into software requirements and ultimately design elements fostering trust from the behavioristic understanding is limited to a few publications that rather describe the overall idea than present a concrete process [21–23].

We propose a process consisting of four steps that enables developers to systematically derive trust-supporting design elements (TSDEs) from theory on trust in automation (Fig. 2) for their application to render it more trustworthy for the application's users. Our process covers aspects of trust theory as well as requirements engineering and system design, by taking users' uncertainties, pinpointed in the application context with the users' help, as a basis to identify antecedents from behavioral trust theory as remedial measures that help overcome the uncertainties and hence foster users' trust. These antecedents are then translated into functional requirements, which serve as input for arriving at concrete TSDEs in the final step.

While the process is intended to deliver tangible TSDEs that can be directly integrated into a system, it is also possible to only perform steps 1–3 and uses the resulting functional requirements as input for a traditional system development process. Decoupling the procedure from the conventional software engineering process has two advantages. First, it does not interfere with different process styles and can be used in projects following a rigid structure like the V-Model XT as well as agile projects, for example, using SCRUM. Second, this standalone process can be used ex-post, that is, it can be applied to an existing system that needs improvement.

3.1 Identifying and prioritizing uncertainties

Following the definition of trust by Lee and See [4], it becomes evident that trust is only relevant in usage situations characterized by uncertainty. Hence, the first step toward developing a more trustworthy system is to develop an understanding of the future system's users' uncertainties. Trust in ubiquitous systems is affected by characteristics such as multiple data sources, data processing by several service providers as well as context awareness and adaption to contexts. Literature on requirements engineering as well as human–computer interaction mentions a plethora of different techniques that can be used—sometimes with a small shift in focus—to systematically determine uncertainties in a systematic manner [6]: A technique widely used for identifying requirements are *interviews* with future users, which can be easily adapted to also identify uncertainties the users might have. Real-world *scenario descriptions* can be used to gain insights into interviews, even if only abstract system descriptions are available [24]. *Viewpoint-oriented approaches* take into account different perspectives on the system usage and allow discovering conflicts that can result in uncertainties [25]. Even *ethnographical methods* can be used, where the use of a system is observed from the outside in order to understand uncertainties.

In order to be able to decide which of those uncertainties to address first, they are prioritized based on their threat to a successful adoption of the system by the user. To systematically determine an appropriate ranking of uncertainties, established requirements prioritization methods [26] as well as methods from requirements negotiation and collaboration engineering can be used [27]. Resulting in an ordered list of uncertainties to address, a selection can be made to reduce additional development effort, and thus costs.

3.2 Identifying remedial antecedents from theory

In the second step, the previously found uncertainties are being matched to antecedents that are identified as remedial for the uncertainties by theory. The aim in doing so is to determine which antecedents that foster trust are related to the uncertainties and can then be used to systematically derive functional requirements in the next step. Depending on the type of trust relationship affected by the uncertainty, different basic theories for finding antecedents may be used. In the case of trust relationships between humans—for example— regarding assumption about the motives and benevolence of the system developer—an organizational trust model is applicable [12]. For the more prevalent relationship between a user and the system, collections of possible antecedents are presented in [4, 16]. An excerpt from these collections is given in Table 1. The selection of an antecedent to counter the uncertainty is then based on how well the antecedent's definition in theory matches the uncertainty description.

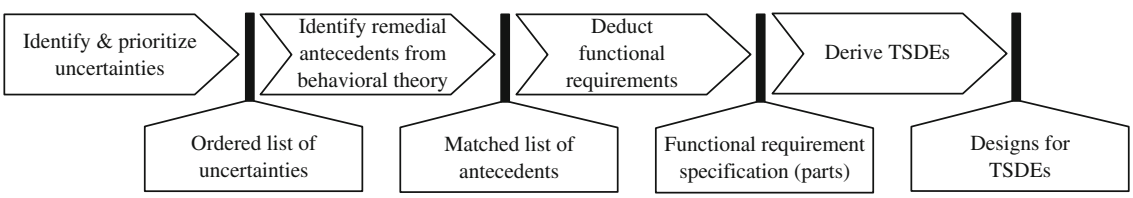

Fig. 2 Process for systematically deriving TSDEs

Table 1 Exemplary trust antecedents from [4, 18, 19], antecedents used for case study in italics

Ability	Discreetness	Persistence
Accessibility	Expertise	Personalization
Availability	Fairness	Predictability
Benevolence	Faith	Reliability
Communication	Familiarity	Shared values
Competence	Image appeal	Shared vision
Concern	Integrity	Sincerity
Confidentiality	*Information accuracy*	Social presence
Consistency	Judgment	Support
Continuity	Motivation	Timeliness
Control	Motives	*Understandability*
Dependability	Openness	

As there may be multiple antecedents applicable for alleviating a single uncertainty, it can also become necessary to decide which of the antecedents to select for the coming steps in the development process. Due to the lack of other research on how to perform this selection, we advise developers to choose as few antecedents as possible—since each antecedent will result in at least one functional requirement to be implemented, thus raising the development effort. Thus, multiple antecedents are only used for uncertainties identified as critical. We also suggest choosing the antecedent that is described as having the highest influence on trust in theory.

3.3 Deducting functional requirements

During the third step, concrete requirements that can be used in the system's specification are deducted from the list of antecedents, identified in the previous stage, and the uncertainties the antecedents aim at ameliorating. Requirements are classified as either *functional*, that is, defining system functions, or *nonfunctional*, that is, defining system properties like response time or availability [6]. The antecedents identified before can be interpreted as a subtype of nonfunctional requirements, so-called underspecified functional requirements [28]. These requirements allow for a wide range of interpretations of what exactly they mean concerning the future system's characteristics, hence they need to be refined into functional requirements [28]. To remove any ambiguity during system design, a number of methods found in the requirements engineering literature can be applied [29–31].

When deducting functional requirements, it is also necessary to consider the situational context, for example, the usage situation, in which the antecedent is supposed to counteract the initial uncertainty, since the functionality defined here is used in the same context. A compilation of a reusable set of antecedent-to-requirement translations—analogous to software requirement patterns [32] or patterns of interaction [33]—helps to both reduces the effort needed in this step of the process and at the same time improves the quality of the functional requirements derived. It is important to note that antecedents can also be influenced by trust-supporting measures that cannot be defined as functional requirements—and hence not result in software design elements. An excellent example for this is expertise, the user's opinion whether the developers have the means to create a high-quality application, where prior own experiences, media coverage, gossip, etc., have a great impact [4]. While established developers like Apple, IBM and Microsoft already benefit from their position, newcomers like smaller companies and individuals can try to improve the perception of benevolence by making their actions more public, for example, by offering open support forums, or by using marketing tools like websites or application reviews by trusted third parties.

3.4 Deriving trust-supporting design elements

The fourth and final step to incorporate behavioral trust theory into system design is the design of elements that support trust based on the functional requirements gathered in the previous step. Like with all design activity in the system development process, different designs are created and evaluated before making a decision which design best matches the requirements and is feasible for realization [6]. As this is a creative process, and creative processes can only be supported methodologically to a limited extent, deriving trust-supporting design elements requires prior experience in system design [6]. While this step is arguably the most vague in the overall process—like in every system development process—existing insights and conclusions from computer science and human–computer interaction can be incorporated into the design.

In today's literature on trust in computer science, many different aspects of trust are described and hence are available for consideration when deriving TSDEs for a certain functional requirement. Current research in computer science, or more precisely dependable computing and information security, presents many potential technical components that can be used to realize TSDEs, for example, measures to ensure information security, establish access control and improve reliability in networks and distributed systems [1]. However, those solutions do not alter the user's perception unless they are "advertised," for example, in the system description or by adding a graphical element—like a padlock icon—to the user interface.

The final result of our process is a design of concrete trust-supporting elements for an application that are noticeable for the user. These design elements can then be

implemented and integrated into the application to help increase the users' trust in it.

4 Incorporating trust-supporting design elements—the case of DinnerNow

In order to illustrate the suitability of our proposed process, show its application in practice and evaluate its results, we present the case of a ubiquitous application, the context-aware restaurant recommender system "DinnerNow." In the following section, we describe a naïvely designed version of DinnerNow that does not contain explicitly designed components to improve the user's trust. Subsequently, we describe how our proposed process is applied to derive trust-supporting design elements and create a trustworthier version of DinnerNow. For the sake of clarity, we will presume that all the research found in computer science applicable to recommender systems—most importantly policy-based trust and reputation-based trust models—have already been taken into account, and the system is hence secured against malicious attempts to tamper with its functionality. Hence, we focus only on those aspects of trust related to the end user's perception of DinnerNow as a whole.

We will refer to the initial naïve version as DinnerNow LT (for "low trust") and to the version enhanced with TSDEs as DinnerNow HT (for "high trust") for easier understanding.

4.1 The original DinnerNow application

The application's main goal is to support the decision making when two or more people spontaneously decide to have lunch or dinner together in an unfamiliar environment. DinnerNow allows the user to freely select the filter criteria for the recommender system. In the naïve approach—that is, without factoring in the support of trust—the user may choose whether to use personal preferences like the ethnicity of the cuisine, the restaurant's ambiance and previous personal experiences. He may also want to include ratings from internet-based rating portals like qype.com or google.com. Additionally, DinnerNow automatically takes the users current location into account when generating the recommendation. Since both past interaction—in form of the user's or company's previous experiences—and profile attribute information—like preferred type of cuisine—are taken into account, DinnerNow is an example for a hybrid recommender system combining collaborative filtering (historical interaction) and content-based filtering (profile attributes) [34].

After selecting the input criteria and requesting a recommendation, the user is presented with a screen showing him the best option based on the input for the current location. Included into this screen are some details about the restaurant as well as functions to call the restaurant (e.g., for a reservation), to open a navigation window to the restaurant and the option to see the next best choice found by the recommender system. Should the user be dissatisfied with the recommendations, he can visit the start screen again, change the settings and generate a new set of recommendations. Figure 3 shows the layout for the screen for changing the search options and the result screen giving a recommendation.

4.2 Designing a more trustworthy version of DinnerNow

Using the implementation of DinnerNow LT as a recommender system without explicitly taking trust into account, we follow the process for systematically deriving TSDEs laid out above in order to create a more trustworthy version of DinnerNow. The effect of the TSDEs is then evaluated in a study where subjects perform tasks using either DinnerNow LT or the enhanced DinnerNow HT—without being told which version they are using—and afterward answer a questionnaire about their perception of DinnerNow's trustworthiness.

4.2.1 Uncertainties when using DinnerNow LT

The first step in our process is to identify and prioritize the users' uncertainties when using the application that lessen their trust in a system. The ubiquitous application's intended use is to provide a restaurant recommendation based on multiple persons' preferences, rating from the Internet and their current location. Using this description along with the DinnerNow LT prototype as an input, the uncertainties for the scenario were identified using the Thinking Aloud Method with three potential users [35, 36]. All three had an extensive knowledge of mobile, pervasive, ubiquitous technology and come from a technical background. The process helped to identify the following nine distinct uncertainties.

Two uncertainties arise when selecting the search options for the recommender system. Users are uncertain whether DinnerNow really has access to the information in the users' profiles on social networking sites and if the selection of preferences really has an influence on the algorithmic determination of a recommendation.

Another two uncertainties are directly associated with the system's core feature, the recommendation. Firstly, users are not certain if DinnerNow finds a recommendation based solely on the selected search options or if other factors play a role too, for example, the suspicion that the application provider manipulates the recommendation due

Fig. 3 DinnerNow search options (stretched) and results screen without TSDEs

to kickbacks received from certain restaurant owners. Hence, they are unsure about the recommendation's quality. Another concern is whether the opinion of a broad group of internet users giving ratings for restaurants will reflect the user's taste—this is especially true when using the application in a foreign country immersed in a different culture.

Five more uncertainties are associated with the way the recommendations are presented. The question if the information displayed is up-to-date is one of them, as the user has to rely on the restaurant's still being in business and on the fact that the restaurant's quality has not changed. If a user is not happy with a recommendation, he has the option to get the next best suggestion by selecting "Next," or he can completely start over using different search options. The three test users reported that this was very annoying, since they felt that they could influence the interaction process more directly instead of only choosing "Next" when they were not satisfied with the first two recommendations. If the user has found a recommendation he is happy with, he has the option to open a navigation window and see an estimate for the time needed to get to the restaurant. Uncertainties in this context are whether his position has been determined correctly, whether the calculated itinerary is the best available and whether the estimates for the distance and time needed to get to the restaurant are accurate.

After having identified those nine uncertainties, their relative importance for the users' was determined in a moderated discussion [37], using an outranking-based approach [26]. The users agreed that the uncertainty about the quality of the recommendation ranked highest, followed by the uncertainty of not knowing the list of recommendations and the applicability of the opinions voiced by strangers on Internet portals. These three uncertainties were chosen to be alleviated; the complete ranked listing of uncertainties is given in Table 2.

Table 2 Ranked list of uncertainties for DinnerNow, uncertainties used for case study in italics

Rank	Uncertainty
1	*Does DinnerNow really use my input to give me a recommendation?*
2	*Why are choosing "Next" or restarting with different search options my only possibilities when I am not satisfied with the generated recommendation?*
3	*Do people whose opinion is used by DinnerNow share my taste?*
4	How do my selected options influence the recommendation?
5	Does DinnerNow really have access to my social networking profile?
6	Are the distance/time estimates correct?
	How up-to-date is the restaurant information?
8	Does DinnerNow find the best route to the restaurant?
	How good is the localization of my position?

4.2.2 Identifying relevant antecedents for DinnerNow

In the second step of the process, the relevant antecedents for trust are identified for each of the uncertainties found in

the previous step. As described above, this is done best by choosing an existing antecedent from trust theory. Compilations of antecedents and their origin—like those found in [4, 16, 17]—help speed up this process immensely. Using those compilations, the following antecedents were identified as relevant for the three top uncertainties when using DinnerNow.

The uncertainty whether DinnerNow really takes the user's input to generate a recommendation is an indicator of the user want to be able to comprehend how the recommender systems works internally in order to arrive at the best restaurant choice. A remedy for this problem is to improve the antecedent of *understandability* by informing the user about how the recommendation was found.

Wondering that they are limited to choosing "Next" or restarting with different search options when they are not satisfied with the recommendation indicates that the users want to interact more directly with DinnerNow in this specific situation. An antecedent suitable for countering this uncertainty is *control*, since the reports of the users indicate that they want to alter the default recommendation process. Thus, increasing the control of the users in this specific situation should counter this uncertainty.

The users' uncertainty about the applicability of opinions posted on Internet portals shows that they question the information resources used by the recommender system. The antecedent associated with this is *information accuracy*. To alleviate this uncertainty, the user should be able to see how well the information resources match his needs or even supply own resources (Table 3).

Having identified one antecedent per uncertainty provides the foundation for initiating concrete measures in system development to alleviate those uncertainties, starting with deducting additional functional requirements.

4.2.3 Functional requirements for DinnerNow HT

In the third step of our process, the antecedents identified before are used to deduct functional requirements for DinnerNow HT. For the three antecedents at hand, this process is very straightforward, also given that DinnerNow only covers a small context and has few points of interaction.

Table 3 Top uncertainties for DinnerNow and their antecedents

Rank	Uncertainty	Antecedent
1	DinnerNow's use of input	Understandability
2	Being limited to choosing "Next" or restarting the search process	Control
3	Applicability of opinions from Internet portals	Information accuracy

In the previous section, we assumed that the user needs a better *understanding* of how the recommendation was found in order to trust its quality. As the main functionality of DinnerNow is to create a restaurant recommendation based on contextual factors, especially the user's and his company's preferences, this process needs to be made easier to follow. One way of explaining to the layman user how the system came up with its recommendation is to inform him about the degree to which his individual preferences were considered when finding the recommendation. The functional requirement covering this would be that

R1. To better understand how the recommendation was generated, the user can view details on how his preferences are integrated into the recommendation process when viewing the search results.

To reassure the user that he is in *control* of the system, he needs to be able to influence the recommendation process if he is not satisfied with the recommendation. One possibility to achieve this is to give the user access to the ordered list of recommendations created by DinnerNow. Instead of being forced to follow the list as it is and click through it on an item by item basis, the listing of the recommendations enables the user to quickly check and alter the result. It has to be possible for him to review the complete list of recommendations and optionally also make changes (i.e., re-order the list). The functional requirement covering this is that

R2. To improve control over DinnerNow, the user has the option to browse the list of recommendations—sorted by DinnerNow—and change the sort order once the recommendation is generated.

To resolve the uncertainty whether the information accurately matches the user's need, we found that the user should be able to integrate trusted sources of information into the recommender system and see if he is satisfied with the outcome when viewing the results of his search. While there are multiple choices for "trusted sources," we have chosen to allow the integration of personal friends' recommendations as this nicely reflects the social nature of the recommender system [38]. The "fit" is made explicit by displaying the information from this trusted source along with the system's recommendation. We deducted two functional requirements for DinnerNow HT from this

R3. To improve information accuracy, the user is able to integrate his friends' ratings as an input to generate the recommendation.

R4. To demonstrate the information accuracy, the user's friends' ratings are made explicit when presenting a recommendation.

4.2.4 Deriving trust-supporting design elements

During the final step of our process, design elements are derived from the functional requirements we just deducted. As we focus on the user's perception of trustworthiness, we limit our description to those elements appearing in the graphical user interface and do not go into detail about the "invisible" changes to the recommendation engine. An overview over the DinnerNow HT search and result screens is shown in Fig. 4.

We realized the first requirement by adding another button labeled "Fit" to the result screen (Fig. 4, TSDE1). Clicking this button takes the user to another screen that shows in detail to what degree the recommended restaurant meets the preferences he chose on the search screen. We decided to put this information on a different screen that has to be requested by the user, in order to allow the user to choose whether he would like to see the details of the recommendation fit or not.

To fulfill the second requirement, we added another button labeled "Self Selection" to the result screen (Fig. 4, TSDE2). By clicking this button, the user can review the complete list of restaurants as sorted by DinnerNow and change the order by clicking on the respective column in the list header. We placed this option to choose an entirely different restaurant at the bottom of the result screen, as it is only needed when the user is completely dissatisfied with the results by DinnerNow.

The third requirement is approached by offering the user to include ratings by his friends in addition to general customer ratings from Internet portals when specifying the search options (Fig. 4, TSDE3). For the current prototype, we decided to summarize the latter in one option instead of listing different portals.

Lastly, we exchanged the general "customer rating" information for ratings specific to the user's friends to fulfill the fourth requirement (Fig. 4, TSDE4). The label "Friends' Ratings" makes it obvious to the user that the information is based on what his friends think rather than on what conclusion strangers might have drawn.

4.3 Evaluation of DinnerNow

After successfully applying our process to derive TSDEs from behavioral theories and integrating them into a sociotechnical ubiquitous system, the actual impact of the TSDEs on potential users needs to be evaluated. This is a necessary step to show that the use of TSDEs derived by this process leads to a higher trust in the overall system.

For our study, 166 subjects were divided into two groups. Each group was assigned one of the two versions of DinnerNow—without being told whether they were using the LT or HT version—and asked to complete predefined tasks and document the results achieved. After having experienced DinnerNow, subjects were asked to fill out a questionnaire consisting of 7-point Likert scales capturing

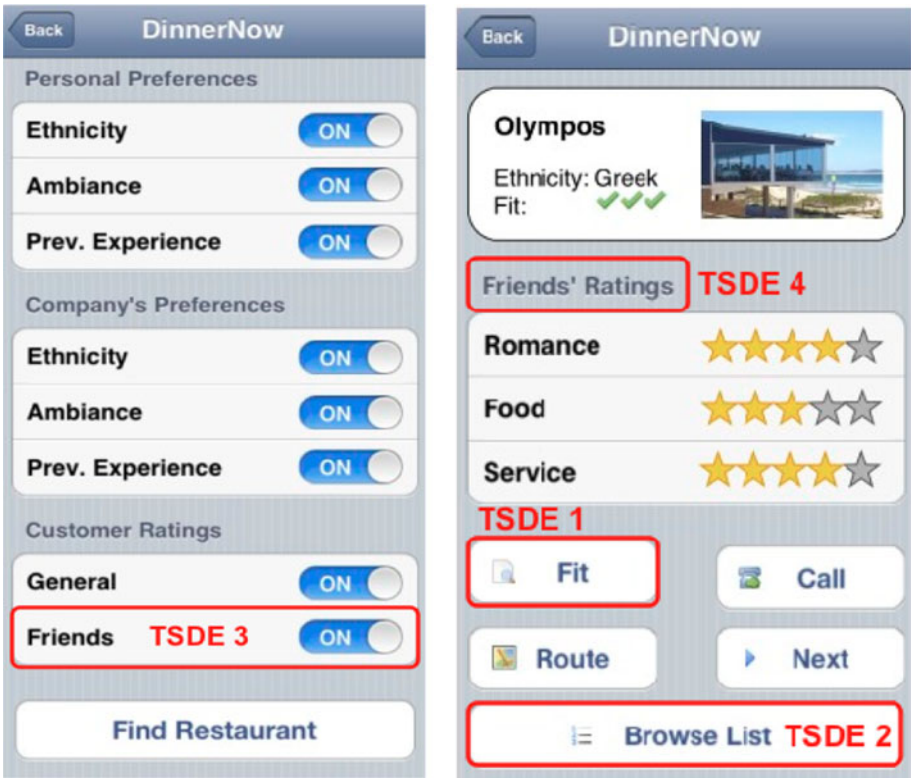

Fig. 4 DinnerNow search options (stretched) and results screen with TSDEs highlighted

Table 4 Changes in user perception and trust for Low and High Trust prototypes

Variable	Mean LT	Mean HT	*t* value
Understandability	4.76	5.24	1.946**
Control	5.79	5.80	$0.056^{n.s.}$
Information accuracy	4.49	5.06	2.640***
Trust	4.81	5.11	1.455*

Level of significance: * $p < 0.1$; ** $p < 0.05$; *** $p < 0.01$; *n.s.* not significant

the items necessary for the evaluation. Of the 166 questionnaires collected, 143 could be used for further analysis and 23 had to be discarded due to statistical inconsistencies in the answers given. DinnerNow without TSDEs (DinnerNow LT) was evaluated by 75 subjects; the other 68 subjects evaluated DinnerNow including the TSDEs (DinnerNow HT).

Table 4 summarizes the change in perception for the antecedents we aimed to influence with TSDEs—*understandability*, *control* and *information accuracy*—as well as the change in the users' trust in the DinnerNow recommender system. Answers on the questionnaire were given using a seven-point scale in the Likert response format (1 = strongly agree, 7 = strongly disagree).

The results of the evaluation show that the average assessment of all the variables tested improves. For all variables, this improvement is significant, except for the *control* antecedent. Values for control are extremely high in both the LT and the HT version of DinnerNow. We consider two possible explanations: Either the closed experimental design we have chosen hinders the correct measurement of the perceived control a user has over the system or the users overestimated their control over the prototype. In line with the latter explanation is the observation how careless many users treat private and personal information in their daily routine with "social" services like Facebook or Twitter [39].

Still, the other two antecedents directly addressed in DinnerNow HT, *understandability* and *information accuracy*, significantly improved due to the integration of the TSDEs. Finally, the users' trust in DinnerNow improves significantly with the inclusion of the TSDEs.

5 Discussion and conclusion

In this article, we argued for the importance of integrating insights into behavioral theories into the system development process in order to improve the end user's trust in the resulting system. To facilitate this integration effort, we propose a process based on the theoretical foundation of trust in automation, which covers the trust relationship between a user and an automated system. The process consists of four key activities, creating a path from behavioral trust theory to concrete system design elements. Thus, the task of improving trust is broken down into smaller subtasks that can be handled more systematically.

Using our process, we were able to derive four TSDEs for improving the trustworthiness of our prototypical implementation of a ubiquitous recommender system. The evaluation of prototypes that either contain or do not contain these TSDEs shows that incorporating them resulted in an increase of both end users' trust in the system.

Hence, our research shows that the behavioral theory of trust in automation can be applied to a system development setting. It can be integrated well into the software development process, bridging the gap between the understanding of trust in computer science, often focused on security aspects or authentication, and the trust concept in the behavioral sciences, mostly concerned with human perceptions, assessments and assumptions concerning computer systems. As we were able to replicate the successful use of our process using one of our other applications—a mobile multimodal learning game—and two applications of other research groups—a mobile and context-aware event organizer and an ubiquitous social conference guide—we conclude that the overall approach is well suited for adoption and further research. However, there are a couple of limitations to our research that need to be addressed.

For our *evaluation*, we chose a laboratory-based experimental setting, as the complexity of the recommender system barred more sophisticated evaluation settings, for example, a real-life prototype for a city. The question arises whether our test subjects really behaved similar to such a real-life evaluation and whether their perceptions are comparable to what they would have experienced when using DinnerNow outside of our closed setup. On the positive side, this type of evaluation helped to directly influence the course of events during the evaluation. This allowed us to both pretest an earlier version of similar TSDEs and directly modify the prototype without our subjects noticing this and hence not tampering with their perceptions [40].

Also, in our *evaluation,* the median level of trust rose by about one third of a point on the Likert scale. However, the significance of this change is on a low level. While this is not optimal, further analysis of the data available using a related theory of technology acceptance found in [16, 41] indicates that addressing more uncertainties (i.e., include more TSDEs) than the three we limited our study would lead to a more significant change in trust.

In the description of our *proposed process to derive TSDEs*, instructions on how exactly to find matching antecedents, to deduct functional requirements from them

and how to design system features to support trust are still vague. This problem is well known in computer science, as both requirements engineering—especially elicitation and analysis—and system design share this problem of a creative process that is hard to fit into the framing of a method [6]. However, unlike existing approaches, where designs are proposed based on implicit assumptions about uncertainties, antecedents to resolve those and functional requirements mapping the antecedents to functions, our approach makes the whole process explicit and the respective decisions both comprehensible and traceable. As a result, ineffective TSDEs are not discarded without the chance of learning from them, and the design process can be traced back and intermediate decisions be revised.

Both from the limitations of our current work and the vast body of knowledge concerning the handling of trust in computer science, a couple of aspects for further research arise. Three major fields for further research are (1) to determine the applicability of our process for the development of a wider variety of ubiquitous applications, (2) tracking the effect of TSDEs over a longer usage period (i.e., track the effects of the TSDEs after the users have grown accustomed to the system) and (3) formalizing the behavioral trust concepts to be able to model and reason about trust computationally. Additionally, a compilation of TSDEs that have proven successful in evaluations could serve as the foundation of a line of trust-related design patterns that render creating trustworthy systems even simpler. In general, the aim of future research should be to open up behavioral insights into a wider audience, for example, by including them into CASE tools for modeling and analyzing trust requirements for novel ubiquitous systems.

Acknowledgments The authors thank Hesse's Ministry of Higher Education, Research, and the Arts for funding their research within the VENUS research cluster at the interdisciplinary Research Center for Information System Design (ITeG) at Kassel University as part of the research funding program "LOEWE". Parts of this research was developed in the scope of the project Value4Cloud, funded by the German Federal Ministry for Economics and Technology (FKZ: 01MD11043A).

References

1. Artz D, Gil Y (2007) A survey of trust in computer science and the semantic web. Web Semant Sci Serv Agents World Wide Web 5(2):58–71
2. Stephens RT (2004) A framework for the identification of electronic commerce design elements that enable trust within the small hotel industry. Paper presented at the 42nd annual Southeast Regional Conference (ACM-SE), Huntsville, 02–03 April 2004
3. Gerd tom Markotten D, Kaiser J (2000) Usable security—challenges and model for e-commerce systems. Wirtschaftsinformatik 6:531–538
4. Lee JD, See KA (2004) Trust in automation: designing for appropriate reliance. Hum Factors 46(1):50–80
5. Söllner M, Hoffmann A, Hoffmann H, Leimeister JM (2012) How to use behavioral research insights on trust for HCI system design. Paper presented at the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2012), Austin, 05–10 May 2012
6. Sommerville I (2007) Software engineering, 8th edn. Addison-Wesley, Harlow
7. Gefen D, Karahanna E, Straub DW (2003) Trust and TAM in online shopping: an integrated model. MIS Q 27(1):51–90
8. Viega J, Kohno T, Potter B (2001) Trust (and mistrust) in secure applications. Commun ACM 44(2):31–36
9. Avizienis A, Laprie J-C, Randell B, Landwehr C (2004) Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Dependable Secure Comput 1(1):11–33
10. Sillence E, Briggs P, Fishwick L, Harris P (2004) Trust and mistrust of online health sites. Paper presented at the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2004), Vienna, 24–29 April 2004
11. Rousseau DM, Sitkin SB (1998) Not so different at all: a cross disciplinary view of trust. Acad Manag Rev 23(3):393–404
12. Mayer RC, Davis JH, Schoorman FD (1995) An integrative model of organizational trust. Acad Manag Rev 20(3):709–734
13. Gambetta D (1990) Can we trust trust? In: Gambetta D (ed) Trust: making and breaking cooperative relations. Basil Blackwell, Oxford, pp 213–237
14. Vance A, Elie-Dit-Cosaque C, Straub DW (2008) Examining trust in information technology artifacts: the effects of system quality and culture. J Manag Inf Syst 24(4):73–100
15. Jarvis CB, Mackenzie SB, Podsakoff PM (2003) A critical review of construct indicators and measurement model misspecification in marketing and consumer research. J Consum Res 30(2):199–218
16. Muir BM (1994) Trust in automation: part I. Theoretical issues in the study of trust and human intervention in automated systems. Ergonomics 37(11):1905–1922
17. Söllner M, Leimeister JM (2010) 15 years of measurement model misspecification in trust research? A theory based approach to solve this problem. In: 10th Academy of Management Annual Meeting, Rome
18. Shankar V, Urban GL, Sultan F (2002) Online trust: a stakeholder perspective, concepts, implications, and future directions. J Strateg Inf Syst 11(3–4):325–344
19. Söllner M, Hoffmann A, Hoffmann H, Leimeister JM (2011) Towards a theory of explanation and prediction for the formation of trust in IT artifacts. Paper presented at the 10. Annual Workshop on HCI Research in MIS, Shanghai, 04 December 2011
20. Shirey R (2000) RFC 4949—internet security glossary, version 2. The Internet Society, Geneva
21. Leimeister JM, Ebner W, Krcmar H (2005) Design, implementation, and evaluation of trust-supporting components in virtual communities for patients. J Manag Inf Syst 21(4):101–135
22. Manouchehri S, Söllner M, Leimeister JM (2010) Trust as a design aspect of context aware systems. In: Beigl M, Cazorla-Almeida FJ (eds) Proceedings of the 23rd international conference on architecture of computing systems (ARCS 2010), Hannover, Germany, pp 183–190
23. Patrick AS, Briggs P, Marsh S (2005) Designing systems that people will trust. In: Cranor L, Garfinkel S (eds) Security and usability: designing secure systems that people can use. O'Reilly, Sebastopol, pp 75–100
24. Sutcliffe A (1998) Scenario-based requirements analysis. Requir Eng 3(1):48–65
25. Kotonya G, Sommerville I (1996) Requirements engineering with viewpoints. Softw Eng J 11(1):5–18
26. Herrmann A, Daneva M (2008) Requirements prioritization based on benefit and cost prediction: an agenda for future research.

Paper presented at the 16th IEEE International Requirements Engineering Conference, Barcelona, 08–12 September 2008
27. Boehm B, Grünbacher P, Briggs RO (2001) Developing groupware for requirements negotiation: lessons learned. IEEE Distrib Syst Online 18(3):46–55
28. Pohl K (2008) Requirements engineering. dpunkt Verlag, Heidelberg
29. Chung L, Nixon BA, Yu E, Mylopoulos J (2000) Non-functional requirements in software engineering. Kluwer, Boston
30. Cleland-Huang J, Settimi R, BenKhadra O, Berezhanskaya E, Christina S (2005) Goal-centric traceability for managing non-functional requirements. In: 27th international conference on software engineering. ACM, St. Louis, pp 362–371
31. Gross D, Yu E (2001) From non-functional requirements to design through patterns. Requir Eng 6(1):18–36
32. Withall S (2007) Software requirement patterns. Microsoft Press, Redmond
33. Martin D, Rouncefield M, Sommerville I (2006) Patterns for dependable design. In: Clarke K, Hardstone G, Rouncefield M, Sommerville I (eds) Trust in technology: a socio-technical perspective. Springer, Doordrecht, pp 147–168
34. Melville P, Sindhwani V (2010) Recommender systems. In: Sammut C, Webb G (eds) Encyclopedia of machine learning. Springer, Berlin, pp 829–837
35. Lewis C, Rieman J (1993) Task-centered user interface design: a practical introduction. University of Colorado, Boulder
36. Nielsen J (1993) Usability engineering. Morgan Kaufmann, San Francisco
37. Kotonya G, Sommerville I (1998) Requirements engineering: processes and techniques. Wiley, Chichester
38. Forrester Research (2009) North American Technographics Media and Marketing Online Survey. Forrester Research, Inc., Cambridge
39. Acquisti A, Gross R (2006) Imagined communities: awareness, information sharing, and privacy on the facebook. In: Danezis G, Golle P (eds) Privacy enhancing technologies, vol 4258. Lecture notes in computer science. Springer, Berlin, pp 36–58. doi: 10.1007/11957454_3
40. Salber D, Coutaz J (1993) Applying the Wizard of Oz technique to the study of multimodal systems. In: Bass L, Gornostaev J, Unger C (eds) Selected papers from the third international conference on human-computer interaction, vol 753. Lecture notes in computer science. Springer, Berlin, pp 219–230
41. Muir BM, Moray N (1996) Trust in automation. Part II. Experimental studies of trust and human intervention in a process control simulation. Ergonomics 39(3):429–460