Designing Software Transparency: A Multidisciplinary Endeavor

Larissa Chazette*, Melanie Busch*, Maximilian Schrapel†, Kai Korte‡ and Kurt Schneider*
{larissa.chazette, melanie.busch, kurt.schneider}@inf.uni-hannover.de, schrapel@hci.uni-hannover.de, wendt@iri.uni-hannover.de
Leibniz Universität Hannover
*Software Engineering Group, † Human-Computer Interaction, ‡ Institute for Legal Informatics

Category: Research Preview
Target Group: Requirements Engineers and Researchers

Context and Motivation Software systems support end-users in making daily decisions, thanks to ubiquituous computing. As long as the system behaves in accordance to users' expectations, the user continues to trust the system. In the case of unexpected software behavior, there is often a lack of transparency and comprehensibility, since the user cannot understand the reason for the resulting behavior. In such situations, the system appears like a black box: it delivers results but lacks transparency, since it is difficult for users to apprehend how results were generated.

An extensive variety of requirements can be involved when we consider software transparency. These requirements are, most of the times, interrelated in a complex network, influencing each other. As Requirements Engineers, we believe that, to achieve software transparency, different knowledge areas need to be involved in the requirements process, in a multidisciplinary fashion. Different perspectives need to be assessed in order to correctly understand what needs to be tackled in the project and what it represents in terms of requirements. In this paper, we explore the opportunities of a multidisciplinary work and the possible challenges in achieving software transparency. We use a navigation scenario to explore the possibilities in terms of a system. The expected result of this joint dialogue is to identify challenges and constraints in the requirements operationalization, in the requirements process itself, and discuss adequate ways to overcome them.

Scenario Jane would like to use a navigation app and look up a route on her smartwatch. She is in a foreign city and she chooses the shortest route to a point of interest. On her way, the software suddenly suggests to take another route, even though she followed all the instructions and she is almost at the destination. The system does not present any justification for the change and she is not sure why the system decided for another route. In this scenario, some alternatives can be named to justify the sudden change: the road may be closed because of an incident, or this is due to a technical error, e.g., a radio tower shut down and the GPS is not accurate enough on her device. Jane does not understand what is happening and she asks herself whether she should trust the system anyways.

Understanding system decisions is an essential step towards software transparency. An improved understanding may be achieved through explanations. Explainability, in the software context, is the software ability to provide explanations and clarify questions about, e.g., how the system comes to its decisions. It can be used to increase users' awareness of the reasons behind the system decision making and of data collection. Explainability is a non-functional requirement (NFR) which impacts directly on software transparency [1].

Data transparency is another important aspect to consider towards software transparency. When using such devices, large amounts of data are collected, recorded, and sometimes forwarded. The internal processes, what kind of data is being collected and the purpose of this data collection are not clearly visible to the user. Data has to be collected to provide some of the functionalities in the system. Such functionalities may present several advantages and facilitate the end-users' lives. But end-users also know little about which data influenced the software decisions and the reason they need to share their data.

End-users used to be less conscious about data sharing in the context of software systems. This perception is starting to change as a consequence of public data breaches scandals. With the ever-growing pervasive nature of software, ethical principles that foster software transparency need to be considered in a software project and treated as key design concerns. Some concerns include security, privacy, data management, and explainability. Those quality concerns that help to enable system transparency at many levels [2].

Requirements Engineering The corresponding requirements process comprises the identification of NFRs, their interdependencies and how they will impact on other system qualities and functionality. The

use of disseminated models such as goal models is an essential way to understand the interrelations between the requirements within the system and how they translate into functionalities. It is also possible to assess the trade-offs between what is expected of the system and what is possible in terms of hardware (e.g., smartwatch) and other constraints. Requirements workshops can be held to have a better view about the problem at hand, gathering knowledge from different perspectives. The experience and know-how of participants may facilitate the identification of the challenges and opportunities involved, leading to well-suited and realistic solutions.

Using the scenario above, software engineers worked together with a law scholar and a human computer interaction specialist. We assessed their perspectives on the same problem, in the implementation of two NFRs that help to build more transparency in software: privacy (towards data transparency) and explainability. Next, we present the first step towards this joint discussion.

Human-Computer Interaction An important aspect to consider is the difference between user groups. Each user group has a different need for information regarding the system states and, therefore, different explanation needs. Since we cannot assume that the user has a technical understanding of the system, instructions may be kept as simple as possible. Wrong position data or blocked pathways can be visualized in a map view to increase explainability. However, due to the small display size, there is little space to show information about the current system state. Using Nielsens principles [3], often only arrows are displayed, which results in less transparency. Likewise, errors in route selection cannot be easily traced by the user. Further information, such as explanations of route modifications, can also be given in an auditory way but such feedback methods may also be perceived as obtrusive. Therefore, how explainability will be operationalized and its impact on the interface and on the interaction is strongly dependent on context, user and the system.

Privacy-by-Design Privacy-by-Design refers to the approach that data protection must be proactively embedded into the design and architecture of an ICT system [4]. This approach can save time to improve existing data processing systems and address existing data protection issues. It also protects users' rights, strengthens their confidence in the technologies they use, and ensures organizational accountability from the beginning [5]. Privacy-by-design follows an approach in which the basic settings of the technology are configured so that only the really necessary data is processed. Thus, the principles of data avoidance and data economy are also met and the user can configure all further data processing conditions on his own responsibility.

In practice, this means that the different areas must be coordinated right from the beginning. The objectives, purposes and software requirements, as well as the necessary processing operations, should be compared with and aligned to the legal requirements. The objectives and purposes of data processing should be defined in joint discussions in order to comply with the principles stipulated in Art. 5 GDPR. Moreover, the data processing must be adapted to the requirement of data minimization, as well as technical and organizational measures. Concept notes and data flow diagrams are approaches explicitly suggested to support the analysis of such requirements by data protection experts. For this analysis process, technicians and lawyers need to come together to outline objectives and purposes, mapping the data flows, and designing a coherent infrastructure.

Further Steps We aim to put theory into practice in an interdisciplinary collaboration, with the goal of jointly plan a software prototype in the context of smart mobility. One of the main objectives of this collaboration is to understand how more transparency can be achieved in software systems and what it implies for the requirements engineering process.

We also want to explore what kind of strategies should be developed, as well as which methods should be used, in order to facilitate 1) the interdisciplinary work itself and 2) the analysis of these transparency-related requirements. As next step, we will conduct a workshop for goal analysis, identification of trade-offs, and strategies for interdisciplinary collaboration.

References

[1] L. Chazette, O. Karras, and K. Schneider, "Do end-users want explanations? analyzing the role of explainability as an emerging aspect of non-functional requirements," in *Proceedings of the 2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019.

[2] I. Ozkaya, "Ethics is a software design concern," *IEEE Software*, vol. 36, no. 3, pp. 4–8, 2019.

[3] J. Nielsen, "Enhancing the explanatory power of usability heuristics," in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 1994, pp. 152–158.

[4] A. Cavoukian and M. Dixon, *Privacy and security by design: An enterprise architecture approach*. Information and Privacy Commissioner of Ontario, Canada, 2013.

[5] D. Van Rooy and J. Bus, "Trust and privacy in the future interneta research perspective," *Identity in the Information Society*, vol. 3, no. 2, pp. 397–404, 2010.