Challenges in Rendering and Maintaining Trustworthiness for Long-Living Software Systems

Azadeh Alebrahim
azadeh.alebrahim@paluno.uni-due.de

Nazila Gol Mohammadi
nazila.golmohammadi@paluno.uni-due.de

Maritta Heisel
maritta.heisel@paluno.uni-due.de

paluno - The Ruhr Institute for Software Technology
University of Duisburg-Essen, Duisburg, Germany

Abstract

Trustworthiness plays a key role in acceptance and adoption of software by the end-users. When maintaining long-living software systems, trustworthiness has to be addressed since trust of the end-user is volatile and can change over time. In this paper, we discuss the challenges regarding trustworthiness of long-living software systems. Trustworthiness should be considered in the whole life-cycle of a long-living system, i.e., in all development phases aiming at building trustworthiness into the core of the system at design-time and later maintaining it during run-time. But, our focus in this paper is on challenges in requirements engineering and also planning for the run-time activities, e.g., what are the needed monitor interfaces, what are the planned actions and how are the execution interfaces for performing those actions.

1 Introduction

During the life-cycle of long-living software systems requirements and particularly quality requirements evolve in response to changes. Such changes can have different origins, such as environment, usage profile, and business demands [DKK+12]. Long-living software systems must continuously satisfy their evolving requirements.

Trust is defined as a "bet" about the future contingent actions of a system [Szt00]. The components of this definition are belief and commitment. There is a belief that placing trust in a software or a system will lead to a good outcome. Then, the user commits the placing trust by taking an action on using the system [GMPB+13]. This means, when a user decides to use a service, e.g., a health care service on the web, then she is confident that it will meet her expectations. Trust is subjective and different from user to user, e.g., organizations require confidence about their business critical data, whereas an elderly person using a health care service (end-users) may be more concerned about the usability. These concerns manifest themselves as trustworthiness requirements. Thus, software systems and services need to be made trustworthy to mitigate the risks in engaging software systems and trust concerns of their users.

Trustworthiness is a quality of the system that potentialy influence the trust in the system in a positive way. Trustworthiness has been used sometimes as a synonym for security and sometimes for dependability.

However, security is not the only aspect of trustworthiness. Most existing approaches have assumed that one-dimensional properties of services lead to trustworthiness of such services, and even to trust in it by users, such as a certification, the presence of certain technologies, or the use of certain methodologies. Maintaining security requirements for long-living software systems has been studied in [GRB+14]. However, trustworthiness is rather a broad-spectrum term with notions including reliability, security, performance, and usability as parts of trustworthiness attributes [MHX12]. Trustworthiness attributes are domain and application dependent, e.g., in health care applications, the set of attributes which have primarily been considered consists of availability, confidentiality, integrity, maintainability, reliability and safety, but also performance and timeliness.

Likewise, trust is dependent on different factors, i.e., two different stakeholders may have different levels of trust for the same system depending on factors such as age, gender, cultural background, or the level of experience. Trust concerns of the end-user are volatile and can change over the time [GMPB+13]. This introduces a challenge for long-living software systems, in which the long period of time can cause new trust concerns of the user based on her experienced incidences or advancing in age. Hence, trustworthiness requirements evolve by either new trust concerns of the end-users over a long period of time or by modifications in the environmental settings such as change of third party service provider, infrastructure migration, and regulations can negatively affect trustworthiness properties of long-living software systems. For example, if you consider service provisioning or service substitution, this should be in conformance to Service-Level Agreements (SLAs) [ADC10]. SLAs are documented in a contract between the customer and the service provider. A new service substitution has to guarantee the agreed (trustworthiness) SLA. But, new service or service provider policies may not respect the established trust conditions. The user should have the capacity to trust evaluation in order to enforce the agreed service terms and trust relation conditions. Figure 1 shows the boundaries of the system and its context with involved actors as well as the influencing factors to established trust relation. It illustrates exemplary origins of the changes which may affect the trustworthiness of the system with respect to the established trust relation.

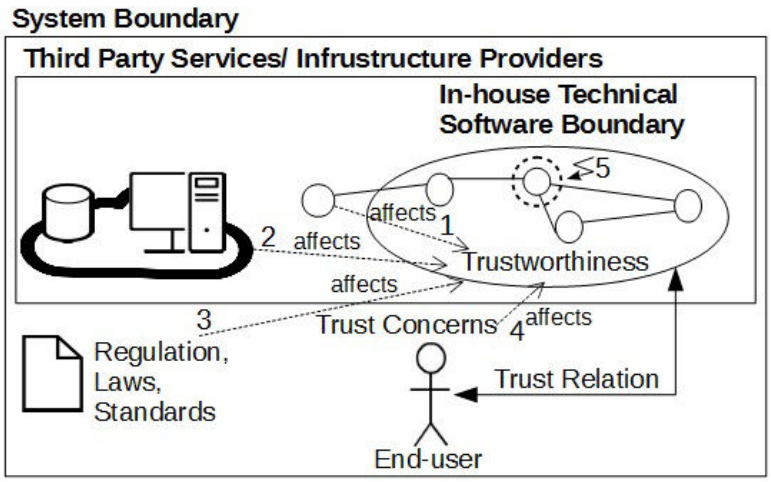

Figure 1: Boundaries of the System for Trustworthiness

2 Challenges in Addressing Trustworthiness

As mentioned before, the major challenge is to maintain the trustworthiness over a long period of time considering long-living software systems. We break down the main challenge to the following questions which need to be answered:

How can trustworthiness-related issues be identified? As illustrated in Figure 1, trustworthiness properties of the software can be compromised by different parties within the system boundary. For example, substitution of one service which is a valid adaptation action with regard to performance might cause violation of trustworthiness in terms of privacy (see 1 in Figure 1). Therefore, analyzing the effect of applying the adaptation actions with regard to trustworthiness of the software has to be performed before applying the adaptation.

How can a situation causing a change in trustworthiness be identified? There are different kinds of changes in the system which can affect trustworthiness. Some of them are observable by the software, e.g., reconfiguration, substitution of third party services (see 2 in Figure 1), etc. Other kinds of changes are not observable by the software, such as changes in regulations (see 3 in Figure 1) and trust expectations of the user

(see 4 in Figure 1). Examples for such a situation are:

- A user has already engaged using a software. She obtains awareness about the consequences of sharing sensitive data. Therefore, privacy plays a role in her trust concerns now. This may lead to changes in the trustworthiness requirements (see 4 in Figure 1).
- A software provides new features to the user requiring access to the location data. The user is not willing to share her location data. Updating the software with the new features is violating the trustworthiness as the old features did not require access to the location data (see 5 in Figure 1).

How can the new trust concerns of the users be gathered? Since the trust concerns are not directly observable by the software, a mechanism should be developed to capture the new trustworthiness requirements of the user timely before the trust shaking incidents happen.

Our presented challenges are upon to the trust relation establishment, i.e., we identify the challenges in maintaining and keeping thresholds in order to respect and avoid any violation in terms of trustworthiness. Since any violation in those trust promising properties may lead to lost of trust. In our above mentioned example, we assume that the user has placed her trust based on some analysis of the service provider past behavior, e.g., history, reputation and etc. The user would not have committed to use the service from a provider who has violated the user's privacy in the past. Since, the usage patterns of service providers may change in the future and those changes are not always in conformance to established trust relation. Therefore, there should be capabilities to identify and evaluate the trustworthiness of long-living software systems. Providing the capabilities for the user to easily analysis in making decision whether engage a system is also important, but it is out of our interest for this work.

3 Conclusions

During the life-cycle of a software, (quality) requirements evolve in response to changes. Long-living software systems must continuously satisfy their quality requirements. This becomes even more critical and challenging when considering trustworthiness requirements. In this paper, we identified the challenges for maintaining trustworthiness of long-living software systems. To address those challenges, continuous monitoring of trustworthiness requirements changes is needed. As consequence, an evolution mechanism in order to recover the desired trustworthiness level of the system has to be provided. Assessing requirements of long-living software systems with respect to trustworthiness is worthy since software design is directly affected by its requirements. Such an assessment offers useful support for requirements engineers upon the reported incidents that shake the trust. Further studies in introduced research directions are needed.

References

[ADC10] M. Alhamad, T. Dillon, and E. Chang. Conceptual service level agreement framework for cloud computing. In *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, pages 606–610, April 2010.

[DKK+12] Z. Durdik, B. Klatt, H. Koziolek, K. Krogmann, J. Stammel, and R. Weiss. Sustainability guidelines for long-living software systems. In *Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 517–526, Sept 2012.

[GMPB+13] N. Gol Mohammadi, S. Paulus, M. Bishr, M. Metzger, H. Könnecke, S. Hartenstein, T. Weyer, and K. Pohl. Trustworthiness attributes and metrics for engineering trusted internet-based software systems. In *Proceedings of the 3rd International Conference of Cloud Computing and Services Science CLOSER*, pages 19–35, 2013.

[GRB+14] S. Gartner, T. Ruhroth, J. Burger, K. Schneider, and J. Jurjens. Maintaining requirements for long-living software systems by incorporating security knowledge. In *Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE)*, pages 103–112, Aug 2014.

[MHX12] H. Mei, G. Huang, and T. Xie. Internetware: A software paradigm for internet computing. *Computer*, 45(6):26–31, June 2012.

[Szt00] P. Sztompka. *Trust: A Sociological Theory*. Cambridge, UK: Cambridge University Press, 2000.