Security and Privacy as Hygiene Factors of Developer Behavior in Small and Agile Teams

Kai-Uwe Loser, Martin Degeling

Security and Privacy as Hygiene Factors of Developer Behavior in Small and Agile Teams

Kai-Uwe Loser, Martin Degeling

Institute of Work Science, Ruhr-University Bochum

martin.degeling@rub.de

Abstract. User motivations are often considered in human computer relations. The analysis of developer behavior often lacks this perspective. Herzberg's distinction of motivators and hygiene factors adds a level for the analyses of those sociotechnical phenomena that lead to skipping of security and privacy requirements especially in agile development projects. Requirements of security and privacy are not considered nice-to-have, but as necessary hygiene factors of systems attractiveness, motivation for extra effort is low with respect to those requirements. The motivators for developers – functionality that makes a system special and which is valued by customers and users are dominant for the decisions about priorities of development – hygiene factors like many security requirements get a lower priority. In this paper we introduce this theory with relation to known problems of (agile) development projects with respect to implementing security and privacy. We present this with a case study of mobile app development in a research project that we analyzed by security and privacy aspects.

Keywords: security and privacy, agile development, Herzberg's theory, motivation.

1 Introduction

Agile development aims to quick results to fulfill requirements and user needs for early prototypes and feedback. But with the speed of development security and privacy requirements tend to be overseen. This is especially relevant for IT landscapes like those of mobile apps which is shaped by high innovation pressure and a large number of similar applications developed by very small teams or individuals. At first security awareness approaches focusing on development teams seem appropriate to improve this situation, but in many cases they are found to be not sufficient. Analyzing this situation with a background on work motivation can help to find solutions to this problem. This paper aims to show the relevance of this background with an analysis in relation to the motivator-hygiene-theory of Herzberg. To contextualize the theory we report on a study of an agile development project where we analyzed the security and privacy awareness in comparison to the implementation details of the developed systems. Results are similar to assumptions that others as we

argue to increase intrinsic motivation of software developers. Our study was part of a large research project on technology enhanced learning, where a strong relevance of privacy and security was recognized early due to the application domains (e.g. healthcare). The research and development project was split up in several small development teams, each developing (mobile) applications for different kinds of for learning in work situations. The overall project had a strong commitment on security and privacy issues and several steps were taken to actively support and foster those topics in the developed applications. Nevertheless only a few of them in the end incorporated them in their basic design and early versions. In the final phase of the project we found various requirements being open issues, we then took a closer look at the list of requirements and from our personal involvement contextualized their history.

The next two sections will give an overview of Herzberg's original theory and related work. The following three sections will present the case study, which we will analyze in relation to Herzberg's theory. We will conclude with some ideas on solutions to the problem situation that becomes obvious by the analysis.

2 The Herzberg Two-Factor Theory

As a psychologist and work scientist Frederick Herzberg conducted several studies about satisfaction at work in various countries [1, 2]. He found that one set of needs of employees is closely linked to work satisfaction another to work-dissatisfaction. The hypothesis was that satisfaction is a key to work motivation and work performance.

The first group of needs and factors is called *motivators*. Motivators for work are for example challenging work, recognition of the work by others or self-determination at work. Improving motivators, Herzberg suggested, can lead to higher job satisfaction.

The second group of factors is called *hygiene factors*. Examples for hygiene factors are salary, workspace qualities and conditions or work-life balance. Hygiene factors are supposed to create dissatisfaction if they are not reaching a certain level. Raising those factors beyond that level will not lead to improved satisfaction.

Within the field of organizational psychology the theory was developed further and integrated into more elaborate theories of motivation. Miner [3] refers to various critics to the empirical methods. Especially the assumed link between satisfaction and work motivation and therefore work performance has been critiqued to be too simplistic and studies show that some results are no longer tenable [3]. But there are still several researchers referring to the theory e.g. Bassett & Jones [4]. Especially the simplicity leads to an easy analysis of phenomena with relation to motivation and can help to find ways to improve it. Herzberg himself proposed "Job Enrichment". Therefore the theory was adopted in several domains. In general it is used as an analogy e.g. for user experience design [5] or behavior analysis [6].

This distinction between motivators and hygiene factors can also be transferred to requirements in software engineering. Hygiene requirements may include usability aspects or functionality that is expected, like a user interface that includes known elements and forms with buttons and menus and basic features to work without errors.

Some other requirements may become features that seem to be the motivating aspects for a buy or use decision like innovative features other applications do not offer or a user interface that stands out from others. This complements a "nice-to-have" vs. "mandatory" distinction. At first this seems similar, but in detail it regularly leads to different clusters and to a higher priority to "Nice-To-Haves". Many hygiene aspects would be considered as mandatory on the general level. Nevertheless importance of an effort in fulfilling the detailed requirements and motivation of developers to implement these may be low. Usability and security requirements are examples of this: protection for example by encrypted data transfer is – although simple and should be mandatory - often omitted because the implementing developer will not get positive feedback when fulfilling such a requirement. Developers realize that certain functionality is motivating for customers or users to use an application (Motivators), whereas others are not provoking positive feedback once (Hygiene). So customers' views are often closely linked to the developer perspective. We therefore use the distinction of Herzberg's work motivational aspects as an analogy to separate different kinds of requirements. We think that differentiating requirements into the two categories will help tackle especially the hygiene requirements because those will need extra effort to become appropriate attention. Having the distinction of hygiene factors and motivators in mind, we will go on with the problems of security and privacy engineering and then describe a case study to illustrate the application of Herzberg's distinction especially for security and privacy requirements.

3 Security Engineering

Security Engineering aims at structured and controllable processes to implement reliable systems and applications. Common goals of security engineering are confidentiality, integrity and availability of data. These goals focus on the infrastructure and the organization as a whole. From a privacy engineering perspective new approaches [7, 8] argue to integrate unlinkability, transparency, intervenability in a similar way as privacy protection goals into software engineering.
Several researchers ([9–11]) have identified that secure software not only requires secure algorithms but that their usage has to be fostered in software engineering processes. To be able to do this it is crucial that software developers are aware of security and privacy issues as well as of possible solutions. Although they are trained to do so it is necessary that relevant topics are regularly practiced.

There are publications (e.g. [10]) offering a comprehensive list of possible solutions, best-practices and approaches to develop secure information systems. They mainly include the strategy to do proper system engineering, make an extensive risk analysis and threat assessment, improve an application to avoid the problems identified and to introduce various steps to control the process. Most approaches suggest adding additional process steps in the waterfall software engineering process and propose to introduce security assessments and threat analysis. These work for a lot of situations but fail for those projects that do not have a strict outcome oriented management. Especially in agile projects where requirements change over time, are re-evaluated and re-prioritized or user stories are developed continuously require more flexible approaches. Siponen et al. [12] for example propose to introduce

security aspects into feature driven development by identifying and classifying security relevant actors and objects for each requirement. Boström et al. [13] developed a method to include security assessment practices in XP programming and Besznosov et al. [14] propose to stick with those common agile methods and artifacts that allow security reviews without extending documentations and use semi-automated testing to assure security in software.

In contrast to these rather formal techniques that are integrated with agile software development models we want to focus on the behavior of software developers. Firstly because they have to translate user requirements for security and privacy into the technical domain. And secondly because in agile development approaches and small teams where development models are not followed strictly developers have to make decisions about how the requirements are implemented and are often responsible for assessing the trade-offs between for example security and usability [15]. In addition their knowledge of the domain and technologies and presumptions about the importance of a specific requirement is crucial for the decisions about implementing privacy and security. As decisions made by developers in agile projects are much more related to individual knowledge and perception of a problem motivational factors become more relevant as they decide on whether or not and in which depth a security and privacy aspects are addressed.

4 Case Study Description

Our work is based on the experience in a research project on technology enhanced learning in workplace situations. It included several smaller agile software development projects building apps for various domains. About 20 applications (apps) were developed to support employees in learning about their work practice by capturing data from day to day practice. The apps were based on multiple platforms from desktop to web-based systems and mobile apps for tablets and smartphones. Development teams consisted of 1-3 people that developed around 20 apps in three years. From the beginning security and privacy were important issues for the development since capturing work performance data to allow employees to learn from it, implies that this data has to be persistently stored and evaluated. The apps were developed in domains ranging from employees in hospitals, care homes or sales agents in an IT company. They included automated to manual tracking of work e.g. with a tracking application on personal computers that records how long which application with which file was used; recording who meets whom during a work day with proximity sensors people were wearing; allowing users to frequently capture their mood during meetings by clicking mood maps or asking them to write down their personal views and self-assessments after talking to clients or patients. To elaborate on the idea of the project and the awareness and presence of security and privacy issues we will present one app as examples where the security shortcomings are also visible in the next section.

Right from the beginning severe privacy and security implications were identified. The threat model includes that a specific user and practice can be identified and analyzed by abusers from outside a company or organization, which are interested in work practices of a competitor. An internal threat is that employees fear the misuse of

their data by their employers and supervisors trying to control their work and assessing their efficiency which is especially important due to the nature of the project were users were free to participate or not. This leads to a number of higher level requirements and discussions about privacy and security problems and possible solutions for individual apps as well as for an overall framework connecting all developed apps.

4.1 Approach to raise security awareness

In the beginning of the project, we found security awareness to be an important part of approaching the goal of secure systems. One basic idea was that, if all developers were aware of issues, the apps' quality with respect to those requirements will meet the standards. The following list gives an overview of the actions taken:

1. Organizational support to target security and privacy issue

(a) In the original work plan a work package was dedicated to privacy, security and system integration testing as an activity all development partners have to participate in.

(b) Project management fostered discussion about this topic by proposing workshops at every project meeting covering privacy and security issues.

1. Raise developers awareness

(a) Again workshops within the groups of developers where the topics were discussed in detail.

(b) A "developers' cheat sheet" for privacy and security measurements was developed including condensed information about when and how to encrypt data and what data (not) to collect.

1. Make security and privacy assessments

(a) Later on we did some example privacy and security assessments for a sample of the apps and published the results to the developers. Additional requirements were discussed and developed from the results. This led to a continuous presence of the topics.

(b) We made a user survey to generate more requirements from the user level that were mostly about personal control and transparency [16]. The evaluation focused on revealing details about the difference of assumed privacy awareness of users and actual behavior. The results have shown that most users are very critical about privacy and security in general and especially have a limited trust in their employers to be honest in how they use their data.

4.2 Application Example

Apps where developed for different work contexts. Some of the domains were less critical from a security point of view than others. One app for example was used to aggregate information from public sources like twitter to allow members of emergency teams of large mass events to support reflection about the current situation to better coordinate the helpers involved. Since we want to focus on the requirements for privacy and security and due to the lack of space we focus on an app that had higher security needs.

Requirements engineering took place in workshops that were held with users and other stakeholders that focused on the applicability and usability of the app. As we followed an agile development approach the first ideas were based on storyboards produced in workshops with application partners. Based on those storyboards user stories were written and adapted after prototypes realizing those stories were tested.

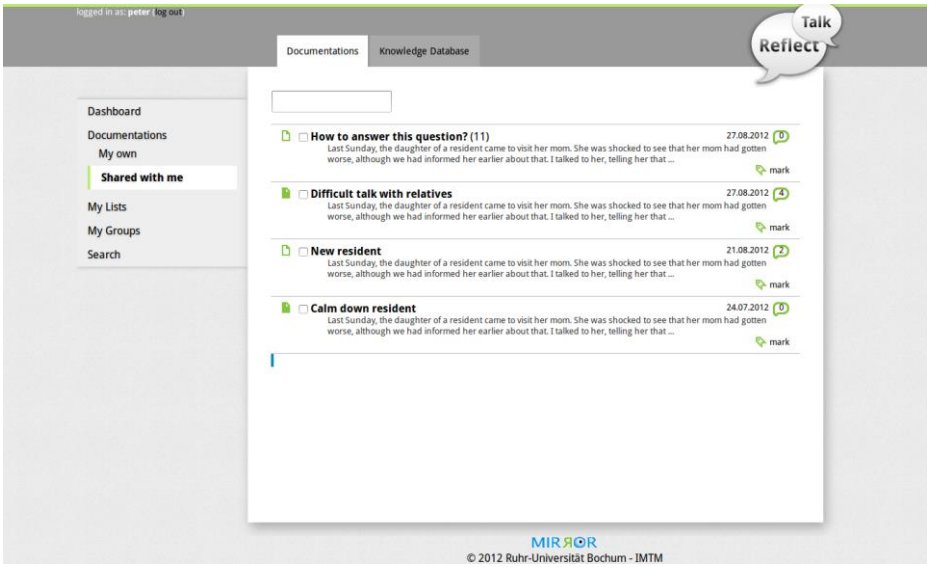

Fig. 1. Screenshot of the TalkReflection App showing a list of documented conversations.

The TalkReflection app (s. figure 1) is used by staff in a hospital and homes for dementia care to document conversations they have with patients and relatives, especially to document discussions with them and record rationale of care decisions. They document the conversation by stating, what it was about, who was involved and what the decisions were. In addition users are asked to document their own perception of the conversation, how they felt, how the conversation partner reacted on an emotional level and what hypothesis about reasons for their own mistakes they have. Afterwards the users are able to share their documentation and comment on others'. In more detail the idea behind the app is that there is on the one hand a demand to document those conversations and rationale for reasons of liability. If, for example, some sort of accident happens after a decision a physician made with a relative about a patient who cannot be asked themselves, it would be positive if details about the

decision and the context of the decision are documented. In addition the app can be used by staff members to reflect on how they behaved and felt during a conversation. This was found to be useful as for most physicians it is not part of their training to learn how to cope with difficult conversations and how to improve their reactions towards relatives that are often stressed by the situations and diseases of their relatives.

The app, similar to others developed, has serious implications with regard to security and privacy that were categorized on two levels. From **an organizational perspective** there were high requirements for the security goals integrity and confidentiality. In the first place the organizations wanted to protect their company data; in the second, and especially for companies working in the healthcare sector, they also wanted to protect third party data like information about clients or patients..

From **the users' perspectives** a separate group of questions about privacy and the security of the applications arose as employees thought they might be used by their employers to control their work practice and may sanction certain kinds of behavior. Therefore features had to be included addressing this kind of problems by access control and prevention of unauthorized access and usage of the data.

After the overall concerns and contexts were evaluated several workshops were held with experts and stakeholders from the hospital itself including participants like the data protection official, the director of IT-security and the works council. In addition external IT-security experts were consulted as the rather small development teams of 2 to 3 persons were not considered as experts in the field.

To the time of writing only few of the security requirements are currently implemented and those that were considered, were supposed blocking the field experiments. The IT department required encrypted connections for the apps since they only opened up those ports in the firewall.

After a large number of prototypes the apps are evaluated and used by the application partners on a regular basis now. Nevertheless they are still considered "beta" and the following sentences are cited from a statement of a developer on the current status of privacy and security: "The privacy aspect has been considered within the planning of the application. Although there is no detailed plan for privacy safeguarding at the moment, we considered multiple approaches within the applications architecture. Thus we are primed for different kinds of privacy safeguarding." The developers' excuse is that still everything is possible – but the requirements are not met currently. The following section will present various examples from the case study to illustrate and motivate the distinction of requirements into the hygiene and motivator categories.

5 Discussion

To come back to Herzberg's theory of motivators and hygiene factors we consider security and privacy requirements in general similar to hygiene factors and have to compare them with motivators. Potential users will not use a system just because it has well implemented privacy and security mechanisms, but it will inhibit them if it is not assured or will result in discontinuity of use if disclosed. We assume that security and privacy requirements are usually seen as a necessary presupposition in a world

where not everybody is able to test if all the requirements are met. Since they regularly are not visible to end users, they are always assumed to be implemented. From a developers' perspective in a world where users are free to choose to use or not to use an application the product acceptance (e.g. number of installations from mobile app stores) has become one of the main motivators. In this logic privacy and security should be implemented but are at the same time not resulting in positive feedback since they are often not visible at all. Therefore they can be considered hygienic factors.

We want to discuss this hypothesis in context of some observations of decisions made during the development of the described web application.

5.1 Motivator type requirements: a personal link

Motivators in our case are bound to those requirements which are related to the innovative ideas, aspects that make the prototypes new, interesting and exciting. While these are motivators especially for users that use an application because of its novelty this, in the same term, motivates developers to implement them. The observations showed that this especially holds true if a requirement is linked to a personal promoters. If a feature is requested by a (possible) user within a workshop with developers, this specific technical implementation gets linked to a personal relationship; a personal motivation for implementing grows.

5.2. Hygiene type requirements: implemented when easy

Also risking negative feedback several security and privacy related requirements were not implemented in the prototypes. Although implementation is simple using the right tools only those requirements were implemented that were easy to achieve e.g. encrypted data transfer through https and certificates for xmpp servers. Other, like data storage encryption, were not implemented although it might, in the long term, hindering wide spread application.

One explanation for this behavior is that security and privacy requirements are develop top down. Security and privacy and related requirements are derived after the functional requirements are clarified. The functional requirements especially in agile development processes are always bound to people and their problem situation. Security and privacy on the other hand are difficult to link specific situations but related to overall needs, which leads to a lack of focus on these requirements.

5.3 Not implemented requirements: organizational solutions

Especially agile development leads to regular priority decisions. Therefore more complex security requirements or requirements which need a higher effort were skipped with various excuses and strategies to defer the task. An example is the ability for users to (easily) delete their personal content, which was identified as a privacy requirement. Since databases were regularly reset after a prototype test was conducted, some apps are still missing the possibility to delete content by users. And

they are not only missing a delete button but also the routines that handle deletions and keep data structures and for example threaded discussions intact.

Other requirements were moved from technical to organizational solutions. For example one proposal was to build modules that automatically scan written documentations for real names of patients and relatives to change them to an unlinkable pseudonym. Since that was never implemented, the requirement was omitted and replaced by additional rules for the handbooks saying users are not allowed to use real names and instead should use codes generated from room numbers and initials. Another common pattern for questions of how much data may be collected was to answer them with additional consent forms having the idea in mind that any data collection is legally allowed and almost every data processing is possible if the users decided to allow it in the first place. Those examples are instances of strategies leading to a transfer of effort (and responsibility) to others. If a requirement is important enough, somebody else can implement that.

5.4 Ideas for approaching the avoidance of hygienic requirements

Theoretically, requirements bound to motivators naturally become a higher priority while for the implementation of hygiene factors processes need to be actively redesigned. While for some basic features like encrypted storage technical solutions might be feasible that just make implementation easier. The general problem that a sophisticated implementation is not visible to any user, cannot be solved that way.

Instead requirements should be linked to motivators that foster *intrinsic motivations* of developers. One motivator is the personal link to a user in the process: are there promoters of a specific requirement? Somebody that you offend, if you do not fulfill a specific requirement? A better motivation would be if you would feel to help somebody personally. The positive motivations (promotion) are better motivators than negative outcomes (prevention) [17]. The question a developer should answer is: "Is there anybody giving positive feedback if security requirements are met?"

Another idea can be borrowed from motivational approaches in other domains like User Experience Design, to improve hedonic aspects: with the idea of "Funology" [16] in mind another approach would be to make it a game. Two groups of app-developers could reciprocally attack the others' system, leading to improvements based on competition.

These arguments on motivational considerations can also be discussed in relation to proposed approaches in the literature. Instead of adding security considerations to feature cards as proposed by [12] which are already the formalized perspective of involved people and features, it might be better to start earlier on the level of user stories and including security aspects in interviews for requirements engineering. Discussing implications of specific security flaws with data subjects may give security requirements a face and a personal background to add other levels of motivation and a motivator background.

6 Conclusions

The discussion of the motivational aspects is trying to focus on the motivational aspects coming from the distinction between motivators and hygiene factors. As mentioned earlier the theory was attacked to be too simplistic and also in this case some forces of development are not considered in detail, and some problems might have been avoided by more strict development processes for example. On the other hand the case study described regularly visible real world phenomena and points to other approaches to overcome the problems.

Our experience from the case study is that security awareness and external analysis is not sufficient to ensure that especially agile development projects adequately consider security and privacy aspects although developers need guidance in developing secure and privacy aware systems. We tried to explain the problems recognized with motivational aspects. In development processes developer motivations are not sufficiently reflected. From this point of view security requirements have to be considered as hygienic factors: users and customers do expect their fulfillment but do not actively promote them. Our experience shows that with limited resources of time this kind of requirements will be deferred and easily skipped. Even though developers are highly aware of the security and privacy needs this is regularly the case.

We presented an example and discussed motivational aspects as one part of socio-technical topics of development of privacy and security. That awareness for security and privacy issues in software development is not enough when developing apps in small teams became obvious. The results point to improvements in development processes including the categorization of motivators and hygiene factor requirements, where different kinds of process controls should be followed. Knowing that Herzberg's Distinction is a too simplistic view on motivation for work one may state that especially the simplicity seems to make it useful, like other abstractions. Further research in relating theories about motivation to security processes and development processes seems to be an important extension of this work.

References

1. Herzberg, F., Mausner, B., Snyderman, B.B.: Motivation to work. Transaction Publishers (1959).
2. Herzberg, F.: The motivation-hygiene concept and problems of manpower. Pers. Adm. (1964).
3. Miner, J.B.: Organizational Behavior 2: Essential Theories of Process and Structure. M.E. Sharpe (2005).
4. Bassett-Jones, N., Lloyd, G.C.: Does Herzberg's motivation theory have staying power? J. Manag. Dev. 24, 929–943 (2005).
5. Hassenzahl, M.: Experience Design. Technology for All the Right Reasons. Morgan and Claypool, Penn State University (2010).
6. Crompton, J.L.: Adapting Herzberg: A conceptualization of the effects of hygiene and motivator attributes on perceptions of event quality. J. Travel Res. 41, 305–310 (2003).

7. Hansen, M.: Top 10 Mistakes in System Design from a Privacy Perspective and Privacy Protection Goals. In: Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., and Russello, G. (eds.) Privacy and Identity Management for Life. pp. 14–31. Springer Berlin Heidelberg (2012).
8. Rost, M., Pfitzmann, A.: Datenschutz-Schutzziele—revisited. Datenschutz Datensicherheit. 33, 353–358 (2009).
9. Sodiya, A.S., Onashoga, S.A., Ajayi, O.B.: Towards building secure software systems. Issues Informing Sci. Inf. Technol. 3, (2006).
10. McGraw, G.: From the ground up: The DIMACS software security workshop. Secur. Priv. IEEE. 1, 59–66 (2003).
11. Anderson, R.: Security engineering: a guide to building dependable distributed systems. Wiley, Indianapolis (2008).
12. Siponen, M., Baskerville, R., Kuivalainen, T.: Integrating Security into Agile Development Methods. Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS '05. p. 185a (2005).
13. Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., Kruchten, P.: Extending XP practices to support security requirements engineering. Presented at the (2006).
14. Beznosov, K., Kruchten, P.: Towards agile security assurance. Proceedings of the 2004 workshop on New security paradigms. pp. 47–54 (2004).
15. Spiekermann, S., Cranor, L.F.: Engineering Privacy. IEEE Trans. Softw. Eng. 35, 67–82 (2009).
16. Degeling, M., Ackema, R.: D9.1 User studies on privacy needs, privacy model and privacy guidelines. (2011).
17. Higgins, T.: Promotion and Prevention: Regulatory Focus as a Motivational Principle. Advances in Experimental Social Psychology. Academic Press (1998).