Can Security Become a Routine? A Study of Organizational Change in an Agile Software Development Group

Andreas Poller
Fraunhofer SIT
Darmstadt, Germany
poller@sit.fraunhofer.de

Laura Kocksch
Fraunhofer SIT
Darmstadt, Germany
kocksch@sit.fraunhofer.de

Sven Türpe
Fraunhofer SIT
Darmstadt, Germany
tuerpe@sit.fraunhofer.de

Felix Anand Epp
Fraunhofer SIT
Darmstadt, Germany
fepp@sit.fraunhofer.de

Katharina Kinder-Kurlanda
GESIS, Köln, Germany
katharina.kinder-kurlanda@gesis.org

ABSTRACT

Organizational factors influence the success of security initiatives in software development. Security audits and developer training can motivate development teams to adopt security practices, but their interplay with organizational structures and routines remains unclear. We studied how security consultancy affected organizational routines in a software development group. Security consultants tested their product, reported vulnerabilities, and delivered a security training. We followed the group during and after consultancy events. As a result of the consultancy, group members improved their understanding of security issues, but could not effect a change of routines within the given organizational structure. They handled vulnerabilities in a stabilization routine without changes in feature development, where security remained intangible. Interestingly, group members acknowledged an unfulfilled need for change but defended the structure inhibiting change. Security initiatives need to consider this interplay of structure and situated practice, and manage change in addition to providing expertise and tools.

Author Keywords

Software development; software development teams; structure-and-agency duality; organizational factors; organizational routines; IT security; organizational change; Scrum; agile development; penetration test; security training

ACM Classification Keywords

D.2.9 Management: Programming teams; K.4.3 Organizational Impacts: Computer-supported collaborative work; K.6.5 Management of computing and information systems: Security and Protection

CSCW 2017, February 25–March 1, 2017, Portland, OR, USA.

http://dx.doi.org/10.1145/2998181.2998191

INTRODUCTION

Developers need to address IT security concerns in their work, but useful approaches to promote and support security work are hard to find. The CSCW community has repeatedly inquired into different facets of software development settings [21, 30] and collaborative tools for development [14], but little research has focused specifically on IT security concerns. However, recent research investigates security tool design and adoption, and finds that introducing security tools is made difficult by the intangibility of their benefits [36] and possible interferences with other programming work [37]. These studies indicate that changing software development to integrate security activities depends on human and organizational factors [34]. But we still lack the big picture of what these factors are and how they can be addressed not only in the design of security tools but also by the people promoting security activities in organizations.

Within a long-term empirical study we explored the following research question: *How are a development group's work routines affected by an external security audit and training?* We investigated a mature, yet security-inexperienced software product group whose product underwent a first-time security audit and who participated in a security training. Audits and trainings are common attempts to raise security awareness among developers and to provoke changes in practices to integrate security work into development [22, 13, 16]. Both measures are often part of a contracted security consultancy package, as in our case.

At the researched company the security consultancy was organized and steered by an internal security initiative that aimed to promote security practices across product groups. This initiative was a prospective measure as the company had not yet faced any significant security incidents or complaints. The security initiative thus did not aim to prevent a reoccurrence of past incidents or to fix an urgent problem, but rather to trigger *endogenous* change within the product group. Hiring security consultants was a considerable investment intended to support this change. However, even the security experts driving the security initiative raised the question to what extent current practices and structures within the prod-

uct group could actually support continuous improvement of security activities after an initial training and audit.

As a source of endogenous change, the literature points to organizational routines, which are the "repetitive, recognizable patterns of interdependent actions" [9] taken by various actors within an organization. Traditionally organizational routines have often been associated with inertia and inflexibility, a concern shared by the people driving the security initiative. However, research over the last decade has developed a more differentiated view and focused on how the interaction of structure and situated practices can unfold dynamics within an organization that eventually lead to sustainable change over time [9, 17]. The agile Scrum framework shaping the routines in our case allows for exactly such an interplay of structure and practice: Defined roles and set collaboration rituals create a space for self-organization and continuous improvement.

The security consultants' task was not to advise the product group on how to change their organizational routines, but to challenge and teach them about security issues of their product. In this scenario, only the group's established routines themselves could engender a robust change over time. To research this process, we followed the product group for 13 months collecting data in questionnaires, by observing a multi-day training workshop, by reviewing documents, and by conducting interviews. We used the collected data to inquire into the practices of product group members, the structures shaping these practices, and how both evolved during and after the consultancy.

We witnessed how security awareness among product group members rose after the consultancy. Group members believed that security should be an integral part of development work and tried to adapt routines. But changes could not be sustained over time despite the fact that routines were rather flexible in the self-organized scrum teams. This self-organization structure, however, led both developers and managers to frame security as a mere implicit quality issue instead of an explicit requirement to the software. Hence, managers expected developers to address security—like every other quality issue—only within their teams, and there was no further necessity for interactions between managers and developers about this topic. This perception of security together with the inherent intangibility of security work outcomes led to development practices not being affected by the consultancy. Group members were concerned about this result, but their ineffective view of security was an implication of their organizational structure they themselves considered valuable.

We argue that integrating security work into development routines requires active management of change by the people driving security initiatives. We suggest that security experts need to combine their security knowledge with insights of a company's organizational structures and prevailing development practices. Organizational routines can be a source of change towards security work if triggered in an effective way, but can also inhibit attempts to promote security activities if perceptions of security are misaligned to routines' specific characteristics. In our case, neither the internal security initiative of the company, nor the developers or the managers found an approach to security that fitted their routines shaped by Scrum, labor division, and self-organization of teams.

BACKGROUND

Software development is a source of security vulnerabilities. Software-developing organizations therefore need to pay attention to security and apply secure development practices. However, managing software development is a challenge in itself even without the added complexity of security work. Agile methodologies like Scrum are commonly applied today and have advantages compared to other approaches, but how to integrate security practices into them remains an open question.

Developing Secure Software

Security—dependability in spite of efforts to compromise [12]—differs from other requirements in its aim to prevent intelligent, adaptive adversaries from achieving their goals [29]. Vulnerabilities are defects or features of software that allow attacks to succeed. They often exist in side-effect behavior irrelevant in normal use but exploitable through unanticipated inputs or actions [33]. Vulnerabilities are therefore easily introduced inadvertently: Any technology, design, or implementation decision in any part of a system can have security implications [22, 13], but neither for developers nor for users are these vulnerabilities readily visible. To find vulnerabilities one has to apply a particular "security mindset" [29, 7] and look for ways to manipulate a system. Numerous security defect patterns are known, but their consequences vary by context and the risks even of known problems are therefore hard to assess.

To prevent or to uncover and fix vulnerabilities in their products, software vendors need to employ activities like threat modeling and tools like code analyzers. Guidelines such as Microsoft's Security Development Lifecycle (SDL) [16] and McGraw's security touchpoints [22] recommend various security practices, but fall short of discussing how to get an organization and its development teams to adopt them. Security work concerns developers, who have to integrate tools and activities into their day-to-day work, but also the organization as a whole, which has to steer and support security efforts. Two factors make security work difficult to manage: The benefits of security work can be difficult to see due to its delayed effect [36], and security is only one of many goals.

Security audits by consultants and developer trainings are two commonly recommended practices [22, 16] and often among the first adopted by companies starting a security initiative. Both aim to influence developers' practices: Training raises security awareness and imparts knowledge, and security audits uncover vulnerabilities introduced and overlooked in the past. In a security audit consultants combine penetration testing [24, 10, 3] and code review to find vulnerabilities. They report their findings to the developers, who should then not only fix the specific reported security defects but also analyze the root causes and devise ways to prevent reoccurrence of similar vulnerabilities. Anecdotal evidence suggests, however, that development teams often cut corners and only fix

the specific security defects reported. McGraw [22] hence calls penetration testing "a very common but often misapplied software security best practice," criticizing that "the developers don't learn anything profound" by only patching the holes uncovered by a security consultant. Why this happens and whether an added training workshop helps developers to learn from audit results remains unclear.

Self-Organizing Agile Teams

Agile development practices [1, 15, 25, 35] shaped the organizational structure in our setting. The studied product group organized its work using Scrum [27, 28], a popular agile framework. Scrum defines a concise system of roles, artifacts, and ceremonies that enables development teams to work under uncertain requirements [25], to respond to change rather than trying to eliminate it [15], and to continually improve their ways of working. As a framework Scrum provides a structure for teams to work, negotiate, learn, and self-organize.

Following the principles of the Agile Manifesto [1], Scrum emphasizes the frequent delivery of working software increments in collaboration with customers. Agile development avoids the bureaucracy of extensive up-front planning, design, and documentation in favor of continuous, interactive, and empirical improvement. Agile approaches assume that teams of motivated, skilled people in a supportive environment can be trusted to get a job done [6]. Some amount of management is nonetheless required to keep projects on track [4].

Scrum defines three distinct roles. The *product owner* is responsible for requirements, the "what" of development. The cross-functional *development team* takes care of the "how," the technical work including analysis, architecture, implementation, and testing. The *scrum master* facilitates communication and collaboration. The product backlog as a key artifact tracks work to be done, prioritized by the *product owner*. Everyone meets at the beginning of each iteration (*sprint*) for work planning and at the end to review the results. Each sprint also includes a retrospective meeting for the *development team* to reflect on and improve its practices.

Scrum aims for an optimal balance between self-organization and management. The *product owner* steers the project, but only through the precisely defined interface of backlogs and Scrum meetings. Other stakeholders may interfere with the project only through the *product owner*. The developers as a team are expected to make their own decisions within the scope of technology and work practices. In exchange the *development team* is also fully responsible for delivering working software increments at the expected quality. The *scrum master* keeps this scheme working, preventing one role from overpowering the other.

Agile frameworks like Scrum have become a common way of managing development projects. Although the agile principles call for "continuous attention to technical excellence and good design" [1], there is a widely held suspicion that agile development tends to neglect security [2, 12]. On the one hand, traditional approaches to security assurance, like certification, are tailored to waterfall processes and unable to handle frequent change and lack of documentation. On the other hand, agile development seems indeed to pose challenges when it comes to non-functional requirements [26].

ANALYTIC FRAMEWORK: STABILITY AND CHANGE IN ORGANIZATIONAL ROUTINES

Implementing security practices in software-developing organizations is often a matter of changing an established organizational setting as it was the challenge for the internal security initiative at our field site. We account for this situation by choosing an analytic lens enabling us to focus in particular on processes of change and stability in organizations.

HCI researchers repeatedly have investigated how organizations change and how organizational change can be facilitated. Of particular interest has been social computing as a means of intervention [23, 32] to effect organizational change. These studies highlight that change can emerge from the interaction of human actors and organizational structure. This view contrasts with functionalist understandings of organizational structures as external, independent forces on human behavior [5]. Instead, organizations are described as incorporating a duality of structure and situated practice, both co-constituted in an ongoing structuration process [11]. Structure and practice cannot be described separately, but have to be understood in their complex relation, enabling and restricting each other. This structuration perspective has not yet been employed to understand how organizations can change towards integrating security work, although there is an increasing HCI interest in security practices.

Studies on security tool design and adoption emphasize the role of organizational factors for tool success but do not investigate how practices and organizational structure interact [36, 37]. Software engineering studies, focusing on related topics like software quality [19, 31] or software project success [20], provide detailed insights into how organizational structure affects practices, but do not account for organizational structure as being *both* a premise and a product of human action.

Software development follows repetitive patterns of actions, which we conceptualize as a concatenation and alignment of multiple organizational routines. For instance, Scrum defines a development procedure with short iterations that becomes a routine for an organization working with Scrum. To explain change in organizational routines from a structuration point of view, Feldman and Pentland take on organizations as constituted through routines that have ostensive and performative aspects [9], hence reflect a structure-agency duality [18].

For example, a software development company might have a rule to regularly test software code. This rule, included in a development handbook, is the ostensive aspect of a software testing routine. It guides developers' actions and enables them to account for what they do. However, a rule does not explicate in detail how developers actually have to execute the tests, and it is not intended to. Each enactment of the routine is slightly different, taking into account concrete situations. For example, database code may require different tests than user interface code. This is the performative aspect of the

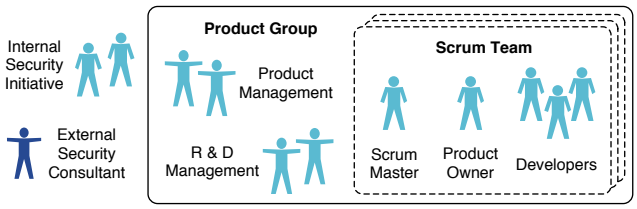

Figure 1. Roles in and around the Bizzview product group.

testing routine. Performances, although variable, are aligned to ostensive aspects. They can even lead to changes in ostensive aspects over time. For instance, if a code test tool turns out as useful to achieve the work, employing this tool can become a common expectation and the company might amend its development handbooks respectively. Feldman and Pentland pointed out that this way, routines can become a source of change if both ostensive and performative aspects of the routine are allowed to alter and do so in interconnected ways. This view departs from the conventional perspective of organizational routines as creating inertia in organizations.

We transferred Feldman and Pentland's approach to our case, as we needed to conceptualize the whole processes of an entire product group to analyze the reciprocity between human actions taking place after the consultancy, and pertinent organizational structures like self-organization, labor division, and agile development. We explored this entanglement to understand how the consultancy impacted the group and to make sense of our results. However, ostensive aspects are particularly difficult to describe and analyze. Howard-Grenville's refined model of flexible yet persistent routines supported our understanding of ostensive aspects as being visible in social expectations and material artifacts that inform actors on performing a routine for a situation at hand [17].

INVESTIGATED CASE: GLOBESOFT CORP.

Our field site is Globesoft Corp., a multinational software-developing enterprise situated in Europe, North America, and Asia. Like for all other names of products and people we use a pseudonym to protect the interests of our sources. Globesoft has more than 3'000 employees and an annual revenue of more than one billion US dollars.

Globesoft markets multiple business lines of enterprise software products. Although Globesoft makes off-the-shelf products, it is customer-oriented to the point of allowing urgent customer requests to interfere with planned development work. About ten years ago Globesoft introduced agile development methodologies across the company in an effort to increase software quality. Before the introduction of agile development methods, Globesoft used multi-year release cycles that relied on comprehensive, tightly structured top-down plans. Integration testing took place only at the end of a cycle; integration errors arose often and were expensive to fix.

Most agile teams at Globesoft use Scrum but some work with Kanban. Globesoft provides to its teams a centrally managed development infrastructure, such as a defect tracker, source code version control systems, and automated build and test systems. Together with agile methodologies, Globesoft has also adopted the idea of continuous improvement.

Our study investigated one of Globesoft's product groups developing Bizzview, a web dashboard for business data visualization that was part of a business analytics product line. At the time of our study, the product group comprised five scrum teams with 37 developers. The developers were distributed among five premises in Europe and North America; additional staff in Asia supported software testing.

An R&D manager (I13) involved in daily operations and a group lead (I6), who was ultimately responsible for Bizzview as a product, steered the product group. Each scrum team was coordinated by a *scrum master* and had a *product owner*, see Figure 1. Complementing the R&D management, the Bizzview product management was responsible for product strategy, customers, and sales. R&D management and product management had no hierarchical relationship. Setting up and maintaining a product strategy for Bizzview required R&D managers and product managers to collaborate.

Globesoft continuously added new functionality to Bizzview, e. g., business analytics functions currently demanded by the market. Also, Bizzview was an integral component of other Globesoft products. Due to Bizzview's growing importance for the firm's business, management decided to execute a thorough security audit. This audit was performed by an external consultancy company that had repeatedly been contracted by Globesoft for such work.

Security audits were part of Globesoft's internal security initiative. Globesoft perceived an increasing need to care about product security and to answer customer inquiries. Three years before this security audit, Globesoft had hence established a small centralized development security team. This team operated automated testing tools, handled incidents concerning development, gathered and reported security defect metrics, and offered awareness training and guidelines for developers. The security team did, however, not have sufficient resources to provide substantial support to individual development teams. Globesoft's security initiative remained low-key at the time of our study as the company was neither pushing security as a marketing claim nor had it experienced major security disasters in the past.

For two months the security consultants evaluated the Bizzview source code and tested running instances of the software against potential attacks. The consultancy firm handed over the audit results to the product group by submitting an audit report and by transferring individual findings into an internal issue tracker system. In addition, two months after the audit, one of the consultants performed a three-day security training workshop for Bizzview's developers and R&D managers.

METHODOLOGY

The chosen analytical framework suggests to analyze both situated practice and structural aspects for understanding organizational routines as a potential source of change. Taking on this view we investigated practices of the Bizzview product group members and their structural embedment. We planned

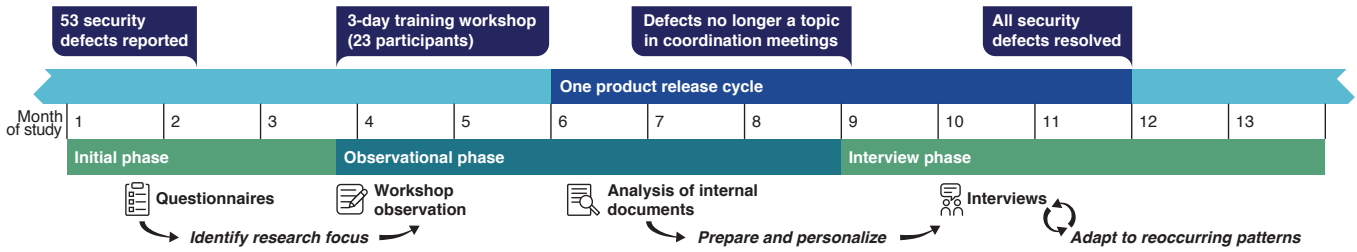

Figure 2. Activities of the product group (top) and the researchers (bottom) in the 13-month study.

our study over a relatively long period of time so that dynamics and possible change could become visible. The various methods described in the following gave us different insights into routines, their ostensive and performative aspects, and people's attempts to change them.

Our study took place over 13 months starting shortly after the security audit began. It comprised three phases: an initial phase aimed at getting a general overview of the team and the product, a second observational phase, and a third and final interview phase, see Figure 2.

During the initial phase we executed two anonymous questionnaire surveys, the first before the consultancy company handed over the audit results and a second two weeks before the training workshop. The questionnaires contained 17 and 25 questions respectively with free text fields and Likert-scale items as answering options. 15 out of 37 group members responded to our first questionnaire and 12 to our second questionnaire (response rates 42 % and 33 %).

The first questionnaire asked 17 questions in four topics. We asked the participants about their individual experience as software developers and in software security. Questions focused on security awareness, practices and attitudes in their development team, and their expectations regarding the audit results and the consequences of the consultancy. For example, we asked the participants whether their team discussed security matters on a regular basis, and whether security guidelines existed and were followed.

With the second questionnaire (22 items in five topics) we aimed to understand participants' immediate reaction, individually and as a team, to the audit results. We also inquired about the expected impact of the audit on the product group and expectations for the upcoming workshop. The questionnaire enquired, for example, whether the number and severity of audit results met participants' expectations and what the team had already done to address vulnerabilities. The questionnaire also asked whether fixing the reported vulnerabilities would remain a one-time security push and whether the workshop might help the developers to address the findings. Using the questionnaire data we identified first aspects of the group's organizational routines that seemed to be relevant for group members to engage in security work.

The next phase started with our observation of the training workshop. Two researchers participated in the training and took notes on every part of the event. Special attention was paid to two distinct areas: first, the consultant's actions with a particular focus on his didactic concept, conveyed security and risk models, and references to daily work procedures; and second, the participants in their reception of the conveyed knowledge, their attitudes towards the workshop, interaction with the consultant and with each other. Our aim was to gain insights into developers' engagement with security, and to observe how the consultant enacted his role as an acknowledged and experienced security expert, e. g. whether he focused attention onto himself or preferred to foster team discussions. Each researcher created an individual field diary recording observed sequences of interactions, impressions of participants' actions, and the workshop setting.

After the workshop, for a period of four months, two researchers reviewed material artifacts that informed group members on structure and current or past performances of development routines: the content of an internal task management system, a project wiki, and the internal defect tracking tool. Our goal was to reconstruct the product groups' activities and practices as well as how these were organized for the time shortly after the workshop. We specifically looked for references to security in the collected documents. We also created new representations of the collected data, e. g., timeline visualizations of the group's handling of the reported security defects, tables showing how issue fixing work was assigned, and a catalog of security- and audit-related text passages from the internal wiki and the task management system.

During the final phase, we executed semi-structured one-on-one interviews with 14 product group members and the external security consultant. We included participants from different scrum teams, work areas, and with different roles, see Table 1. The interview process took five months and was conducted by visits to five premises in Europe and video calls with US participants.

The focus of our interviews shifted over time as certain patterns were starting to repeat while new topics emerged. Initially, we focused on getting an overview of how software development was organized for Bizzview, what happened in the development teams after the training workshop, and how participants assessed the workshop. We also asked security-specific questions: how security was understood and communicated across the product group, what priority it had, and how resources were assigned for work on security topics. After discovering that decision-makers' actions had a noticeable effect on how security work was considered for orga-

ID	Role	Assignment	Location	Special interview focus
I1	Developer, software architect	Swordfish team	US	His alleged security expert role, product group's security history
I2	Developer, *scrum master*	Swordfish team	US	His *scrum master* role, coordination of defect fixing
I3	Developer	MightyNerds team	EUR 4	His experience with security as senior developer, defect fixing process
I4	Developer	Swordfish team	US	Bizzview's security history, defect fixing process, product management
I5	Security consultant	External company	EUR 1	Consultancy concept for Bizzview and Globesoft in general
I6	Product group lead	R&D management	EUR 3	Labor division, group members' roles, defect fixing process, security risks
I7	*Product owner*	MightyNerds team	EUR 2	His *product owner* role, security audit, scrum team self-management
I8	Quality auditor	Quality management	EUR 3	Defect fixing process, security as a quality requirement
I9	Developer, *product owner*	Swordfish & Inno team	US	Product group history, cross-product security, defect fixing process
I10	Developer, *scrum master*	Dashhovers team	EUR 5	Team-internal security initiatives, defect fixing process
I11	Developer	Dashhovers team	EUR 5	Team-internal security initiatives, defect fixing process
I12	Developer	Dashhovers team	EUR 3	His security expert status, defect fixing process, security in product management
I13	Bizzview development manager	R&D management	EUR 3	His role of managing the group's operations, security in product management
I14	Director product line	R&D management	EUR 4	Security as key performance indicator, security in product management
I15	Product manager	Product management	US	Security in sales and product management

Table 1. Interviewed developers and managers in varying roles and working at different premises (US - United States, EUR - Europe, numbers distinguish sites). Interviews followed a general structure, but also had a specific focus. Scrum teams had individual names chosen by their members.

nizational routines, we refocused to include aspects of how developers and managers interact and organize. We also acquired the product manager for Bizzview and the superordinated product line manager as interviewees. In preparing the interviews we extensively used the document analysis data and we also presented interviewees with our own data visualizations and discussed wiki and project management artifacts.

Overall we collected, transcribed, and analyzed 14 hours of interview recordings, and more than 100 pages of field diaries from the three-day workshop observation. Our four-month document analysis comprised more than 100 pages of protocols of defect fixing work from the internal defect tracker, internal protocols of weekly coordination meetings, Scrum retrospectives and R&D management meetings, as well as documentation of software development requirements, wiki content and data from internal collaboration tools.

Questionnaire data analysis aimed on finding general trends. Workshop field diaries were analyzed by their authors using an open coding scheme. Three major code categories emerged: "participants" for interactions of participants, "topics" for the relation between topics and interactions, and "meta aspects" for theoretically informed observations.

Like the field diaries, interviews were also coded by starting with an open coding approach. Other than the diaries, every interview was coded independently by two researchers, and category development was guided by our theoretical framing. We looked for actions that had been taken by multiple group members, required collaboration and were part of regularly occurring activities. To describe more general patterns of routines, we sorted them into two categories: *stabilizing the product* and *feature development.* We then looked for ostensive and performative aspects and analyzed how they corresponded to development artifacts we found during the document analysis. Reports for each project phase were reviewed and discussed among all researchers involved in the project.

RESULTS

Before we met the product group members in person, the results of our questionnaires provided us a first impression. To check whether our anonymous sample was appropriate to represent the product group, we asked participants to describe their work in a few words: Answers were diverse, describing different kinds of development tasks and technology expertise. This gave us confidence that most areas of the product group had been covered.

The group comprised experienced software developers; the average respondent had 10.1 years of development experience ($N = 15$, $\sigma \approx 8.0$) and roughly half of the respondents assessed themselves as software development experts, the other half as "experienced." In contrast, 12 respondents assessed their experience with security as "advanced beginners," and only one stated to be "experienced." We assumed that participants had internalized software development routines but lacked security knowledge. The second survey confirmed this trend. Eight out of 12 participants agreed or strongly agreed with the statement that their respective team would struggle to deal with the audit results without the workshop ($N = 12$, 5-point Likert scale, 2 neutral, 2 disagreed). Hence, we expected that the consultancy would bring up a new topic that had not been a focus of the development teams before.

Respondents' assessments of current security activities before the consultancy varied. For instance, five respondents agreed with the statement that their respective team often postponed security work as long as possible, three gave a neutral opinion, six disagreed, one even strongly disagreed ($N = 15$). This result together with responses to questions about former trainings, use of security tools and guidelines, and management support for security indicated that no coherent way to deal with security matters existed across the product group as a whole. Nonetheless some teams or individual developers might have worked on security issues before, while others disregarded security.

Regarding the developers' expectations of the consultancy, we found that 10 out of 15 respondents agreed or strongly agreed to the statement that it should serve as an impulse for the team to continuously work on the security of Bizzview. Moreover, nine respondents agreed or strongly agreed that they expected the consultants to find serious security defects

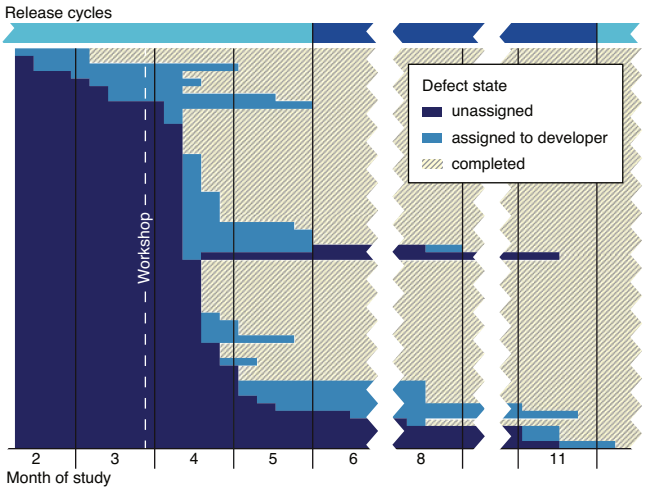

Figure 3. Status time line of security defects ordered by date of the first assignment to a developer. Time frame starts in the 2nd study month. Each horizontal bar shows the progress on one single defect.

in Bizzview. It followed that respondents were aware of shortcomings in Bizzview's security and also mostly felt that some kind of change was required. We concluded that the consultancy was conducted in a situation that favored its success. Throughout our study we did not find anything that contradicted these initial assumptions about the group.

Stabilizing the Product

Our questionnaire results suggested that security was not a topic addressed by the product group's routines. We wondered how they would deal with the high number of security vulnerabilities found by the consultancy firm.

Participants named fixing activities as the immediate reaction to the training workshop and the security audit results. The security consultant reported 53 security vulnerabilities in Bizzview six weeks before the workshop. For each finding he created a separate defect report. Creating defect reports triggered a distinct organizational routine in the product group: removing defects from the software code and thereby stabilizing the product. From our interviews we identified the following main procedures to be constituting this routine: First, defect reports were distributed to teams, or teams themselves picked up defect reports. Second, reports were assigned to individual developers within the teams. Reports at this point changed their state in the defect tracking system indicating that someone was working on a solution. Third, developers resolved defects, and defect solutions were tested for whether they broke other software functions. Finally, defect reports were marked as "completed" and archived.

Right before the workshop we observed that only two of the 53 reports had been marked as "completed"; only five of the others had already been assigned to developers. The number of completed reports suddenly increased to 38 only one month after the workshop and to 45 after another month, see Figure 3. We understood these patterns as performances of the stabilization routine. Apparently, performances directed towards security immediately followed the workshop. This raised the question of what the particular contribution of the workshop to the product group´s routines had been.

The workshop triggered engagement with security

When we asked participants for their thoughts on the workshop, everyone said they had enjoyed it and commended its quality and the consultant's commitment. In short, participants considered the workshop to be a good investment (I7, I4). We compared their statements to our own observations of the workshop: We also had experienced the appearance of the security consultant as eloquent, persuasive, and omnipresent. He convinced participants with his security expertise and always sought to get his audience involved. The knowledge gap concerning security topics between the consultant and workshop participants gave weight to his expert status.

Overall, we had witnessed the workshop as a productive event where developers focused on the workshop topics and the exercises. We concluded that the workshop could result in an improved understanding of software security as it conveyed to the developers means and methods to develop secure software. However, it attracted our attention that the workshop covered only individual developers' practices. The consultant himself explained that he had not been contracted to change the processes in the teams. But this was obviously a topic for the developers, as we observed how discussions emerged during the workshop concerning collaboration and coordination in development teams. However, the consultant did not follow up on them.

Despite its limited scope the workshop triggered a very positive attitude towards security and a desire to talk about it. Participants became confident that the product security of Bizzview required attention and that they were prepared to tackle the found defects. One of the interviewed *scrum masters* explained that the audit results had gained relevance through the workshop and that without it the reported vulnerabilities would *"probably not"* (I10) even have caught his attention. Another one explained that he would have needed longer to fix the issues without the workshop (I3).

Participants were so much motivated they even used the workshop to set up an ad-hoc meeting for discussing the defect reports. The director of the product group remembered it as follows: *"[...] Sure, there was euphoria because of the training: We have these security holes – let's tackle them." (I6)*. Clearly, the workshop and the security audit created a new situation for the team because they had never engaged in security practices at this scale before. However, through the workshop they felt empowered to fix the defects. For us the question remained whether this first-time encounter would lead to a change in their routines over time.

Attention faded as soon as the defect count dropped

The internal documents showed that for two months after the workshop the defect reports from the consultant were a regular topic in the weekly coordination meetings between the R&D managers and the key representatives of all scrum teams. Meeting minutes indicated that two teams worked on solving the defects. Progress was carefully tracked each

week. Two months after the workshop, however, tracking stopped and the remaining open defect reports were only occasionally mentioned in further meetings with respect to challenges for code integration. Our interviewees confirmed that security had initially been a topic, but was soon omitted. The R&D lead reported: *"Security aspects are so far no special topic [at the coordination meetings], moreover it was one among many other work packages." (I6)*

While the coordination of fixing was not driven by security considerations, it was strongly affected by the product line's internal politics: A high number of open security defects contributed to the overall number of open defects for a product and was perceived as a drop in product quality. It was only because of this perceived drop in quality that the R&D managers monitored the number of security defects. Along these lines, the Bizzview R&D manager described the fixing process as a matter of *"getting the counter down"* intended to reestablish a *"good standing"* for Bizzview. He explained:

"We argued that we already solved 56 of the 60 defects – or 45 of the 60 – whatever. For the internal discussions that means: The pressure [on the group] is off. And then we said: Okay, we still have six defect reports open but that's not so bad because we can provide reasons. [...] We accepted that we still have reports open because we also have other things to do. We had a good standing internally, [...] ninety percent done, the remaining ones we can leave open for some time." (I13)

Once the "counter went down" quality was back in its boundaries and hence security faded as an interest in the coordination meetings.

Defect fixing as a self-contained exercise

To fix the identified security defects, developers and managers performed the stabilization routine on the basis of established social expectations: Continuously maintaining the quality of Bizzview was an important goal for the product group, and if defect reports showed a lack of quality, the group had to react quickly to re-establish it. However, we found that the focus on immediate quality assurance measures hampered reflection. Performing the stabilization routine did not cause developers to re-consider how prevalent development practices had supported the introduction of software defects in the first place. Because developers' actions were neither aimed at producing additional insights into how defects were caused nor at facilitating exchange on these lessons, necessary knowledge was missing for further enactments of development routines that could have helped to prevent similar problems in the future.

For example, because the high number of defect reports required a quick response, the reports were assigned to individual developers pragmatically according to who had knowledge of the affected product components and taking into account individual developers' workloads. Security expertise of developers that could be potentially involved in the process was not a decisive criterion. As a case in point, one interviewee (I12) reported that he was required to support a colleague who had been assigned to fix one of the defects although this colleague had not attended the workshop and lacked required knowledge. On the other hand, the interviewee himself was not asked to fix any of the identified vulnerabilities even though he had attended the workshop.

Also, involved developers did not review the actual security risks of defects. Instead of trying to develop an understanding of the security risks for their product, developers employed an already established understanding of risks they had successfully applied in the past: A "risky" defect was difficult to fix without breaking other functions of the product. One *scrum master* who coordinated fixing of risky defects described:

"The smaller things that we know that are not going to be of much risk, we can put directly under this trunk of code line, fix it right there and go straight into the product. For something bigger that is going to affect multiple components, we create sort of a copy of the trunk code line and it gets all the updates that all the developers are doing." (I2)

Another reason for hampered reflection on the role of security for development work was that little knowledge exchange took place between developers who had been involved in the fixing process and those who had not. This was a concern for many interviewees. For example, developers not involved critically remarked that nobody informed them about the resolutions. Moreover, during our analysis of content from the group's wiki we did not find any documents related to the fixing of the security defects although we observed developers extensively using the wiki to exchange information on other matters. One developer, who had solved some of the security issues, criticized: *"Just a lot of stuff was coming on our plate that we needed to get done and there was no real consideration on how we avoid this in the future." (I4)*

In summary, performances of the stabilization routines did take the defect reports into account and work was strongly motivated by the training workshop. However, we did not observe any impact of the security defect fixing on further performances of organizational routines or even changes in ostensive aspects; participants were concerned about this. As the product lead put it: *"The only formalized reaction [to the consulting] was that we said we would fix the defects." (I6)*

Feature Development

The product group performed the stabilization routine to correct software defects introduced while new software functions had been added to Bizzview. Creating new functions was the purpose of another organizational routine: the feature development routine. It required a strong agreement between different actors in the company. As features shaped how Bizzview as a product was perceived by customers, people directly involved in its business activities needed to align with technologists and their managers. To balance the creation of new features with the fixing of past mistakes, phases of feature development and product stabilization alternated during the course of a six-month release cycle.

To align business requirements with development, product managers that were in contact with customers and R&D managers collaboratively created feature requests. These documentation artifacts manifested their understanding of how

business goals and the sales strategy for Bizzview should be mapped to functional requirements. This mapping is visible in the text structure of the artifacts that had two respective sections: a description of required software functionality and an outline of its expected business value.

Once a request was created, implementation was coordinated between R&D managers and the scrum teams. Costs for implementation were estimated, requests were prioritized, assigned to scrum teams, and scheduled for implementation. Further procedures took place solely within scrum teams. Team members divided a request into tasks, distributed work, analyzed possible technical solutions, created the necessary software changes, and documented and tested the results. Eventually, the software adaptations would be scheduled for integration into an upcoming software release and handed back to managers as a new feature.

Security conceptualized as a quality attribute

During our interviews and the document analysis we investigated whether and how developers took security into account for feature development after the consultancy. We found that their managers considered security as one quality aspect among others. As such they expected that security would be added to every requested feature without further need to explicate this requirement. This expectation not only existed for security but also for any other implicit quality requirement, like usability or software maintainability. The product line director explained:

"[Security] plays the same role like all the other '-ilities,' how we call them. That means: usability, security, performance, scalability. [...] Every time we move forward on the feature side we should not deteriorate the '-ilities.' Security might play a prominent role here, but is only one among other things." (I14)

We found that the developers shared their managers' view that security is ultimately a quality issue. The *scrum master* of one development team outlined that security is a required quality every new feature should possess: *"[...] anything that we add into the product is a feature that is gonna have to have security baked into it." (I2).* Our participants pointed out that as a quality attribute the developers also did not expect security to become visible and explicit in interactions with managers.

Developers as self-organizing "technology shepherds"

We saw that the development teams, as the Scrum framework suggests, were self-organizing units enjoying some degree of autonomy in Globesoft. In particular, teams could develop own procedures of how to implement the functional requirements explicated in feature requests. This autonomy was beneficial for both managers and developers: Developers could take work on the technology mostly into their own hands, and managers were not burdened with dealing with technical details of the software.

However, every agreement between managers and developers needed to take this autonomy into account. This constraint limited the managers' room to maneuver: They could not simply demand security work by prescribing security activities as a part of a feature request. The director explained:

"There exists no rule book saying 'for finishing this feature please spend two hours on security' [...] The idea is to set up teams to be self-learning so that they consider it in the process from the very beginning, kind of trying to channel the '-ilities.' " (I14)

As the director suggested, managers could steer security as quality only indirectly, for example, through monitoring measurable quality indicators, which was considered difficult. Or managers could raise developers' awareness with training measures like the workshop. As a third possibility, one interviewee mentioned that balancing developers' expertise and personal traits when setting up new teams was a means to influence the maturity of teams (I7).

The self-organization of teams and its implications on the ways how management and developers could interact stood out as an ostensive aspect of the feature development routine in the Bizzview group. We wondered how the security audit and the workshop continued to have effects on feature development practices given this specific organizational framing.

Security awareness increased after the consultancy

We found that developers intensely disliked managers' interventions when they touched the scope of their self-organization. One *scrum master* stressed this point when he talked about whether management interventions could be helpful to encourage security activities in his team:

"Actually I don't want that [strict guidelines] ... I don't wanna say it is necessary that someone from the top starts asking us to do certain things." (I10)

But developers also clearly stated that it was the responsibility of each developer in every team to keep Bizzview secure. One developer summarized: *"I would say, because we are working Scrum-like, every team should take up these questions." (I12)* Developers expressed that it was part of their self-conception as professional technicians to deliver good quality, sound, and also secure software features. Along these lines, developers understood the security consultancy not as an intervention by the management but rather as a chance to improve. The consultancy was an *"eye-opener"* (I6), as one participant told us, that fostered awareness among developers that security topics were an issue they needed to look after in their daily work. We observed that a dedicated "awareness part" was a salient training topic where the consultant explained potential security threats to the developers; we link the awareness increase in particular to this part.

Developers observed the improved awareness in their interactions with colleagues. After the consultancy, it was easier to attract attention for a security concern on the management level but also within the teams. One of the *product owners* summarized this perceived change:

"I mean that was the key that was missing. So everybody thought that they knew about it, everybody kind of was aware of it in the background, but nobody ever connected the dots and said, ok, I am building x, y and z, and I have to be careful

about what are the possibilities or what are the loopholes ... That sort of changed. [...] You can talk to people even in the hallway or talk to – meetings, and saying, oh, I am fixing CSRF issues and I would say maybe six months ago nobody would even know what the hell that meant." (I9)

It appeared as if developers were in a state of watchfulness for security problems after the consultancy. This was also the view of R&D managers who expected that developers were now able to identify security problems when they arose. But when we asked developers for concrete actions to take security issues into account, they reported only the one-time performance of the stabilization routine to fix the found security defects, very few if any concrete changes to their feature development practices, and no changes in the overall structure and enactments of their development routines.

Developers took the initiative, and failed

When we recognized that developers became watchful and aware about security but did not adapt their practices, we directed our attention to possible reasons for this observation. We focused on a finding from our document analysis where we observed that one scrum team, named the "Dashhovers," did in fact add new security items to their feature delivery check lists they used to organize feature development work. For instance, one item added requested developers to *"[t]hink of possible security issues caused by the new chart type and try to do something evil"*.

When we talked to the "Dashhovers" developers, they reported that it was solely the responsibility of the individual developers how thoroughly this check was actually going to be attended to. One developer expressed concerns that team members performed these checks only superficially:

"This [check] gets lost in the daily work since we always have time pressure, the release needs to be finished, tests need to be done ... One never arrives at the point to work on it in detail where we would be able to find the really bad things." (I12)

Taking into account further interview statements from "Dashhovers" developers we concluded that the team's approach did not work out well. But, turning our attention to the other scrum teams, we found developers using variations of the "being lost in daily work" argument when explaining why they themselves did not engage in concrete security activities after the consultancy. In addition to that, developers often simply pointed to a lack of resources as a reason.

We got the impression that developers nonetheless had spent time thinking about potential improvements of their development practices that could work for their teams, because they often spontaneously told us about their respective ideas during the interviews. For example, one interviewee suggested regular internal hacking challenges to update developers' security knowledge. Others proposed to run annual security retreats to discuss upcoming security topics with the teams, or to set up a Bizzview security task force, or, at least, a security go-to person to build up security expertise and provide assistance if security questions arose. Another idea was to introduce special development iterations (*sprints*) focused on security. But in difference to the attempt of the "Dashhovers" team, these ideas did not even reach the stage of creating the necessary preconditions to try them out.

Decision-makers' view of security

Developers' explanations that their practices did not change because of a perceived lack of resources and a conflict with other activities raised the question how security was prioritized against other development goals. As one answer, several interviewees pointed to management staff and argued that managers would see security as being in a resource conflict with feature development. Bizzview's R&D manager, responsible for the product group's operations, addressed this problem when he reflected on the developers' viewpoint:

"... what the developers are saying, we actually need to have more time [for security], is exactly the same I'm trying to explain: That would be a [higher] management decision – we are building fewer features and focus on something else. From my perspective this is currently not considered." (I6)

However, we also asked managers involved in the strategic decision-making for Bizzview about their considerations of security. We found that although security had not yet come to their specific attention, they did not explicitly neglect security as a requirement for Bizzview: The product manager for Bizzview, partially responsible for creating the feature requests, considered security a matter of quality just like the R&D managers and developers did. She trusted the development teams to deal with security as a technical issue. Hence, she did not pay particular attention to it (I15). The product line director explained that he was generally in favor of supporting security activities of the Bizzview developers. He stressed that he was the one who initially had suggested the security consultancy for Bizzview. He also offered support for introducing security tools in the development processes if there was a plan how this could be done (I14).

We concluded that there was no general trend of decision-makers opposing security activities that could explain developers' perception of a lack of resources and priorities for security. Instead we saw the reason for developers' inability to establish security work in the interaction of both their organizational structure and the perception of security as not being a feature but a matter of quality. We describe this interaction in the following.

Security was intangible and of little perceived value

The self-organization of teams required strong agreements between managers and developers about expected deliveries of the teams; feature requests were manifestations of these agreements. We realized that the obligations imposed by feature requests as ostensive aspects of the feature development routine had the consequence that developers could only account for their actions if they served these agreements. Since none of our interviewees considered security a feature for Bizzview, it could not become visible in feature requests. We wondered whether developers' orientation towards serving feature requests had an effect on how they themselves valued the results of security work. One interviewed *product owner* explicitly addressed this problem:

"But if we only develop security features [...], the product manager has nothing [...] for the next sales training. [...] he has no shiny new features to show [...] no further checkbox to tick in a sales brochure. This is the mindset these folks are thinking in." (I7)

Similarly, another developer detailed his own thoughts about the effort he would be willing to spend on security activities. He asked *"[...] if security is not on the list [of features], then is it really worth the time and extra energy to do it?" (I4)*

However, we not only found that developers were concerned about security work lacking valuable outcomes for stakeholders outside the development teams, but developers also expressed that security as a quality requirement lacked visibility within the teams. One *scrum master* referred to this problem by describing security as not being *"omnipresent"* (I10), and concluded that engagement with security would likely decline in the long run after the workshop. He was already starting to observe this process: He reported of a *"hacker challenge"* event planned in his team that eventually did not take place. As a reason he suspected that a *"story line"* (I10) for security was missing that would have triggered engagement.

Concerns that developers did not have an intrinsic motivation to work on security were also brought up by another developer. As a reason he suspected that in difference to work for creating software features, the contributions of security work were not apparent to developers. He explained:

"I mean we are developers because we enjoy it, I don't think any software developer does it because they are just making a paycheck [...] what you really enjoy is putting something together and seeing it work. [...] Security is not one of those things for most people I think, but it does need to be emphasized and we do need to prevent something from happening [...]." (I4)

External triggers such as customer feedback on security issues would be able to generate visibility, as one developer explained, but this had so far not been the case: *"Apart from the findings from the workshop there was never any feedback from the customer [...] That [feedback] would definitely motivate us." (I12).* Security work in feature development at Globesoft might also have been triggered by actions of the internal security initiative.

Product group critical of internal security initiative

During the time of our study, Globesoft's security experts attempted to establish a formal method to elicit security requirements for products on the basis of a formalized documentation of a product's software architecture. The product group lead of Bizzview reported that when this approach was presented to the product groups, it immediately sparked resistance from managers and developers. He himself considered it a *"very theoretical approach"* and *"plain bureaucracy" (I6).* In particular, he doubted that the approach could prevent security issues like those the security audit had revealed; he remarked:

"[...] the security initiative had a couple ideas of what should be documented somehow ... well, we are supposed to document whether it [Bizzview] is a web application [...] which is obvious and will not help to somehow prevent any of these issues found by Richard [the consultant]."

He stressed that this method would undermine developers' knowledge, distract them and require considerable extra work. At the same time the approach would miss establishing security as a tangible development goal.

DISCUSSION

Our study aimed to identify *how the security consultancy affected the organizational routines of the development group.* We also paid attention to signs that the encounter with the security consultants could lead to endogenous change in the product group in the medium and long term. We found that the security consultancy had effects on the product group and led to some isolated adaptations of organizational routines, e. g., it motivated performing security defect fixing and encouraged one concrete attempt to improve team-internal rules for feature development. However, we did not find that security work became an integral part of the product group's organizational routines as it was not incorporated in their ostensive and performative aspects. Developers themselves considered this outcome insufficient and had expected more.

We conclude that security work did not attract more engagement for two major reasons: First, developers could not account for security work because the ostensive aspects of the feature development routine constrained how security work could create a tangible value for Globesoft's business. Second, group members' idea of security as an implicit quality goal did not transfer into modifications of the feature development routine because it was at odds with how developers understood their work and made sense of their actions. Looking at the overall situation of the company, we also found it remarkable that the activities of the internal security initiative were disconnected from the requirements of the organizational setting at Globesoft, hence could not foster change towards integrating security work either. Ultimately, security work remained a reactive, unguided activity mostly dependent on unsystematic actions of individual developers.

Security Work Discouraged by Organizational Structure

Feldman and Pentland stressed that the ostensive aspects of organizational routines enable people to account for what they do. Engaging in specific work requires people to make sense of their actions towards others by referring to a routine as a common idea that all involved stakeholders share [9]. We observed that developers could not account for security work in the product group's specific organizational setting, because security was not visible in binding agreements between the managers and the development teams.

Globesoft in general did not actively market security to its customers and no serious security incident had occurred so far. As a consequence, not only for Bizzview but also for other Globesoft products, the market value of security had never been explored, and potential outcomes of security work could not be linked to Globesoft's business interests. But as agreements between managers and development teams were

built on exactly such perceived links between business and development goals, agreements did not address security.

Exactly these agreements guided developers in how to enact the feature development routine, how to prioritize tasks and eventually which work to carry out. Feature request artifacts, which were visible to all group members and served to organize and monitor development activities, further emphasized the multiple capacities of the agreements. Moreover, because managers could use feature requests to evaluate and appraise developers' work results, these requests reproduced managerial control by explicating superiors' expectations to developers. Strong adherence to agreements and always trying to fulfill feature requests were hence in the very interest of developers: both served to maintain the management's trust that development teams were indeed working towards fulfilling business goals. Had they lacked confidence in the binding nature of agreements, managers would not have granted autonomy and room for self-organization to the scrum teams, because doing so unavoidably weakened managers' control in the first place.

It is not surprising that agreements and activities in a commercial company are directed towards business goals. Secondary goals lacking a visible connection to the business side may catch less attention despite the fact that they can be important, too. However, management staff at Globesoft was not aware of the relationship between organizational structure and developers' preferred orientation towards articulated links between business and development. Interestingly, this connection also appeared counter-intuitive to managers: Managers cherished developers' expertise and trusted them to tackle technical issues on their own. But managers did not realize that development teams spent much effort to preserve their autonomy by first and foremost serving the *explicit* agreements with their management. While performing the feature development routine mostly autonomously, the development team followed their managers' stated goals and priorities.

A response in the organizational setting described here would be to investigate the potential business value of security, thus making it a more tangible development goal. If Globesoft had taken this step, the embedding of security as a topic in the product group's routines would have been rather different: Requiring security from a business perspective would make security an explicit product goal in the agreements between managers and developers, thus enabling developers to account for doing security work.

Stable Routine Performances Despite Interventions

The explicit feature agreements as outstanding ostensive aspects of the feature development routine neither guided developers to engage in security work, nor did they support developers to justify spending any effort on security issues. While this finding clearly showed how Globesoft's organizational structure hampered integrating security work, it alone cannot account for the little effect the consultancy had on the product group's organizational routines.

Although certain practices, such as security work, might not be a firm part of an organizational routine visible in the routine's ostensive and performative aspects, this condition can change when people have to respond to new circumstances or situations when enacting a routine. Indeed, Feldman and Pentland have stressed that new requirements, external changes or reconsiderations of current practices can lead people to search new ways of doing things [9]. Resulting adaptation can bring up variations of routines; if variations turn out to be advantageous and become repeated patterns, they eventually transfer into the ostensive aspects of organizational routines.

The consultancy created such a new situation for the development team: While the development group had rarely dealt with security issues in the past, the security test results and the training workshop raised noticeable concerns among developers and managers that security required more of their attention. To address these concerns product group members concluded that security should be a mandatory quality of each software feature to develop, similar to other quality requirements such as usability or maintainability. Moreover, the training workshop built momentum that the security defects found by the consultants should be fixed quickly.

The consultancy was apparently a turning point for group members' views of security. But nonetheless further performances to deal with security issues either focused solely on fixing the already reported defects or did not take place at all. This response was particularly surprising given developers' strong motivation after the training and the fact that the applied agile development principles provided a flexible framework to manage and organize development work.

Before the security audit, Globesoft staff and the consultants had decided to convey its results as defect reports. This might had been an unconscious decision because it was a common way of dealing with security audit results. But in the first place it restricted the group's response to enacting the stabilization routine. Achieving a lasting effect on the product group's routines was hence difficult from the very beginning, because the stabilization routine was not oriented towards reflecting on how the product was created. Enacting the routine did not encourage group members to reconsider whether security was addressed appropriately in the feature development routine, instead it focused solely on short-term fixing actions.

Nevertheless the training brought up a common view among developers and managers that they needed to keep an eye on security issues; after the training, an insecure feature was perceived as being of bad quality. The moment this conception of security as a quality issue interacted with the labor division between managers and developers, the idea arose that security matters were in the hands of development teams and that ultimately every single developer was in charge of building *secure* features. The consultancy supported this idea by presenting security as a merely technical issue usually within the scope of developers and security shortcomings as mistakes caused by single, inattentive people.

But the view that after the consultancy developers take care of security when implementing features was merely a desire to change the feature development routine. When one of the

scrum masters explained that any added feature "is gonna have to have security baked into it," he actually advocated a change but did not describe the situation in the teams. Feldman pointed out that the problem with discussions about *desired changes* in an organizational structure is that articulated ideas to change a routine can fundamentally contradict people's general understanding of how work is actually achieved in their organizational setting [8]. In that case, the actions an advocated change requires from people are unlikely to happen. Performances of routines remain stable and hence routines as a whole may resist change.

We saw that development team members' understanding of their work, which motivated their actions, was bringing together their individual work achievements with those of colleagues in Bizzview as—in a technical sense—a functioning software and—from a business perspective—a marketable product; developers liked "putting things together and seeing it work," as one participant described his way of making sense of what he does. The embeddedness and visibility of development activities as well as the tangibility of development goals contributed to team members' understanding of developing Bizzview in a collaborative team effort with intelligible business and technological challenges.

The idea of security as a quality matter solely directed at each developer's individual work contradicted this understanding of developing software *together* and in an experienceable way. Security did not appear as a present development challenge but as a desired intangible technical property of the software. Developers struggled to anticipate how isolated, individual achievements of security work without perceivable goals could contribute to the big picture of the software. Security work also did not appear as a team effort where the interdependency of tasks encourages people to collaborate and commit themselves to achieve desired work results. This perspective was not a topic during the security training either. Developers by themselves pointed to these issues and concluded that security work would unlikely become an essential part of development routines, but rather an unsystematic, personal activity.

Security Initiative Incapable of Triggering Change

Security experts in software developing companies often advocate changes in organizational routines to better integrate security work, even to the point of directly opposing developers and managers' decisions. Globesoft's internal security initiative took on multiple tasks, e. g., teaching developers in security measures, organizing product security audits like the one for Bizzview, and providing security software tools to development teams. But the security experts driving the initiative also tried to intervene directly into teams by prescribing changes in their development routines.

Security efforts at Globesoft relied on the internal security initiative. This raises the question whether this initiative actually had the ability and intervention power to influence the organizational setting. Interestingly, the security experts were not yet aware of how Globesoft's organizational structure affected their efforts to promote security work. When Globesoft had established agile development structures in all product lines about ten years ago, Globesoft managers had made a trade-off between the advantages of self-organization in teams and the resulting limitations to their power to control work taking place in the development teams: Managers could no longer simply impose rules for how self-organizing teams should do their development work.

With the company's shift to agile development methods, the knowledge how to steer scrum teams with indirect methods became crucial for Globesoft's management. New management methods like introducing additional quality indicators or adapting team staffing now relied on creating new situations for development teams instead of concise upfront planning of a whole development process. To compensate for the limited control over team-internal rules or procedures, managers encouraged developers to deal with the new situations they themselves created, and this way aimed to change developers' organizational routines in the long run.

It became evident that the security experts lacked skills and knowledge of how to manage self-organizing scrum teams successfully, hence their ability to promote change in Globesoft was substantially limited. They had been promoted to their position primarily because of their outstanding security expertise but not because of managerial skills or experience. From this perspective, it was not surprising that the constraints limiting interventions in Globesoft's development teams had not yet caught their attention. In contrast, the interviewed managers clearly anticipated that the security experts' interventions would likely be of no avail.

The different distribution of management and security knowledge at Globesoft was unfavorable for improving security work: While the security experts had limited skills and knowledge how to manage change in the organization, the managers with the expertise to steer agile teams lacked security expertise to understand and articulate the actual security requirements to their products. Both Globesoft's security experts and the managers had not yet tried to collaborate on this issue and to combine their knowledge in order to find a response to security demands in their organization.

IMPLICATIONS

Our study suggests that the success of attempts to integrate security work into organizational routines hinges on how people conceive security in their specific organizational setting. Ideas of security can vary broadly: Security can play the role of a quality issue, a business value, a compliance obstacle, a technical challenge, a liability problem, a technical feature, or a sales pitch, to name but a few. Ideas about security guide decisions about which security activities to apply in an organization and how to integrate different types of security work into organizational routines.

However, any specific idea can hamper the performance of security work if it has implications on practices that contradict the way how work is structured and performed. Ideas might be particularly unfruitful if they neglect important organizational aspects such as the roles and responsibilities of people, prevailing methods of planning, entrenched ways of collaborating in software development, established managerial con-

trol practices, people's shared understanding of development procedures, and the ways in which people make sense of their work in the organization.

If the organizational structure cannot guide security activities or even impedes people to account for security work, and actual performances of security work are not motivated by developers' own ideas of security either, a change towards integrating security into development routines is unlikely to happen. Even in the face of a cue to action [36] to do security work, such as a security audit, development practices may remain unaffected if established organizational routines such as self-contained defect fixing procedures counteract it. Low-key, random engagements in security activities may be the result, just as our case showed.

Reconciling ideas of security with the requirements of specific organizational settings could be a task for security experts in organizations. Taking on this task would require security experts to possess not only technical security knowledge, but also to have insight into the management of software-developing organizations—in general and with particular knowledge of the situation in their organization. To effect change in software-developing organizations we suggest that security experts foster the communication of security as a requirement in a way that actually appeals to the organization itself. This does not necessarily require the experts to perform this communication work by themselves; it could also mean that they empower people in other roles to become security stakeholders for software development.

Agile development is commonplace and thus for many security initiatives determines the environment in which they seek to succeed. It may be surprising that the scrum teams in our study, while being self-organized, closely followed the management's priorities. A balance of autonomy and direction is, however, a key element of agile development, letting teams pursue goals of their organization while protecting them from micro-management and arbitrary interference. Security initiatives should embrace the structures that maintain this balance, paying equal attention to all roles involved and their respective concerns. Our results suggest that an agile approach to security comprises at least three components: (1) ways for *product owners* and product managers to make security an explicit and effective requirement and assess its fulfilment; (2) means to represent security requirements and security work in agile artifacts, such as backlogs; and (3) tools and activities for development teams to accomplish security work. Getting developers to adopt security practices may be the smallest problem once security has become a visible goal.

Security as an "-ility"—a secondary, cross-cutting and often implicit requirement—complicates the matter. We witnessed how our product group, perhaps with some help by the consultant, conceived security only as a non-functional quality attribute, thereby limiting their ability to work on it. Security requires analysis, design, implementation, and testing work, in part in conjunction with and dependent on feature development and in part as separate work concerning the product as a whole. Neither a pure quality perspective nor a pure feature view properly addresses security requirements. This makes security a challenge for any development process rather than a particular weakness of agile development frameworks. Security work may even benefit from agile development because of flexible routines and self-organization: Security concerns evolve with the product and agile practices create a framework for continuous analysis, adaptation, and improvement.

CONCLUSION

We studied a product group at a large software-developing company during and after their encounter with security consultants to find out how their organizational routines changed so as to integrate security work. Specific to our setting were the development of enterprise software products using Scrum and a security initiative still in the process of maturing. We found that, although the intensive consultancy had noticeable effects in the short term, sustainable change like the adoption of new tools or practices did not follow. We identified two reasons for this observation: conceptualizing security as an implicit quality issue in feature development and not a visible business goal; and adherence to the role concepts and interfaces of Scrum. Group members expressed concerns about the final outcome of the consultancy and suspected that security problems might re-occur in the future.

Our study suggests that security initiatives need to take organizational factors into account. Security initiatives should not only be considered from a security engineering point of view but also from a change management perspective, and aim at reconciling people's ideas of security with the requirements of the organizational setting. Organizations should focus on establishing suitable means to communicate security as a requirement in the organization. Making security as a requirement explicit supports steering agile development teams without curtailing their necessary self-organization.

ACKNOWLEDGMENTS

We would like to thank the members of the Bizzview product group for participating in our study, as well as the Head of Research of Globesoft, the staff of Globesoft's security initiative, and the R&D managers responsible for Bizzview for supporting our research and helping us to get necessary internal approvals. We also want to give recognition to Golriz Chehrazi who supported us during interview analysis, Rebekka Kugler for proofreading, and the anonymous CSCW reviewers for their thoughtful, constructive remarks. This research was partially funded by the European Center for Security and Privacy by Design (EC-SPRIDE).

REFERENCES

1. K. Beck et al. 2001. Manifesto for Agile Software Development. (2001). http://agilemanifesto.org/
2. K. Beznosov and P. Kruchten. 2004. Towards agile security assurance. In *Proc. NSPW '04*. ACM, 47–54.
3. M. Bishop. 2007. About Penetration Testing. *IEEE Security & Privacy* 5, 6 (Nov 2007), 84–87.
4. B. W. Boehm. 1991. Software risk management: principles and practices. *IEEE Software* 8, 1 (Jan 1991), 32–41.

5. S. R. Clegg, C. Hardy, and W. R. Nord. 1996. *Handbook of Organization Studies*. SAGE Publications.
6. A. Cockburn and J. Highsmith. 2001. Agile software development, the people factor. *Computer* 34, 11 (Nov 2001), 131–133.
7. G. Conti and J. Caroland. 2011. Embracing the Kobayashi Maru: Why You Should Teach Your Students to Cheat. *IEEE Security & Privacy* 9, 4 (July 2011), 48–51.
8. M. S. Feldman. 2003. A performative perspective on stability and change in organizational routines. *Industrial and Corporate Change* 12, 4 (2003), 727–752.
9. M. S. Feldman and B. T. Pentland. 2003. Reconceptualizing Organizational Routines as a Source of Flexibility and Change. *Administrative Science Quarterly* 48, 1 (2003), 94–118.
10. D. Geer and J. Harthorne. 2002. Penetration testing: a duet. In *Proc. ACSAC'02*. 185–195.
11. A. Giddens. 1984. *The constitution of society: Outline of the theory of structuration*. University of California Press.
12. K. M. Goertzel, T. Winograd, H. L. McKinley, L. J. Oh, M. Colon, T. McGibbon, E. Fedchak, and R. Vienneau. 2007. *Software Security Assurance: A State-of-the-Art Report*. Technical Report. IATAC & DACS.
13. M. G. Graff and K. R. van Wyk. 2003. *Secure Coding: Principles and Practices*. O'Reilly.
14. A. Guzzi, A. Bacchelli, Y. Riche, and A. van Deursen. 2015. Supporting Developers' Coordination in the IDE. In *Proc. CSCW '15*. ACM, 518–532.
15. J. Highsmith and A. Cockburn. 2001. Agile software development: the business of innovation. *Computer* 34, 9 (Sep 2001), 120–127.
16. M. Howard and S. Lipner. 2006. *The Security Development Lifecycle*. Microsoft Press.
17. J. A. Howard-Grenville. 2005. The Persistence of Flexible Organizational Routines: The Role of Agency and Organizational Context. *Organization Science* 16, 6 (2005), 618–636.
18. B. Latour. 2005. *Reassembling the Social - An Introduction to Actor-Network-Theory*. Oxford University Press.
19. M. Lavallée and P. N. Robillard. 2015. Why Good Developers Write Bad Code: An Observational Case Study of the Impacts of Organizational Factors on Software Quality. In *Proc. ICSE '15*. IEEE, 677–687.
20. H. Leung. 2001. Organizational factors for successful management of software development. *The Journal of Computer Information Systems* 42, 2 (2001), 26.
21. S. Matthiesen, P. Bjørn, and L. M. Petersen. 2014. "Figure out How to Code with the Hands of Others": Recognizing Cultural Blind Spots in Global Software Development. In *Proc. CSCW '14*. ACM, 1107–1119.
22. G. McGraw. 2006. *Software Security: Building Security In*. Addison-Wesley.
23. W. Orlikowski. 1992. The duality of technology: Rethinking the concept of technology in organizations. *Organization science* 3, 3 (1992), 398–427.
24. C. C. Palmer. 2001. Ethical hacking. *IBM Systems Journal* 40, 3 (2001), 769–780.
25. M. Poppendieck. 2002. Wicked projects. *Software Development Magazine* 10, 5 (May 2002), 72–76.
26. B. Ramesh, L. Cao, and R. Baskerville. 2010. Agile requirements engineering practices and challenges: an empirical study. *Information Systems J.* 20, 5 (2010), 449–480.
27. K. Schwaber and M. Beedle. 2002. *Agile Software Development with Scrum*. Prentice Hall.
28. K. Schwaber and J. Sutherland. 2013. The Scrum Guide. `http://scrumguides.org/`.
29. C. Severance. 2016. Bruce Schneier: The Security Mindset. *Computer* 49, 2 (Feb 2016), 7–8.
30. I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles. 2015. Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects. In *Proc. CSCW '15*. ACM, 1379–1392.
31. D. L. Stone and E. R. Eddy. 1996. A model of individual and organizational factors affecting quality-related outcomes. *J. of Quality Management* 1, 1 (1996), 21–48.
32. L. A. Suchman. 1987. *Plans and Situated Actions: The Problem of Human-machine Communication*. Cambridge University Press, New York, NY, USA.
33. H. H. Thompson. 2003. Why security testing is hard. *IEEE Security & Privacy* 1, 4 (july-aug. 2003), 83 – 86.
34. R. Werlinger, K. Hawkey, D. Botta, and K. Beznosov. 2009. Security practitioners in context: Their activities and interactions with other stakeholders within organizations. *Int. J. of Human-Computer Studies* 67, 7 (2009), 584 – 606.
35. L. Williams. 2012. What Agile Teams Think of Agile Principles. *Commun. ACM* 55, 4 (April 2012), 71–76.
36. S. Xiao, J. Witschey, and E. Murphy-Hill. 2014. Social Influences on Secure Development Tool Adoption: Why Security Tools Spread. In *Proc. CSCW '14*. ACM, 1095–1106.
37. J. Xie, H. Lipford, and B.-T. Chu. 2012. Evaluating Interactive Support for Secure Programming. In *Proc. CHI '12*. ACM, 2707–2716.