# Onboarding Software Developers and Teams in Three Globally Distributed Legacy Projects: A Multi-Case Study

Ricardo Britto[1*], Daniela S. Cruzes[2], Darja Smite[1,2] and Aivars Sablis[1]

[1]*Department of Software Engineering, Blekinge Institute of Technology, SE-371 79, Karlskrona, Sweden*
[2]*SINTEF Digital, NO-7465, Trondheim, Norway*

SUMMARY

Onboarding is the process of supporting new employees regarding their social and performance adjustment to their new job. Software companies have faced challenges with recruitment and onboarding of new team members and there is no study that investigates it in a holistic way. In this paper, we conducted a multi-case study to investigate the onboarding of software developers/teams, associated challenges, and areas for further improvement in three globally distributed legacy projects. We employed Bauer's model for onboarding to identify the current state of the onboarding strategies employed in each case. We learned that the employed strategies are semi-formalized. Besides, in projects with multiple sites, some functions are executed locally and the onboarding outcomes may be hard to control. We also learned that onboarding in legacy projects is especially challenging and that decisions to distribute such projects across multiple locations shall be approached carefully. In our cases, the challenges to learn legacy code were further amplified by the project scale and the distance to the original sources of knowledge. Finally, we identified practices that can be used by companies to increase the chances of being successful when onboarding software developers and teams in globally distributed legacy projects.

KEYWORDS: Onboarding; global software engineering; global software development; legacy.

## 1. INTRODUCTION

Day after day, companies must deal with different challenges to survive in an increasingly competitive market. For software companies, the source of competitive advantage has always been associated with the competent resources due to the knowledge-intensive nature of the work. Therefore, recruiting and onboarding new employees is one of the key areas of success. Onboarding (also known as organizational socialization) is the process of supporting new employees regarding their social and performance adjustment to their new job [1]. In the context of software development, many reasons may lead to onboarding of new developers such as: i) to replace retired developers; ii) to replace developers that left or will leave the company; iii) to scale up the number of developers in response to the growth of the number of customers; iv) to incorporate new people that may bring new ideas and thus help a company to innovate; or v) to take over the work from original developers (also a type of replacement) in order to free up the experienced developers for starting something new. In response, companies must recruit new people, or, in other words, start the "onboarding" process, by which newcomers make the transition from being organizational outsiders to being insiders [13]. Effective onboarding helps new employees learn attitudes, knowledge, skills, and behaviors required to work effectively [1–4].

[*]Correspondence to: Ricardo Britto, *Department of Software Engineering, Blekinge Institute of Technology, SE-371 79, Karlskrona, Sweden*. Email: rbr@bth.se

The onboarding process can be either formal or informal [5–7]. In an informal onboarding, new employees learn about their new job without an explicit organizational plan, while in a formal onboarding, they are assisted in their new organizations by means of a written set of coordinated policies, procedures, and actions. The levels of formality and comprehension of an onboarding program vary among companies. However, existing literature suggests that successful companies regard onboarding of new employees formally rather than ad-hoc [1].

While there is a vast literature about onboarding in a diverse type of professions [4,8–11], there are only a few dedicated studies in software engineering research and, to the best of our knowledge, there is no holistic study that investigates how the onboarding of software developers is strategized.

To fill the existing gap, we carried out an empirical investigation on how the onboarding of software developers is strategized in three different companies. Our case companies are globally distributed and onboard developers to work in ongoing software product development efforts. We argue that such environments introduce additional challenges for both newcomers and software companies, since onboarding in globally distributed environments may happen both locally and remotely. Furthermore, newcomers may have to deal with large amounts of accumulated legacy code, which was written by the original developers who may not be working in the company any longer. To understand the peculiarities and challenges of onboarding processes in the mentioned environment, we share our findings from an exploratory multi-case study that is driven by the following research question:

- What are the functions (practices, tools, techniques, methods, and technologies) employed by companies to onboard software developers in globally distributed legacy projects?

We analyzed the onboarding strategy employed by each company using a holistic model for successful onboarding proposed by Bauer [1]. The proposed model has been empirically tested and widely adopted and allows evaluating the range of onboarding activities employed by a company.

The main contributions of this paper are:

- A holistic analysis of the onboarding strategy employed to onboard software developers by three software companies (holistic, related to software engineering field, empirical).
- The use of an onboarding model to assist in the process of identifying areas of improvement in the onboarding process of software developers and software development teams (model applicability testing, theory-driven research).
- Recommendations that can be used by companies to improve their onboarding strategies with respect to local and remote developers.

The remainder of this paper is organized as follows: Section 2 describes the background and related work. Section 3 presents the research design, followed by the results in Sections 4. Section 5 presents a discussion of the results and implications for practice. Validity threats and limitations are discussed in Section 6. Finally, Section 7 contains the conclusions and our view on future work.

## 2. BACKGROUND AND RELATED WORK

In this section, we describe the concept of onboarding (organizational socialization), the model for successful onboarding by Bauer [1], and summarize related work that focuses on onboarding of software developers.

### 2.1 Onboarding

To improve the effectiveness of the talent management systems beyond effective recruitment and new employee selection, companies shall consider the strategic use of onboarding. Onboarding, also known as organizational socialization, refers to the mechanism through which new employees acquire the necessary knowledge, skills, and behaviors to become effective organizational members and insiders [12,13]. Research and conventional wisdom both suggest that employees get about 90 days to prove themselves in a new job [1].

Klein et al. [4] affirm that the research on onboarding can be divided into four distinct perspectives:

- **Stages** through which newcomers progress [14,15].
- **Actors** involved with the onboarding of newcomers [16,17].
- **Tactics and practices** employed by organizations for onboarding newcomers [1,13,18].

- **Content** to be learned by newcomers during the onboarding [15,19].

Considering that the main focus of this paper is on onboarding tactics and practices, we elaborate further on this perspective, describing the main models of onboarding: Van Maanen and Shein's model [13], Jones' model [20] and Bauer's model [1].

### 2.2. Van Maanen and Shein's Model

Van Maanen and Shein [13] proposed a theoretical explanation regarding role orientation in the context of onboarding. The model categorizes onboarding tactics in six dimensions:

- **Collective vs. individual** – Collective onboarding occurs when a group of newcomers go through onboarding activities and acquire experiences together (e.g., boot camps). Individual onboarding occurs when newcomers go through separate from other newcomers (e.g., apprenticeship).
- **Formal vs. informal** – Formal onboarding relates to tactics in which newcomers are segregated from other employees (e.g., policy academies). Informal onboarding relates to tactics that have no or little separation between newcomers and other employees (e.g., on-the-job training).
- **Sequential vs. random** – Sequential onboarding refers to the extent to which discrete steps regarding the onboarding phases are specified for the newcomers, while random onboarding tactics do not specify any sequence of steps.
- **Fixed vs. variable** – Fixed onboarding occurs when there is a timetable associated with each step of the onboarding process, so that a newcomer knows the exact time required to complete each step. Variable onboarding does not associate any time with the onboarding steps. Rather, newcomers receive some clues regarding when they should consider an onboarding step as concluded.
- **Serial vs. disjunctive** – Serial onboarding takes place when experienced employees serve as models for newcomers (e.g. a new police officer works for an extended period with some veteran police officer). Disjunctive onboarding refers to the tactics wherein no guidelines or models are provided to newcomers.
- **Investiture vs. divestiture** – Investiture onboarding occurs when an organization prefers that newcomers keep their personal characteristics and make use of their own skills, values, and attitudes. Divestiture takes place when an organization rejects and removes the personal characteristics of newcomers.

According to this model, the way newcomers respond to their roles differs due to the onboarding tactics used by organizations. This means that organizations can support newcomers by giving relevant information in different ways.

### 2.3. Jones' Model

Jones' model [20] was built upon Van Maanen and Shein's Model [13] and reduces the original six dimensions to two:

- **Institutionalized** onboarding occurs when tactics are implemented in structured programs and newcomers receive formal group orientation and mentoring. This dimension is composed by the following dimension categories of Van Maanen and Shein's Model: collective, formal, sequential, fixed, and serial investiture.
- **Individualized** onboarding takes place when newcomers start working from the beginning and must learn the norms, values, and expectations on-the-fly. This dimension is composed of the following dimension categories of Van Maanen and Shein's Model: individual, informal, random, variable, disjunctive, and divestiture.

Institutionalized onboarding is related to formal tactics, while individualized onboarding is related to informal tactics. Companies considered as successful regarding the onboarding of newcomers have more formal onboarding programs (institutionalized onboarding) [18,21,22].

### 2.4. Bauer’s Model

Bauer et al. conducted a series of studies that resulted in an empirically based onboarding model [1,3,10,12,21]. The model was conceptualized to support the design of onboarding programs, capitalizing on the fact that institutionalized onboarding is more successful than individualized onboarding [18,21,22]. While related to Van Maanen and Shein’s model and Jones’ model, Bauer’s model has a finer grain level than the previous models; it aggregates practices, techniques, methods and technologies (functions) that are related to successful onboarding (Figure 1).

The benefits of this model are two-fold: i) it facilitates the evaluation of current state of onboarding programs in real projects, supporting the identification of areas to improve; ii) it provides a set of good practices that can be used by organizations to improve their onboarding programs. Considering the main goal of this paper, Bauer’s model was considered the most adequate model and, thus, used in our investigation.

According to Bauer, onboarding has four distinct levels, known as the Four Cs, which are the building blocks of successful onboarding [1]:

- **Compliance** is related to teaching employees basic legal and policy-related rules and regulations.
- **Clarification** is related to ensuring that newcomers understand their new jobs and the related expectations.
- **Culture** is related to providing newcomers with a sense of organizational norms, including both formal and informal.
- **Connection** is related to the interpersonal relationships and information networks that newcomers must establish.

The extent to which an organization focuses on each C determines its onboarding strategy. The combination of tools, practices, recommendations, performance goals and measurement milestones constitutes an onboarding strategy, which is often formalized in an onboarding plan [1]. The success of an onboarding strategy is related to short-term and long-term outcomes. Short-term outcomes are associated with the adjustment of new employees to their new jobs [1]. They go through a series of four adjustments:

- **Self-efficacy** is the first level of adjustment and represents the degree that new employees feel confident when carrying out the work related to their new jobs. The more self-efficacy, the more motivated and successful an employee has the potential to be [23]. Furthermore, self-efficacy is associated with high job satisfaction and low turnover [21].
- **Role clarity** is the second level of adjustment and represents how well new employees understand their role and expectations. Measures of role clarity are seemed as effective predictors of job satisfaction and organizational commitment and performance [24].
- **Social integration** is the third level of adjustment and represents the extent to which new employees feel socially comfortable and accepted by their colleagues and superiors [25]. Effective social integration is related to committed employees and low turnover rates [1].
- **Knowledge of culture** is the fourth level of adjustment and represents the possession of knowledge about the prevalent organizational culture (politics, goals, values and a company’s unique language) and the extent to which the new employees fit to it [1].

Long-term outcomes of onboarding are related to attitudes and behaviors. In the long-term, effective onboarding leads to: higher job satisfaction, organizational commitment, lower turnover, higher performance levels, career effectiveness and lowered stress [1,26,27]. Although very important, long-term outcomes are not covered in this paper; our investigation encompasses the short-term outcomes of onboarding.

The functions of Bauer’s model are grouped into six categories:

- **Recruiting** - In many organizations, recruiting is not integrated with the onboarding plans and is treated as a separate function. However, existing literature shows that this integration (e.g. through realistic job previews or early involvement of stakeholders) gives to people being recruited a larger and more accurate amount of information about the company and job. As a result, this facilitates the adjustment of new employees, specially self-efficacy, role clarity and knowledge of culture [28].
- **Orientation** – Formal orientation programs help newcomers to understand important aspects of their jobs and organizations, as the company’s culture and values [29]. Moreover, they also help

newcomers feel welcome by presenting them to other individuals within the organization. Computer-based orientation programs can help to keep consistency to the program in different locations. This function facilitates all four types of adjustment [1].

- **Training** – They are mandatory to give to the new employees the confidence, clarity and skills required by their job. New employees can receive training about hard skills and soft skills. The type of training depends on the self-efficacy of new employees in relation to what is demanded by the job. As a result, training facilitates the adjustment of new employees, specially self-efficacy, role clarity and knowledge of culture [1].
- **Coaching and support** – Mentors can teach newcomers about the company, provide advice and help with job instruction. Existing research shows that new employees with mentors acquire more knowledge about their new company than the ones without mentors [30]. Furthermore, mentoring programs and opportunities for informal interaction with colleagues certainly help the new employees to adapt more easily to the new work environment. This function facilitates all four types of adjustment factors [1].
- **Support tools and processes** – Tools and formal processes are of great value for onboarding success. According to Bauer [1], there are three tools/processes that are related to successful onboarding: a written onboarding plan, which is a formal document that contains the timeline, goals, responsibilities and support available to each newcomer; stakeholder meetings, which occur in specific intervals, involve all the onboarding stakeholders, and allow newcomers to get the information they need; onboarding online, which can help to track the onboarding progress against development and career plans, and also help stakeholders to identify any additional help that new employees may need. This function facilitates all four types of adjustment factors [1].
- **Feedback** – Newcomers need constant feedback and guidance to understand and interpret the reactions of their co-workers. Feedback can be mainly provided in two different ways [1]: performance appraisals and 360-degree feedback, wherein the new employees are evaluated and receive developmental feedback and are also able to know how others view them; employee-initiated information and feedback seeking, wherein the new employees proactively seek feedback. This function facilitates the adjustment of new employees, specially self-efficacy, role clarity and knowledge of culture [1].

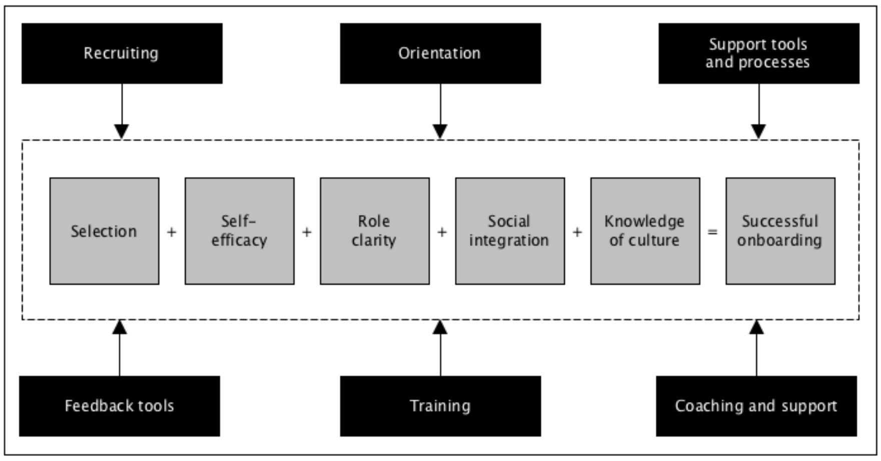

**Figure 1 – Bauer's model (adapted from Bauer** [1]**).**

## 2.5. Onboarding in the software development context

Some studies approach the onboarding topic in the context of software development (See Table 1), commonly addressing the functions of Bauer's model in isolation.

In relation to **recruiting**, Tockey [31] has investigated the recruitment of software developers and has identified a misalignment between what hiring managers ask for and what they really need. By mapping postings for software developer jobs and issues related to software projects, they identified that the postings did not include skills that could be useful to address identified problems (e.g., code review skills to address software quality problems).

Regarding **orientation**, the importance of effective socialization (a part of the orientation process in the Bauer's model) is emphasized by Casalnuovo et al. [32]. The study focused on the links between social relationships and onboarding. They explored GitHub to identify how the technical factors of past experiences and social factors of past connections to team members of a project impact their productivity. They identified that developers prefer to join projects in GitHub where they have pre-existing relationships. They also found that stronger social connections are related to lower initial and higher late productivity.

**Table 1 – Relationship between the related work and Bauer's model.**

| Study | ID | Main focus of the study | Strategy function category covered | Context |
|---|---|---|---|---|
| Tockey [31] | S1 | Recruiting | Recruiting | Unclear |
| Casalnuovo et al. [32] | S2 | Onboarding social factors | Orientation | Open source |
| Matturo et al. [33] | S3 | Soft skill training | Training | Open source |
| Falgerholm et al. [34] | S4 | Mentoring | Coaching and support | Open source |
| Falgerholm et al. [35] | S5 | Mentoring | Coaching and support | Open source |
| Britto et al. [36] | S6 | Mentoring | Coaching and support | Closed source |
| Labuschagne and Holmes [37] | S7 | Mentoring | Coaching and support | Open source |
| Monasor et al. [38] | S8 | Soft skill training tool | Support tools and processes | Academia |
| Steinmacher at al. [39,40] | S9 | Portal to support orientation and first contribution | Support tools and processes; Orientation | Open source |
| Cubranic at al. [41] | S10 | Propose a system to support mentoring | Support tools and processes; Coaching and support | Open source |
| Malheiros et al. [42] | S11 | Propose a system to support mentoring | Support tools and processes; Coaching and support | Open source |
| Canfora at al. [43] | S12 | Propose a system to select mentors | Support tools and processes; Coaching and support | Open source |

In relation to **training**, there is a vast literature that focuses on the training function of onboarding in the context of software engineering. While it is beyond the scope of this paper to cover comprehensively the literature related to training, and especially the hard technical skills, we would like to highlight a study related to soft skill training in the context of software engineering, which is key when onboarding newcomers according to Bauer [1]. Matturo et al. [33] conducted a field study to identify soft skills that are most important for product owners and Scrum masters. They identified that communication skills, customer orientation, and teamwork are the most important soft skills for product owners, while commitment, communication skills, interpersonal skills, planning skills, and teamwork are the most important soft skills for Scrum masters.

Regarding **coaching and support**, many studies focus on the role of mentoring for more effective onboarding. Falgerholm et al. [34] investigated the impact of mentoring on the onboarding of new developers in open source projects. They investigated that involved the onboarding of students from different universities in an open source project. Experienced developers mentored the students for three days. The activity of the mentored developers (students) was compared to the activity of non-mentored developers. They identified that the mentored developers were more active in the first 16 weeks of engagement than the non-mentored developers. In another study, Falgerholm et al. [35] investigated the influence of mentoring and project characteristics on the effectiveness and efficiency of the onboarding process of software developers in open source projects. They used quantitative measurements of source code repositories, issue tracking systems, and developer's discussions to identify how newcomers become contributing members of open source projects. They found that developers that receive deliberate onboarding support through mentoring were more active in the beginning. Britto et al. [36] investigated how software architects mentor new software developers and teams in a large-scale globally distributed legacy project. They identified that it is especially challenging to provide mentoring for teams located offshore, due to geographical and temporal distances. Furthermore, the existence of large amounts of complex legacy code amplifies the difficulty to mentor new remote development teams. Labuschagne and Holmes [37] investigated the effectiveness of mentored onboarding in open source projects supported by Mozilla Foundation. They compared two different onboarding approaches: one with the focus on easy bug fixing without mentoring, and the other focused on mentored bug fixing (more complex bugs). It was found that the programs that implemented the onboarding strategies were not enough to automatically improve the chances that a developer becomes a long-term contributor.

Finally, some **support tools and processes** have been proposed to support the onboarding of developers. Monasor et al. [38] developed a virtual environment to allow students to support soft skill training, which focuses on acquiring software communication skills in a more practical way. The virtual environment allows for the customization of different training scenarios, enabling instructors to adapt them according to different needs. Steinmacher at al. [39] created and evaluated a portal to support the onboarding of new developers in open source projects. The portal is based on a conceptual model of barriers [40] and the evaluation results suggest that it helped to alleviate barriers related to orientation and contribution process. Researchers have tried to facilitate mentoring through recommender systems. Cubranic at al. [41] and Malheiros et al. [42] developed tools that create a project memory automatically. The project memory can answer some questions made by newcomers, decreasing the need for human-based mentoring. In both cases, the tools were evaluated in the context of open source projects and were only successful to help newcomers to solve easy tasks. Canfora at al. [43] also developed a recommender system, but their focus was to allocate appropriate mentors for new developers in open source projects. They identified that the top committers are not necessarily the most appropriate mentors.

Three other studies worth mentioning in relation to our research question do not address any of the function categories directly but focus on barriers faced by new developers during the onboarding process. Steinmacher et al. [40] conducted a systematic literature review on the barriers faced by newcomers to contribute to open source projects. They identified 15 barriers that hinder the onboarding process of new developers in open source projects, five categories of barriers (social interaction, newcomers' previous knowledge, finding a way to start, documentation, and technical hurdle) and three origins (newcomers, community, or product). Steinmacher and Gerosa [44] conducted a survey to investigate in more detail the challenges that new developers face selecting the first task to start contributing in open source projects. They found that new developers do not have enough confidence to choose their initial task and, thus, need support from the open source community to select an appropriate task. In both studies, none of the functions of Bauer's model are addressed directly, but they indicate the importance of implementing them to address the identified barriers/challenges. And finally, Smite and van Solingen [45] studied an outsourcing relationship between a Dutch customer and an Indian vendor and found that it took longer for remote developers in India to climb up the learning curve in comparison with the developers onboarded onsite in the Netherlands. This was primarily due to the distance to the sources of requirements and development knowledge, and the lack of local developers with experience. They also found that onsite developers received more attention and training, while offshore developers were expected to be mentored and trained by the outsourcing vendor company.

As shown in Table 1, *feedback* is the only function category not covered by the related work, while *coaching and support tools and processes* are the most covered function categories. Furthermore, the coverage of onboarding function categories in the existing literature dedicated to software developers is fragmented (function categories are investigated in isolation). Finally, most of the studies we highlighted in this section focus only on open source projects. Therefore, to the best of our knowledge, no study reports a holistic investigation of software developer onboarding, accounting for all the function categories together in closed source globally distributed projects.

## 3. RESEARCH DESIGN

To address the research question formulated in this paper, we conducted an exploratory holistic multi-case study [46]. We investigated three different cases, which were selected through convenience sampling as adequate cases to investigate strategies to onboard software developers/teams. All cases involve globally distributed legacy projects. In this section, we describe each case, unit of analysis, and the data collection, preparation, and analysis processes.

### 3.1. Case Description

**Case 1** is a Polish company that provides services in dedicated IT solutions and applications, IT outsourcing, IT consulting, customizations and training. The main market segments of this company are telecommunications M2M (Machine to Machine), healthcare ERP (Enterprise Resource Planning), business intelligence applications, and finances and banking solutions. The company has subsidiaries in Sweden, Ukraine, and Belarus. The investigation focused on people involved in the development of telecommunication solutions. The people we interviewed work in a project in which the company provides software development services to another company with strong presence in the telecommunication market segment.

**Case 2** is a leading supplier of intelligent transportation systems to the public transport sector, including fare collection, travel information, infotainment, fleet management and traffic management. The investigated two teams consisting of nine members each including one tester in each team (in Poland), product owners are in Norway and customers mostly in Scandinavia. Software development teams follow a Scrum-based process, where a release is about 6 months long. The two teams have a complex environment of products and the domain knowledge is not easy to acquire. A large part of the systems consists of legacy code, which was produced by developers that are not anymore available. Some people are recruited to work on the new part of the system and some to work on the legacy part of the system only.

**Case 3** is a large-scale distributed project associated with the development and evolution of a large software product in Ericsson, a large Swedish company that commercializes telecommunications-related products. The product originated in Sweden and has evolved for almost 20 years, and, by the time of this investigation, involved 188 employees (15 software architects, 134 developers working in 19 formal teams) distributed across Sweden (five software development teams), India (10 software development teams), Italy (one software development team), USA (one software development team) and Poland (two software development teams).

### 3.2. Data Collection

In all three cases, we collected data through semi-structured individual and group interviews, and a workshop. Most of the interviews were conducted face-to-face. We recorded the audio and made notes during the conduction of the interviews/workshop. More details about the interviews are presented in Table 2.

To collect the data in **Case 1**, we conducted six individual interviews and two group interviews (90 min long). We interviewed the three product managers who are highly involved in the onboarding of new developers. We also interviewed the manager of the technical training center, two developers and two teams of developers to have different views on the onboarding process of Case 1 (60 min long). All the interviewees are from one of the sites located in Poland. The interviews were conducted face-to-face in Poland in May 2016.

To collect the data in **Case 2**, we conducted four group interviews, in which we interviewed two teams (45 min long), and four senior developers highly involved with the onboarding of developers (two group interviews, each 40 min long). All the interviews were conducted face-to-face in Poland in November 2016.

Finally, to collect the data regarding **Case 3**, we conducted three individual interviews and one group interview. We interviewed the product manager, who is highly involved in the onboarding of new developers. We also interviewed a software architect (located in Sweden), a senior developer (located in the USA) and conducted a workshop with a team of developers (located in India) to have different views on the onboarding process of Case 3. The data was collected from October 2016 to January 2017. All the interviews (60 min long) and the workshop (80 min long) were conducted in Sweden face-to-face, except for one that involved a developer located in the USA, which was carried out via Skype.

Through the interviews, we could identify how the onboarding functions are implemented in each case's strategy (if so). Furthermore, we could identify aspects that can be improved in each respective onboarding strategy.

**Table 2 – Details about the interviews.**

| Case | Type of interview | Length | Date | Role of interviewees | Experience |
|---|---|---|---|---|---|
| 1 | Individual | 60 minutes | 11/05/2016 | Software developer | 2 years |
| 1 | Individual | 60 minutes | 11/05/2016 | Technical training center manager | 5 years |
| 1 | Individual | 60 minutes | 11/05/2016 | Product manager | 12 years |
| 1 | Individual | 60 minutes | 11/05/2016 | Product manager | 10 years |
| 1 | Individual | 60 minutes | 11/05/2016 | Software developer | 8 years |
| 1 | Group | 90 minutes | 12/05/2016 | 6 Software developers | 2: < 6 months<br>4: >3 years |
| 1 | Group | 90 minutes | 12/05/2016 | 6 Software developers | 4: with < 1.5 year<br>2: > 2 years |
| 2 | Group | 45 minutes | 20/11/2016 | 2 Software developers | <3 years |
| 2 | Group | 45 minutes | 20/11/2016 | 2 Software developers | < 3 years |
| 2 | Group | 40 minutes | 20/11/2016 | 2 Software developers | > 5 years |
| 2 | Group | 40 minutes | 20/11/2016 | 2 Software developers | > 5 years |
| 3 | Individual | 60 minutes | 17/01/2017 | Product manager | 16 years |
| 3 | Individual | 60 minutes | 03/10/2016 | Software architect | 10 years |
| 3 | Individual | 60 minutes | 10/10/2016 | Design lead | 15 years |
| 3 | Workshop | 80 minutes | 21/12/2016 | 5 Software developers | > 2 years |

To conduct the interviews, we developed two semi-structured interview guides, one to interview managers (interview guide 1 in Appendix 1) and the other to interview developers (interview guide 2 in Appendix 2). The interview guide 2 was also used to conduct the workshop in Case 1.

### 3.3. Data Preparation and Analysis

Before analyzing the collected data, we transcribed all interviews related to Case 1 and Case 2. Then, we asked the interviewees to check the transcriptions. Two interviewees from Case 1 identified issues in their respective transcriptions, which were fixed. Regarding Case 3, we did not record the audio of the interviews and the workshop to make the participants more comfortable and prone to speak out. In all interviews, we took notes to facilitate posterior analysis. The notes included key points related to the questions that were posed during the interviews. The notes were discussed with the respective interviewees, to ensure that they reflected what was discussed during the interviews. In the workshop, we asked the participants to provide information about the challenges they faced during their onboarding, with a special focus on their learning process. The developers were asked to write down, independently of each other, the challenges that, in their opinion, impacted their learning processes. After 10 minutes, individual opinions were discussed within the group. The results were aggregated in a report that was verified by the participants.

To analyze the collected data, we followed the coding process described by Robison and McCartan (open coding) [47]. We used the function categories described in Bauer's model as primary codes. We coded the interview transcriptions, notes and workshop report using the defined codes, aiming at identifying how onboarding was strategized (implemented functions) in each case. Then, we determined the order and duration of the implemented functions in each case. We designed diagrams to visualize the functions (see Figures 2-5), emphasizing the functions dedicated to the legacy (functions in gray color) and distinguishing between the functions planned and implemented by offshore sites locally and functions planned and implemented centrally by the main sites. The coverage of the onboarding functions, as well as similarities and differences between the cases, were then summarized (see Table 3) and discussed.

Finally, we derived recommendations for software companies onboarding developers and teams in globally distributed legacy projects (see Section 5). To do so, we paid special attention to functions that were implemented in the investigated cases to address the challenges of onboarding on a distance, or dealing with legacy code. Some of the practices drive the readers' attention to the challenging areas and warn about the necessity to have a proactive action plan, while others propose concrete action. The recommendations resulted from the combination of practices well-evaluated by the interviewees and our observations from the three cases.

**Table 3 – Summary of the onboarding tools.**

| Category | Function | Case 1 | Case 2 | Case 3 |
|---|---|---|---|---|
| Recruitment | Recruitment integrated with onboarding | 🟩 Fully integrated | 🟨 Partially integrated | 🟨 Partially integrated |
| | Realistic job previews for newcomers | 🟩 Summer schools | 🟥 Not implemented | 🟥 Not implemented |
| | Stakeholder involvement in recruitment | 🟩 Senior developers participate in technical interviews | 🟩 Senior developers in already established sites participate in technical interviews and CV screening | 🟩 Senior developers in already established sites participate in technical interviews and CV screening |
| Orientation | Formal orientation course or material for the newcomers | 🟥 Not implemented | 🟥 Not implemented | 🟥 Not implemented |
| | First day at the job is special | 🟩 The whole first week at the new job is dedicated to familiarization with the environment | 🟨 The first month is dedicated to learning on the job. | 🟩 The whole first week at the new job is dedicated to familiarization with the environment |
| Coaching and support | Mentoring programs | 🟩 A mentor assigned to new developers inside a boot camp. Afterwards, a mentor assigned to a new developer or a mentor team assigned to a team of new developers | 🟩 One or several mentors assigned to new developers in a team. | 🟩 A mentor assigned to a new developer or a group of new developers |
| Training | Formal training on hard skills and/or soft skills | 🟩 3 months long training focusing on technical and methodological knowledge, provided in boot camps | 🟥 Not implemented | 🟨 If many new comers, 3 months long training focusing on technical, methodological, and product knowledge |
| Support tools and processes | Onboarding plans | 🟨 Partially supported<br>An intranet with useful company material | 🟨 Partially supported<br>An intranet with useful company material and an instruction page | 🟩 Established<br>Onboarding plans, an intranet with company material |
| | Regular stakeholder meetings | 🟩 Face-to-face meetings | 🟩 Face-to-face meetings | 🟩 Face-to-face and videoconferencing |
| | Own progress monitoring | 🟥 Not implemented | 🟥 Not implemented | 🟩 Progression spreadsheets |
| Feedback | Performance appraisals | 🟩 Face-to-face during meetings with mentors and immediate managers | 🟩 Face-to-face during appraisal meetings with immediate managers | 🟩 Face-to-face during meetings with mentors and immediate managers |
| | 360-degree feedback | 🟨 Feedback from mentors via code reviews and face-to-face | 🟨 Feedback from mentors via code reviews | 🟨 Feedback from mentors via code reviews |

## 4. RESULTS

In this section, we present the results associated with each case. In all cases, the companies employed semi-formalized onboarding strategies. Table 3 summarizes the results, wherein green color represents fully implemented functions, yellow represents partially implemented onboarding functions, and red represents not implemented functions (see Section 3 for more details about the onboarding functions and categories). The results are further elaborated in the remainder of the section.

### 4.1 Case 1

The summary of the onboarding functions implemented in Case 1 is given in Figure 2.

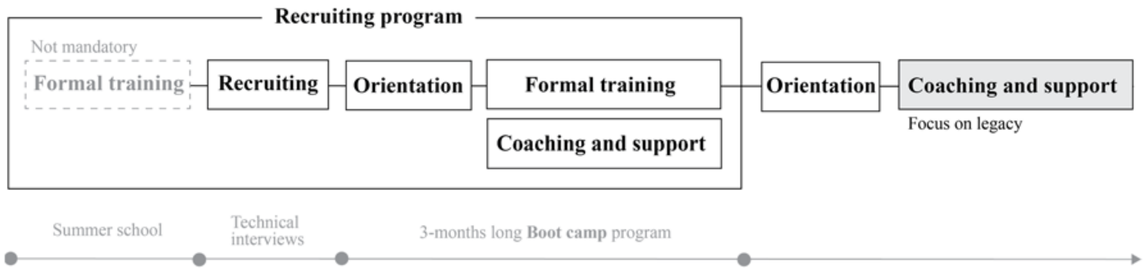

**Figure 2 –** Summary of the onboarding functions in Case 1

In Case 1, *recruitment* is integrated with the existing onboarding strategy. The company organizes summer schools for newly graduated bachelors from the local universities, during which the young professionals receive technical training within the actual software teams. This provides the candidates with realistic job previews. Successful attendees of the summer school receive a job offer. In this case, a series of job interviews may be carried out. Senior developers participate in the evaluation of candidate developers during the technical interviews. Selected candidates proceed to a special onboarding program called "boot camp" and receive an initial 3-month contract, which can be extended if the new developer performs well and shows good potential. Other members of the team, where a new developer is allocated, conduct the evaluation of performance and potential.

A boot camp is a 3-month long program in which small groups of newcomers must carry out a small project, usually involving a toy task for learning purposes and to be able to evaluate the potential of the candidate. Boot camp aims at learning the intended programming languages, tools, environments, and ways of working, and facilitates socialization and team building. During a boot camp, a senior developer is assigned to support a group of developers and is responsible for facilitating the required training and provide knowledge of corporate behavior. Therefore, it is fair to say that boot camps serve as a premise for orientation, training, and coaching.

There is no formal *orientation* of the new developers. The welcoming of the newcomers in a boot camp is done in an informal way is conditional in its nature, since the more permanent recruitment of developers happens upon the successful completion of the boot camp, three months later.

*Coaching and support*, as noted earlier, is a central part of a boot camp program. The mentors assigned to the developers are used as the first-hand contacts for any help needed. After a boot camp, the new developers receive additional support, but the way it is provided by the company differs depending on the current circumstances. If few developers are onboarded, each one of them will be assigned to a mentor (senior developer) within the team. If many developers are onboarded together, often a new team is created with the new developers. The new team is assigned to a mentor team, which is a mature software development team. In general, the new team works on the backlog of the mentor team, "shadowing" the work of the mentor team in the beginning and doing small tasks more independently afterward. This allows an experienced team to take off some workload while helping the new team, without losing overall productivity. Further support is provided through communities of practice, wherein new developers can interact with senior developers to acquire technical and methodological guidance. It is important to note that socialization and seek help from others is emphasized in the company and noted by the new recruits.

Although the company employs a learning-by-doing approach, i.e. there is no big emphasis on traditional classroom training, the company facilitates this through formal *training programs* provided during the summer schools and the boot camps. There are two types of training: external training, which are focused on technical (e.g. domain, programming languages) and methodological (e.g. agile practices) training, and are provided to all units by the corporate training center; and internal training, which are focused on product knowledge and are provided by a unit for its developers/teams. After receiving the formal training, the new hires are further incorporated in the teams to "shadow" the more experienced developers in their daily tasks. This is an important and relatively secure way of acquiring the product knowledge and the knowledge of the legacy code. The company also provides soft skill training (e.g. multi-cultural training), but they are only offered at the later stages of the developers' careers.

The company provides some *support tools and processes*, including an intranet and wiki pages, wherein documents about the ways of working and the product are available.

When it comes to *feedback,* the company facilitates two types of feedback. The candidates have a possibility to provide feedback on the onboarding process by requiring additional training within a boot camp or after being incorporated into a team. The candidates also receive feedback from the mentors face-to-face or via code reviews. Mentors and immediate managers also evaluate and provide feedback about the performance of newcomers during performance appraisal meetings.

### 4.2 Case 2

The summary of the onboarding functions implemented in Case 2 is given in Figure 3.

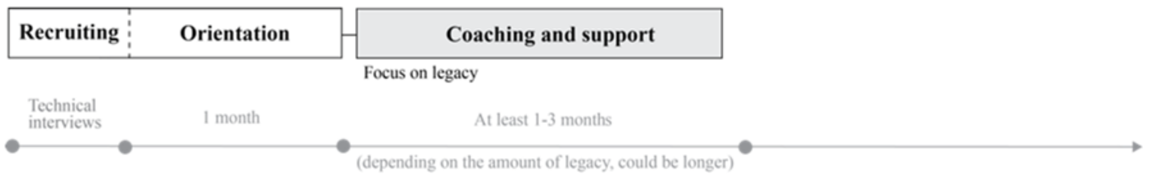

**Figure 3 –** Summary of the onboarding functions in Case 2

R*ecruitment* in Case 2 is partly integrated with the onboarding process. Although no formal realistic job preview is provided, senior developers are included in the process of recruitment, both during the CV (curriculum vitae) screening and when conducting technical interviews. The two main factors of success mentioned during the interviews were related to a good technical meeting and knowing whom they will work with if employed. These factors determined whether the recruitment process was good or not. Also, the existing developers perceived that it is important to be a part of the recruitment decisions. Recruited developers commented that the technical interview allowed them to know more about the work in the company.

The *orientation* of new developers is done in an informal way, i.e. there is no formal orientation program. In general, the first week at the new job is dedicated to familiarizing with the new environment and the new recruits are given one month to learn their way. However, this process is not formalized. One interviewee commented that he would like to have more information about the company structure, and who is responsible for each department. Otherwise those new to the company must ask different people about who is responsible for what, knowing who is who is especially hard in a globally distributed company. Another interviewee also commented that the goals of the project must be clear since the beginning.

The company provides *coaching and support* for the new developers, but the way it is done depends on the number of people being onboarded. In each team, there is someone that does the mentoring for the new employees, depending on which part of the system the new employee will work on. Some projects are very complex with respect to domain-specific knowledge, so there is nobody who is knowledgeable in all aspects. There is one month learning time given by the company to a new employee, but there is no clearly separated time for the mentors to spend time with the new employees. It is informally known that the mentors' productivity will slow down, but they do not have any dedicated time for mentoring explicitly. At the same time, they do activities such as explaining the domain, showing how to setup the environment, doing code reviews and answering the questions as they arise. Work as a mentor puts additional pressure on the mentors and other experienced team members because they must keep up with their own productivity and at the same time take care of the new employees. The best mentoring strategy happens when there is an overlap of the time when the senior developer is leaving the company and the new one is arriving, so the new developer can receive the introduction from the person that he/she is replacing. Some new developers thought that one month was too much time for just learning, and after two weeks they have started to take some simple tasks to solve. The company does not provide any formal *training* for the new employees.

Regarding *support tools and processes*, the company has an intranet wherein documentation is made available, technical sessions about architecture of the system and documentation ("how to" navigate on the wiki pages, confluence etc.), and an instruction page (how to set up environments, virtual machines, where to find information needed, who has specific knowledge and acronyms and glossary of terms). Not all information is equally structured and it may be hard for new developers to find the information they need.

*Feedback* is mainly used to identify whether a new developer needs more support or identify whether it is worthwhile to hire a new developer permanently. Appraisal meeting with all employees are organized every six months, wherein the employees receive feedback on their performance and have an opportunity to speak about their own experiences in the company. The team members also use code reviews to support the onboarding process of new developers.

## 4.3 Case 3

In Case 3, we explored the challenges associated with remote onboarding, which at the same time was carried out in the context of a complex legacy product. Given the changing market demands, developers and teams were frequently onboarded in the project we have studied in Case 3. This happened both locally in Sweden and remotely in USA, Italy, China, Turkey, Poland, and India. During the 20 years of product existence, over 30 teams were onboarded and more than half removed from the project, not to mention individual onboarding. Given the distributed nature of the project some onboarding functions were organized by the central project stakeholders in Sweden, while others occurred remotely and were handled by the local management. The processes for onboarding differed for sites with considerable experience and already established teams from those sites, which just started their engagement in the project. The two onboarding strategies employed in Case 3 are presented in Figure 4 and 5 respectively.

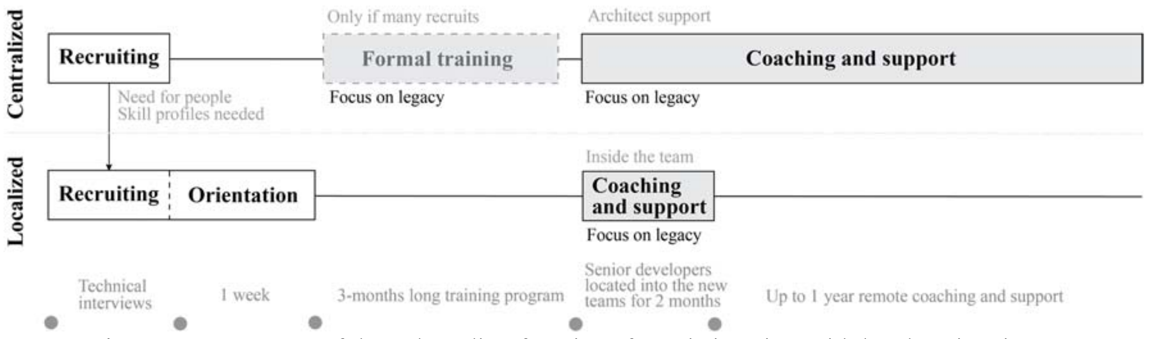

**Figure 4 –** Summary of the onboarding functions for existing sites with local seniors in Case 3.

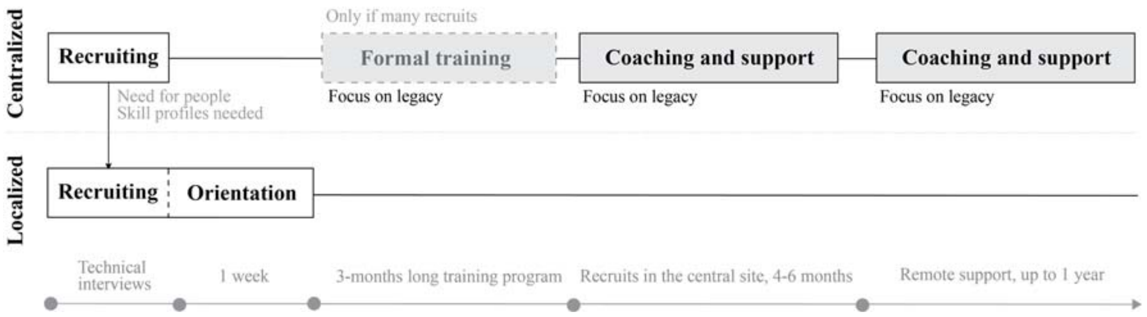

**Figure 5 –** Summary of the onboarding functions for newly established remote sites in Case 3.

The *recruitment* of the new employees is organized by each site individually, while the demanded skill profiles are formulated by the central location. The company does not provide realistic job previews; however, senior developers participate in the recruitment process and the evaluation of new candidate developers during the technical interviews in the already established sites local. In general, inexperienced developers have an initial 6-months contract and experienced developers have an initial 1-month contract, which can be extended depending on the performance of the new employees, which is evaluated during the time of the first contract. Senior developers, commonly those from the central location, evaluate new developers by reviewing their work outcomes, i.e. the code. Overall, we conclude that the recruitment is partially integrated with the onboarding process.

The *orientation* of new developers is carried out in an informal way, i.e. there is no formal orientation program, and is up to the local stakeholders define how to implement orientation. In general, the new recruits are given one week at the new job to familiarize with the new environment, get to know the key people and coworkers (socialization), and acquire the basics about the existing ways of working. The way it is done depends on the number of people being onboarded; if few people are onboarded, the orientation is carried out on an individual level; and if many developers are onboarded at the same time, the company provides a group level orientation. In both cases, a senior local developer or a manager provides an informal walk-around and discussion-based orientation.

The company provides three months long formal *training* for the new employees when many developers are recruited. The training program focused mainly on product knowledge and required technical and methodological skills. The company also provides training related to soft skills, e.g. a cross-cultural communication course for those working in a distributed way, but they are not part of the onboarding strategy. Rather, they are provided in a later stage of an employee's career. If just one person is to be onboarded, no formal training is provided.

The company employs a learning-by-doing approach and new developers soon start to work with real tasks under careful *coaching and support*. Depending on the number of people being onboarded, the company assigns mentors on an individual or a group level. The mentors are responsible for ensuring that the new employee or employees do not cause problems that impact the whole product. Therefore, the new hires often start by developing test cases for the product regression testing framework under careful supervision and then move on to more challenging tasks. Notably, the onboarding process differs for the already existing sites and the newly established sites. When many developers are onboarded in an already established site, the common practice is to group them and allocate one or two senior developers into the new team. In some cases, these groups are separated after two months, the new developers are integrated in existing teams, and the senior developers return to their original teams. The special situation arises when the new developers are onboarded in remote sites with no senior developers present for local coaching and support. It is often hard to relocate enough senior developers from one country to another for a long period of time. To address this, the company first brings the new developers from the remote location to Sweden for both training and the initial period of supervised work (four to six months in total). Then a senior developer follows the developers to the remote location to provide on-site mentoring for the next four to six months. Finally, the new developers continue receiving support on a distance.

*Regarding support tools and processes*, the company uses many resources to make the onboarding of new developers successful. All tools and processes are centralized. For the remote teams, there is a formal onboarding plan, with the goals, milestones, and training associated with the new developers. As for the other functions, the goals are defined at an individual or group level depending on the number of people being onboarded at the same time. For each developer, regardless location, there is an Excel spreadsheet used to track their progression regarding the competence they must acquire. This spreadsheet also contains the main source of knowledge they can use to acquire the required competence. This file must be updated to a system, which is used by immediate managers and mentors to also follow the progression of new developers. Another tool is the corporate intranet, wherein documents about the ways of working and the product are available, and which is maintained centrally.

The new developers receive continuous *feedback* on their work outcomes (i.e. code) through code reviews. Local senior developers (if any) and software architects from the central location evaluate the performance and transfer product knowledge to support the new developers. Based on the received feedback, new developers may require more formal training. At the same time, the status of performance is used by the immediate managers locally to identify whether a new developer needs more support. Such checks are performed together with the mentors on a weekly basis. Code reviews also serve as a track record used in permanent employment considerations. Local stakeholders decide whether to hire or not permanently a new developer.

## 5. DISCUSSION

In this paper, we have used the Bauer's model for successful onboarding to analyze the onboarding functions and strategies in three software companies diverse in domains and size. In the following, we first discuss the degree to which each organization has formalized and structured their onboarding strategies. Then, we discuss the new challenges for onboarding developers to work with legacy systems and when onboarding remotely. Based on the results, we bring forward a list of recommendations that can support companies to improve their onboarding strategies, as well as some implications for future research on this topic.

### 5.1 Formality Level of Onboarding Strategies

As noted earlier, companies may employ formal (institutionalized) or informal (individualized) onboarding processes [5–7], which depends on whether new employees learn about their new job on their own or following coordinated policies, procedures, and actions set by the company. Bauer [1] and other researchers [18,21,22] suggests that successful companies treat onboarding of new employees more formally, which can be done by specifically addressing the four essential components of onboarding – Compliance, Clarification, Culture, and Connection [1]. In other words, to make the onboarding successful, companies are expected to explicitly support newcomer familiarization with the legal policies and regulations, job-related training, organizational norms, and networking and building interpersonal connections with other employees important for completion of the job tasks.

In our investigation, we have specifically focused on the clarification and the connection building blocks in Bauer's model. Unfortunately, we did not study the familiarization with the legal policies, regulations, and the corporate culture because it requires much more observation and an ethnographic approach.

In relation to Connection, the case companies employed a few practices that primarily included daily work practices for all employees and were not formalized as a part of the onboarding practices. For example, walk-through-the-office type of introductions of the newcomers, exchange visits, and other contact building activities are used as a part of the orientation, but are not institutionalized. Participation in communities of practice, and team events were other sources of new contact acquisition. The more formalized functions based on the three studied cases were related to the coaching and support. In Case 1, for example, the company integrated the newcomers in existing teams to foster interpersonal connections. To the best of our knowledge, related literature in software engineering does not focus on this component.

With respect to clarification, we found that in all three case companies the most formally treated activities were those related to ensuring that newcomers understand their new job. Related literature [31] suggests that companies shall also clarify their expectations and provide candidates with detailed information about a particular job, to enable them to better decide whether the offered position matches their aspirations. However, we found that among the studied companies only one (Case 1) provided realistic job previews, but in Case 2, the team members participate in the interviews and can answer the questions of the newcomers and all get to know who they will be working with. After recruiting the new developers, all three companies put a strong emphasis on further anchoring the understanding of the job routines through a formally established mentoring program, as also suggested by related literature [34,35] (S4 and S5). In fact, in two of the cases (Cases 2 and 3), assignment of mentors was the prime onboarding function. The actual job clarification and feedback was enabled through less formal process employed by the mentors and other employees in general. Finally, the three companies employed code reviews (over-the-shoulder, email pass-around, pair programming or tool-assisted code review) to enforce clarification, which is also suggested as a good practice by existing literature [48].

Interestingly, we found that depending on the number of people being onboarded, companies employed different levels of formality to onboard software developers, which we have not come across in the related literature. This impacted the offering of formal training programs, the number of mentors, the allocation of new developers into existing teams or formation of new teams, as well as the duration of training, coaching, and support. A valid question for future investigation is whether this means that large groups of new developers (onboarded in a more formal way) have a better onboarding results, or individual treatment and less formal strategy result in a better onboarding outcome. Unfortunately, we are not unable to answer this question in this paper.

## 5.2 Onboarding Challenges for Globally Distributed Legacy Projects

In all three investigated cases, the onboarding processes were additionally challenged because of the considerable amount of legacy code that the newly employed developers had to learn. To support the newcomers, all companies provided formal training about the product knowledge and different ways to coach and support them on the job. Although we know from existing literature that mentoring is a common practice for onboarding software developers [34,35], we observed in our case studies that in legacy projects, coaching and support may be required for a significantly longer period. We also found that this may negatively impact the mentors, as our interviewees complained about the mentoring role being a heavy slog. It affects the mentors' productivity since they must stop their work to respond to the new employees' requests and spend time on task switching due to disturbances. In Case 3, besides the main retainers of knowledge being in another country, much of the legacy code was written by people that no longer worked in the company, demanding even longer periods of intense mentoring, often provided offsite.

Our findings also suggest that the largest challenge for companies is to onboard remote developers to an ongoing product development, especially if it follows agile methodologies. Existing literature shows that it may be hard for developers to start being productive when they are onboarded remotely (S9) [39,40]. In all three companies, agile ways of working meant that there is generally a higher emphasis on networking with colleagues rather than documenting the progress, work outcomes and the ways of working. Even though all three companies have put effort into making the general guidelines and product descriptions available through the corporate intranet and wiki pages, the maintenance of consistent documentation with high coverage was a challenge. The interviewees from Cases 2 and 3 reported that parts of the products they worked with were not up to date or insufficiently detailed. This meant that new developers were required to keep a continuous dialog with the mentors, which especially in Case 3, was challenged by the temporal and geographic distance.

Networking was also problematic because of the lack of cross-site contacts. Some of the interviewees stated that they do not know in person some of the developers with whom they need to collaborate. Existing literature shows that stronger social relationships are related to higher productivity [32] (S2). Thus, to facilitate more personal contacts, the studied companies have invested into video conferencing facilities, frequent visits from headquarters to the remote sites, and exchange between developers from different sites.

Another issue we identified is related to onboarding strategy fragmentation (related literature does not focus on such an aspect). In projects with multiple sites, it may be impossible to implement all onboarding functions in all sites in the same way. For example, in Case 3 the recruiting function was planned and performed locally, which meant that different processes and criteria were employed to hire developers in each site. It is fair to assume that different ways of planning and implementing the onboarding functions may lead to uneven onboarding results and that a company can be successful in one site but fail in another due to the local differences. Although we have not investigated the onboarding success or the process differences in detail, the very fact that companies do not always have control over parts of the onboarding processes is an interesting finding.

## 5.3 Recommendations and Implications for Practice

In this section, we outline the recommendations based on the cross-company analysis of the onboarding functions and the lessons learned by each of the studied companies, followed by a few general implications that we derive from the execution of our investigation.

For the companies that onboard developers in distributed legacy projects we advise:

- **During recruitment explain the expectations for the new hires**. As one of the interviewees from Case 2 stressed: *"The technical interview made a difference on choosing to work here because I got to know what potentially I would do, what technologies I would be exposed, and how do they work here, and some of technologies and the way they work, like that they follow Scrum."* Among our cases this was the only company providing realistic job previews.
- **As a part of orientation in distributed projects, acknowledge the importance of formalizing and mirroring the onboarding program across sites**. Make the objectives, timelines, roles and responsibilities clear. Revise what a new employee needs to learn and summarize it in a written onboarding plan. Write up a list of orientation activities in a guide for the mentors that perform the

orientation and/or the hand-out material for the newcomers.

- **Provide transparency into the project organization and key roles across sites**. As one of the offshore interviewees from Case 1 revealed: *"One thing that comes to my mind is that our project is managed basically not here. It is managed abroad. Especially at the beginning, it was very difficult to get to understand, how exactly is it managed and what is required. What aside from basic things that we must develop code and deliver somehow, what else is required of us in terms of process and testing, and filling forms and documents. And some meetings that take place in the other site, for instance, perhaps contained some important information that we missed, because we didn't know that some meetings did take place and some information is just passed through unofficial channels and we miss out on that."*
- **Invest in traveling**. While walk-through-the-office is a common onboarding practice when performing orientation of the newcomers on site, developers onboarded in distributed projects shall receive a special kind of orientation to support the establishment of ties with the remote colleagues. One of the interviewees from Case 1 said: *"It is very important to travel to other sites to meet face-to-face people I work. It helps to make you more aware of what is happening, where to pay attention, and who to ask if something is unclear. Before, I was aware of people who can help me and had knowledge I need, but it was hard to ask someone, who I didn't really know. After I visited the other site, it became much easier, it's almost like going to other room in your workplace. It is a lot easier to ask and it is also easier for the person who you ask help from."*
- **Provide extensive coaching and support in legacy projects**. Remember that familiarization with legacy code requires hands-on training, as an interviewee from Case 1 revealed: "*Bootcamp is a good thing […], but there is a huge amount of knowledge to gain [...]. When you start using the things you are learning, working for a while, you remember what you have learned and then everything becomes clearer*". Our studies indicate that the more legacy a product contains, the longer the mentoring period. Therefore, companies shall we prepared to prolong the mentoring, if needed. At the same time, it is important to include coaching in the individual plans for the mentors to reduce their stress.
- **Tailor the training program to suit the developers' needs**. In Case 3, we learned that the provided onboarding activities can potentially clash with the expectations from the new hires offshore. As an interviewee from Case 3 explained: *"Here we expect our developers to learn by doing. So, as soon as possible, we involve them in real tasks. However, people are different and some developers ask for more training than others. In my opinion, this is also related to cultural differences."*
- **Use tools to provide feedback**. In all three case companies, code reviews were used to provide daily feedback on the work outcomes of the new hires. This was appreciated by the newcomers, as one of the recently onboarded developers from Case 2 mentioned: "*We have the process of code review, so we have discussions and comments on our code, and then testing. So, it is a good way to get feedback.*" At the same time, the companies shall also acknowledge the limitations of tool support**.** The case companies emphasized the importance of collocated mentoring in distributed projects. As one interviewee from Case 3 stressed: "*We help the developers located in India mainly via code reviews and Skype. These approaches work, but it is much easier to help them when they are here in Sweden."*

We would like to emphasize the importance of treating the onboarding as a multifunctional process. Our cross-case analysis indicates that the coverage and sophistication of the onboarding programs may differ based on the organizational contexts (e.g., the number of newcomers, the amount of legacy code, and the distribution across multiple locations), It seems difficult to come up with one general onboarding strategy that would be useful in all circumstances. For example, a full-fledged onboarding program, as proposed by Bauer, might be economically unfeasible for onboarding a single developer. Rather, the incorporation of a developer into a well-functioning team and their mentoring and support could suffice in this scenario. Nevertheless, a similar lightweight onboarding program may be insufficient for onboarding many new developers in a large legacy project. Therefore, our recommendation is to tailor onboarding strategies for the onboarding undertaking at hand.

## 6. THREATS TO VALIDITY AND LIMITATIONS

In the following, we discuss the validity threats associated with reliability, internal, construct and external validity described by Runeson and Höst [46].

Reliability is related to the repeatability of a study, i.e. how dependent are the data and analysis on the involved researchers [46]. To minimize this threat, three researchers were involved in the design and execution of this multi-case study. Furthermore, we developed an explicit case study protocol to guide the investigation as suggested by [46]. Finally, the observations and findings were verified with the companies' representatives to avoid false interpretations and inconsistencies. Besides, the researchers have a long-term relationship with the case companies, which means that the researchers know more about the context of the companies, not basing the findings and interpretations only on the focused interviews. However, the collected data is qualitative and is highly dependent on the involved interviewees. We mitigated this factor by using a known theoretical model for onboarding that is used in other areas.

Internal validity is related to factors that researchers are unaware of or cannot control regarding their effect on the variables under investigation [46]. The main internal validity threats related to this paper are investigator bias and interviewee bias. Investigator bias was mitigated by involving three researchers during the design of the interview and workshop guides (investigator triangulation). To mitigate interviewee bias, we interviewed people with different roles (data triangulation).

Construct validity reflects how well the measures used actually represent the constructs the study intends to measure [46]. The main threat to construct validity in our investigation is that we used only one method to measure each construct. Data triangulation (interviewing multiple people) partially addresses this threat and strengthens the produced evidence. Moreover, we conducted a sanity check together with company representatives to validate the collected data.

External validity is concerned with the generalization of the findings [46]. Since the main research method employed in this work is the case study method, the main findings are strongly bound by the context of the selected cases. To mitigate this threat, we conducted three case studies in three different companies that provide services and products in different domains. Furthermore, our main contribution lies in the cross-company comparisons based on which we concluded about the variability of the onboarding practices. As such, it shall be valid despite the limitations of the cases. The main contributions of this paper may be of interest and applicable to researchers and practitioners that work in similar contexts. To allow the transferability of the findings of this work, we detailed the description of the investigated cases, within the limits imposed by the associated non-disclosure agreements. Note that more details were provided regarding the Case 3 since we could describe the case in such a level of detail. We had to omit the name of the companies of Cases 1 and 2 due to the non-disclosure agreements we signed.

## 7. CONCLUSIONS AND FUTURE WORK

In this paper, we investigated the strategies employed by three different companies to onboard software developers in globally distributed legacy projects. To do so, we used the model for successful onboarding proposed by Bauer [1] and evaluated the coverage of the onboarding functions in all three cases.

In response to our research question, we learned that the employed onboarding strategies are semi-formalized and vary from company to company, and depend on the context of onboarding. When it comes to the specifics of the context of the chosen companies, i.e. the distribution and the legacy products, we found that these aspects significantly challenged the onboarding of software developers. We identified that in globally distributed projects some parts of the employed onboarding strategies differ between different sites. This means that some of the onboarding functions are centralized, while others are executed locally on the site level. One important implication of this finding is that the onboarding outcomes cannot be always predicted and controlled by the central management.

We also learned that onboarding in globally distributed projects with legacy code is especially challenging, due to the difficulty to connect newcomers onboarded in offshore locations with the original developers, the difficulty to learn the legacy code, and potential onboarding strategy fragmentation due to the distribution. Our results also suggest that legacy projects, even co-located, require hands-on training and longer mentoring periods than new projects.

The use of the Bauer's model facilitated in the analysis of the coverage and prevalence of the different onboarding functions. Among the case companies, the most common practice is coaching and mentoring, where the companies incorporate the new employees into existing teams and rely on the ability of the experienced team members to help the newcomers in their learning process. Some drawbacks of this approach is that this hinders productivity of the mentors, who experience frustration and stress associated with the inability to do their own work.

In our investigation, we also reflect on the repertoire of the onboarding strategies of the case companies, compared to the experience from companies from various disciplines collected by Bauer. While we are unable to judge on the sufficiency or efficiency of the documented strategies, we can see how the case companies can enhance their onboarding programs. For example, orientation was the most neglected function in our three cases.

Finally, we put forward the following future research directions:

- More empirical studies investigating onboarding in a holistic way. For future studies, we believe, it would be beneficial to conduct more empirical research, both investigating onboarding in a holistic way (accounting for all onboarding function categories together and their interplay) and investigating separate onboarding function categories in depth.
- Quantitative studies that evaluate the effectiveness of onboarding strategies. By the time we conducted this investigation, we were not able to collect quantitative data. However, we believe that such data as retention of developers and performance of the onboarded developers and teams over time might allow understanding which onboarding strategies lead to better outcomes.
- Synthesis of the experiences with training as an onboarding function category. Training is the most studied onboarding function category in existing software engineering literature, with even venues fully dedicated to the topic (e.g. Conference on Software Engineering Education and Training, CSEE&T). However, to the best of our knowledge, there is no study that aggregates the existing empirical evidence. Therefore, we suggest carrying out a comprehensive secondary study to portray the state-of-the-art with respect to training as a part of onboarding strategies.
- Linking the findings from the motivation research and the onboarding research. Existing literature suggests that software developers are a distinct occupational group with particular drives or motivators [49]. Software developers are said to have a high need for growth and independence. It is fair to assume that software developers may, therefore, have also particular onboarding needs in comparison to other occupational groups. One important question for future research is then to understand the ways how to enrich the onboarding strategies to better outline the growth opportunities for the new software developers and provide them with the sense of independence as soon as possible.

## ACKNOWLEDGEMENT

This research was funded by the Swedish Knowledge Foundation (Grant 2009/0249), and by the Research Council of Norway (Grant 235359/O30). The authors are very thankful to Ericsson and all the company employees involved in and being sincerely interested in our research.

## REFERENCES

1. Bauer, T.N. (2011) Onboarding new employees: Maximizing success. *SHRM Found.*
2. Cable, Daniel M.; Gina, Francesca; & Staats, B. (2013) Reinventing Employee Onboarding. *MIT Sloan Manag. Rev.*, **54** (3), 22–29.
3. Bauer, T.N., and Green, S.G. (1994) Effect of newcomer involvement in work-related activities: a longitudinal study of socialization. *J. Appl. Psychol.*, **79** (2), 211–223.
4. Klein, H.J., Polin, B., and Leigh Sutton, K. (2015) Specific Onboarding Practices for the Socialization of New Employees. *Int. J. Sel. Assess.*, **23** (3), 263–283.
5. Zahrly, J., and Tosi, H. (1989) The differential effect of organizational induction process on early work

role adjustment. *J. Organ. Behav.*, **10** (1), 59–74.

6. Louis, M.R. (1980) Surprise and sense making: What newcomers experience in entering unfamiliar organizational settings. *Adm. Sci. Q.*, **25** (2), 226–251.
7. Louis, M.R., Posner, B.Z., and Powell, G.N. (1983) The availability and helpfulness of socialization practices. *Pers. Psychol.*, **36** (4), 857–866.
8. Lynch, K., and Buckner-Hayden, G. (2010) Reducing the new employee learning curve to improve productivity. *J. Healthc. risk Manag.*, **29** (3), 22–28.
9. Rocha, J., Rollag, K., and Johnson, L. (2005) Get your new managers moving. *Harvard Manag. Updat.*, 3–6.
10. Bauer, T., Morrison, E., and Callister, R. (1998) Organizational socialization: A review and directions for future research. *Res. Pers. Hum. Resour. Manag. Vol 16.*, **16** (January), 149–214.
11. Gruman, J.A., Saks, A.M., and Zweig, D.I. (2006) Organizational socialization tactics and newcomer proactive behaviors: An integrative study. *J. Vocat. Behav.*, **69** (1), 90–104.
12. Bauer, T.N., and Erdogan, B. (2011) Organizational socialization: The effective onboarding of new employees, in *APA handbook of industrial and organizational psychology, Vol 3: Maintaining, expanding, and contracting the organization*, pp. 51–64.
13. Van Maanen, J., and Schein, E.H. (1979) Toward a theory of organizational socialization. *Res. Organ. Behav.*, **1**, 209–269.
14. Buchanan, B. (1974) Building organizational commitment: The socialization of managers in work organizations. *Adm. Sci. Q.*, **19** (4), 533–546.
15. Feldman, D.C. (1976) A contingency theory of socialization. *Adm. Sci. Q.*, **21** (3), 433–452.
16. Morrison, E.W. (2002) Newcomers' relationships: The role of social network ties during socialization. *Acad. Manag. J.*, **45** (6), 1149–1160.
17. Ashforth, B.E. (2001) *Role transitions in organizational life: An identity-based perspective*, Mahwah, NJ:Lawrence Erlbaum Associates.
18. Klein, H.J., and Heuser, A. (2008) The learning of socialization content: A framework for researching orientating practices. *Res. Pers. Hum. Resour. Manag.*, **27**, 278–336.
19. Chao, G.T., O'Leary-Kelly, A.M., Wolf, S., Klein, H.J., and Gardner, P.D. (1994) Organizational Socialization: Its contents and consequences. *J. Appl. Psychol.*, **79** (5), 730–743.
20. Jones, G.R. (1986) Socialization Tactics, Self-Efficacy, and Newcomers' Adjustments To Organizations. *Acad. Manag. J.*, **29** (2), 262–279.
21. Bauer, T.N., Bodner, T., Erdogan, B., et al. (2007) Newcomer adjustment during organizational socialization: A meta-analytic review of antecedents, outcomes and methods. *J. Appl. Psychol.*, **92**, 707–721.
22. Cable, D.M., and Parsons, C.K. (2001) Socialization tactics and person-organization fit. *Pers. Psychol.*, **54** (1), 1–23.
23. Saks, A.M. (1995) Longitudinal field investigation of the moderating and mediating effects of self-efficacy on the relationship between training and newcomer adjustment. *J. Appl. Psychol.*, **80**, 211–225.
24. Feldman, D.C. (1981) The multiple socialization of organization members. *Acad. Manag. Rev.*, **6**, 309–318.
25. Morrison, E.M. (2002) Newcomers' relationships: The role of social network ties during socialization. *Acad. Manag. J.*, **45**, 1149–1160.
26. Meyer, J.P., and Allen, N.. . (1988) Links between work experiences and organizational commitment during the first year of employment: A longitudinal analysis. *J. Occup. Psychol.*, **61** (3), 195–209.
27. Maier, G., and Brunstein, J.C. (2001) The role of personal work goals in newcomers' job satisfaction and organizational commitment: A longitudinal analysis. *J. Appl. Psychol.*, **86** (5), 1034–1042.
28. Klein, H.J., Fan, J., and Preacher, K.J. (2006) The effects of early socialization experiences on content mastery and outcomes: A mediational approach. *J. Vocat. Behav.*, **68**, 96–115.
29. Klein, H.J., and Weaver, N.A. (2000) The effectiveness of an organizational-level orientation training program in the socialization of new hires. *Pers. Psychol.*, **53**, 47–66.
30. Ostroff, C., and Kozlowski, S.W.J. (1993) The role of mentoring in the information gathering processes of newcomers during early organizational socialization. *J. Vocat. Behav.*, **42**, 170–183.
31. Tockey, S. (2015) Insanity, Hiring, and the Software Industry. *Computer (Long. Beach. Calif).*, **48** (11), 96–101.
32. Casalnuovo, C., Vasilescu, B., Devanbu, P., and Filkov, V. (2015) Developer Onboarding in GitHub: The Role of Prior Social Links and Language Experience. *ESEC/FSE Conf.*, 817–828.
33. Matturro, G., Fontán, C., and Raschetti, F. (2015) Soft Skills in Scrum Teams A survey of the most

valued to have by Product Owners and Scrum Masters. *SEKE 2015*, (July 2015), 42–45.

34. Fagerholm, F., Sanchez Guinea, A., Borenstein, J., and Munch, J. (2014) Onboarding in Open Source Projects. *IEEE Softw.*, **31** (6), 54–61.
35. Fagerholm, F., Guinea, A.S., Münch, J., and Borenstein, J. (2014) The role of mentoring and project characteristics for onboarding in open source software projects. *Proc. ACM-IEEE 8th Int. Symp. Softw. Engineeering Meas. - ESEM'14*, 1–10.
36. Britto, R., Smite, D., and Damm, L. (2016) Software architects in large-scale distributed projects: An Ericsson case. *IEEE Softw.*, **33** (6), 48–55.
37. Labuschagne, A., and Holmes, R. (2015) Do Onboarding Programs Work? *Proc. IEEE/ACM 12th Work. Conf. Min. Softw. Repos.*, 381–385.
38. Monasor, M.J., Vizcaíno, A., and Piattini, M. (2012) Cultural and linguistic problems in GSD: a simulator to train engineers in these issues. *J. Softw. Evol. Process*, **24** (6), 707–717.
39. Steinmacher, I., Conte, T.U., Treude, C., and Gerosa, M.A. (2016) Overcoming open source project entry barriers with a portal for newcomers. *Proc. 38th Int. Conf. Softw. Eng.*, 273–284.
40. Steinmacher, I., Graciotto Silva, M.A., Gerosa, M.A., and Redmiles, D.F. (2015) A systematic literature review on the barriers faced by newcomers to open source software projects. *Inf. Softw. Technol.*, **59** (March), 67–85.
41. Cubranic, D., Murphy, G.C., Singer, J., and Booth, K.S. (2005) Hipikat: A project memory for software development. *IEEE Trans. Softw. Eng.*, **31** (6), 446–465.
42. Malheiros, Y., Moraes, A., Trindade, C., and Meira, S. (2012) A source code recommender system to support newcomers. *Proc. IEEE 36th Int. Conf. Comput. Softw. Appl.*, 19–24.
43. Canfora, G., Di Penta, M., Oliveto, R., and Panichella, S. (2012) Who is Going to Mentor Newcomers in Open Source Projects? *Proc. ACM SIGSOFT 20th Int. Symp. Found. Softw. Eng. (FSE'12*, 44:1-44:11.
44. Steinmacher, I., and Gerosa, M.A. (2014) Choosing an appropriate task to start with in open source software communities: A hard task. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, **8658 LNCS**, 349–356.
45. Smite, D., and Van Solingen, R. (2015) What's the true hourly cost of offshoring? *IEEE Softw.*, **33** (5), 60–70.
46. Runeson, P., Höst, M., Rainer, A., and Regnell, B. (2012) *Case Study Research in Software Engineering: Guidelines and Examples*, John Wiley & Sons.
47. Robson, C., and McCartan, K. (2015) *Real World Research*, Wiley.
48. Cohen, J.A. (2006) *Best Kept Secrets of Peer Code Review: Modern Approach. practical Advice*, Smart Bear Inc.
49. Beecham, S., Baddoo, N., and Hall, T. (2008) Motivation in software engineering : a systematic literature review. *Inf. Softw. Technol.*, **50** (9), 860–878.

## APPENDIX 1 – Interview guide 1

1. Please provide your background.
2. How does the recruiting process work?
    - a. What are the characteristics that are mandatory for a candidate to be hired?
    - b. What are the characteristics that are not appreciated in new candidates?
    - c. Is the recruiting process evaluated?
3. How does the learning process work?
    - a. What are the scenarios wherein developers need to be trained?
    - b. What are the main learning needs (technical knowledge, methodological knowledge, product knowledge, soft skills)?
    - c. How are the people being trained evaluated before, during and after the learning process?
    - d. What are the learning practices/activities employed during the learning process?
    - e. In which levels does training take place (individual, group)?
    - f. How is the learning process evaluated?
4. How does the onboarding process work?
    - a. What are the scenarios wherein people are onboarded (single or group onboarding)?
    - b. What aspects are considered when forming new or restructuring existing teams?
    - c. What are the team-building activities performed?
    - d. How are the teams evaluated?
    - e. What is perceived as a mature team?
    - f. What is perceived as a stable team?
    - g. How is the onboarding process evaluated?

## APPENDIX 2 – Interview guide 2

1. Please provide your background.
2. How were you recruited?
    - a. What are the characteristics you have that were mandatory for the company to hire you?
    - b. What's your opinion about the recruiting process?
    - c. Have you ever been asked your opinion about the recruiting process? If so, please elaborate on that.
3. How were you trained?
    - a. What are the main things you had to learn to perform your work (technical knowledge, methodological knowledge, product knowledge, soft skills)?
    - b. What are the learning practices/activities employed during your learning process?
    - c. Does your learning process take place in the individual or group level?
    - d. Have you ever been evaluated before, during and after any activity during your learning process? If so, please elaborate on that.
4. How does the onboarding process work?
    - a. What are the scenarios wherein people are onboarded (single or group onboarding)?
    - b. What aspects are considered when forming new or restructuring existing teams?
    - c. Have you ever attended any team-building activity? If so, please elaborate on that.
    - d. Has your team ever been evaluated in relation to its maturity level? If so, please elaborate on that.
    - e. What is perceived as a mature team?
    - f. What is perceived as a stable team?
    - g. Has yourself or your team ever been evaluated in relation to how the associated onboarding process? If so, please elaborate on that.