Improving the Software Testing Skills of Novices during Onboarding through Social Transparency

Raphael Pham
Software Engineering Group
Leibniz Universität Hannover
Hanover, Germany
Raphael.Pham@inf.uni-hannover.de

ABSTRACT

Inexperienced software developers — for example, undergraduates entering the workforce — exhibit a lack of testing skills. They have trouble understanding and applying basic testing techniques.
These inexperienced developers are hired by software companies, where this lack of testing skills has already been recognized. Companies allocate valuable resources and invest time and money in different onboarding strategies to introduce new hires to the organization's testing practices. However, if the lack of testing skills is not addressed properly, the new hire is left to her own devices. This hinders her in becoming a high-quality engineer for the software company.
This thesis proposes to improve the onboarding strategies with traits of social transparency in order to specifically address testing issues of inexperienced new hires. Social transparency has been shown to influence the testing behavior of development teams on a social coding site. An environment that is open for discussion helps newcomers to understand and adapt a team's testing culture.
Tailoring the onboarding process to better address testing skills of new hires makes it more effective and more efficient. This reduces the danger of carrying new hire's testing deficits into commercial software development.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Programming teams

General Terms

Human Factors

Keywords

Testing, Onboarding, Social Transparency.

1. INTRODUCTION

Software Engineering is a multi-facetted discipline. Amongst other skills, the software engineer needs an understanding of the application's design, writing the actual code and testing it properly, its documentation and the overall development process — all while collaborating with other software engineers. This multitude of skills is taught in computer science lectures of educational institutions around the globe.
At Leibniz Universität Hannover, we offer a practical software development course each year. Every computer science student must take it shortly before acquiring her bachelor's degree [1]. We found that these inexperienced developers have trouble applying basic testing techniques [10]. Often systematic testing is neglected for various reasons. In these 4-months-software-projects, collaboration and team skills need to be practiced quickly. The inexperienced developer is overwhelmed by new tasks and assigns a low priority to testing. However, not applying testing techniques early in real world scenarios is problematic. Inexperienced developers misunderstand testing concepts, for example GUI automation and definition of a proper target value in non-numerical cases. Time constraints and the absence of tutorial material keep them from catching up on lack of testing knowledge. Technical barriers — such as complicated test tool chains — impede the commitment to systematic testing.
Without guidance, inexperienced developers misperceive the return on investment value of test automation efforts. They focus on efforts of maintaining a test suite during development and regard it as technical debt [10]. Such a skewed view on software testing is problematic, as these students apply for engineering positions in software companies.

Successful software companies promote a sophisticated software testing culture and award systematic testing the same importance as coding [17]. In an ongoing study, we found that software companies not only recognize the lack of testing skills in new hires, but find them severely lacking.

> *"Usually our new hires are students who has only the basic knowledge about programming. They are not aware of the whole testing stuff and how it affects the development of the product. So they do not usually understand why testing is important and as the result doesn't do it all or simply rely on the Q&A to test all the code they write."* — practitioner's quote from ongoing study.

Companies resort to different onboarding strategies: A senior developer is appointed to mentor the new hire. Specific pairing strategies for the new hire and expert developers target orthogonal skill sets and facilitate *cross-pollination*. Open environments, an open and not accusing discussion

[1]The course's structure is described in more detail in [10]

FSE'14, November 16–21, 2014, Hong Kong, China
Copyright 2014 ACM 978-1-4503-3056-5/14/11...$15.00
http://dx.doi.org/10.1145/2635868.2666604

culture and reviews by expert developers help the new hire to improve her skills. At first, the new hire is assigned to unimportant and self-contained tasks to get accustomed to the code base. Her results must be reviewed by expert developers before committing into production code.
These companies see the hiring and onboarding process as an investment - that can also fail, sinking mentoring costs. However, not all companies offer this kind of training for new hires. Some companies

> *"[...] throw them in and hope they swim :)"* — practitioner's quote from ongoing study.

Here, the new hire has to get adapt to the working culture *and* quickly smooth out her testing deficits on her own.
For new hires, it can be helpful to collaborate closely with experienced engineers and to understand their rationale behind the solutions that were applied. Collaboration between software developers has changed with the rise of social media. Software engineers connect with, provide help to, collaborate with, and learn from one another with unprecedented ease [14]. The high degree of *social transparency* [16] of social coding sites influences the development behavior of software teams that collaborate on sites such as github.com [4]. On github.com, every user has a profile and can upload their project to an online repository. Projects can be browsed by attributes and followed. Contributing to other user's project can be archived with several browser clicks. Development artifacts — such as commits — can be commented and discussed.
In previous study, we studied to what effects the testing behavior of development teams is influenced in an environment with a high social transparency. We found strategies that teams use to quickly diffuse their testing culture to mostly unknown contributors or new onboarders to the team. Clear testing signals — a badge signaling the use of a continuos integrations service or the existence of an extensive test suite — made it clear for contributors that only tested contributions would be accepted. When writing tests, contributors and onboarders heavily relied on an existing test suite. Similar tests were copied, pasted and adapted to their own needs. The open environment and social transparency traits made it easier for contributors to understand the testing culture and mimic it. This effect can be employed to specifically improve the testing skills of novices during onboarding.

2. RESEARCH PROBLEM

In our study of the testing behavior of computer science students, we uncovered fundamental constraints and problems that keep such inexperienced developers from actively practicing systematic testing techniques.
In an ongoing study, we found that this lack of testing skills of inexperienced new hires is a problem for software development companies. They feel that the subject of applying testing techniques is neglected in education, which causes a certain amount of discontent. Practical testing education is perceived as being pushed to the new hire's first commercial projects.

> *"Testing skills (and, worse, good debugging skills... but that's beside the point) are usually lacking in new hires because universities rarely equip undergrad students with the requisite software engineering skills. I believe that to be the result of viewing those skills to be in the realm of the vocational and not in the lauded theory of Computer Science. Students aren't being taught that code is considered to be complete only when it can demonstrate correctness with automated tests. As a result, newly graduated hires are oftentimes unaccustomed to writing automated tests and are rarely knowledgeable about testing technologies."*— practitioner's quote from ongoing study.

Companies have opted to specifically train new hires at the beginning of their career. This allocates resources and is costly and time-intensive. Assigning a senior developer to help the new hire or pair with her, leaves the expert developer less time for other tasks. Prolonging the training period delays the productivity phase of the new hire — although companies *do* reserve some time for training and do not expect full productivity at first.
Not taking care of the new hire's testing issues early may hinder her professional progress in becoming a high-quality software engineer for the company. If a student cannot practice testing in educational projects and the onboarding process fails to address this problem, the new hire will need to learn it on-the-fly in the next commercial project. Eventually, companies fear that this leads to sub-standard code, more failure-prone software products and the associated costs.
Some of the different onboarding strategies are more effective at targeting the lack of testing skills of new hires than others. It is still unclear whether companies specifically target testing issues that were uncovered in our testing behavior study or improve the lack of testing skills on a more general level. The amount of effort put into a new hire's testing issues seems dependent on different attributes, such as the company's size or the prevailing testing culture. These strategies and related environmental factors need to be studied further.
The goal of this thesis is to provide tailored means to systematically improve the onboarding process of inexperienced new hires regarding the lack of testing skills through the use of social transparency. With this goal in mind, several research questions need to be considered:

1. How do software companies currently cope with such testing issues?
2. Which strategies work well and which do not and why?
3. How can social transparency and its influence on testing behavior be used to help companies with the testing skill problems of new hires?

This thesis focusses on the onboarding phase for inexperienced developers with a regard on testing — and not the educational situation of undergraduates before entering the workforce. Universities apply different education models and some fare better than others. However, the industry's requirements are not met yet (c.f. Section 3). Solely improving the educational situation is no longer sufficient and does not reflect the joint responsibility of transitioning undergraduates from education to commercial development. Here, the complexity of actual industrial software testing demands better — and *tailored* — onboarding techniques. Ultimately, better understanding the onboarding process regarding testing can improve undergraduates' testing training.

3. RELATED WORK

Despite different proposals[2] to improve the testing education, our ongoing study shows that industry's needs for testing skills in new hires are not met. In our ongoing online questionnaire, 51% of 72 practitioner respondents are dissatisfied with applicants' testing skills, while only 38% are satisfied. The rest answered in text, mostly negatively. Our sample consists of self-chosen practitioners working at companies around the world. These new hires come from a wide variety of educational institutions worldwide.

Brechner, Director of Developer Training at Microsoft, identified five areas in computer science education, that need improvement in order to meet industry needs. In more detail, these include high testability for the software product and use of automated tests [3]. Assessing current testing practices [8], practitioners describe university training in software testing as "widely second-rate" and note that most testers are self-taught. The absence of a universally agreed-on terminology and testing body of knowledge is criticized. Ng et al. [9] conducted a survey about testing practices in Australia and arrived at a similar conclusion. However, not all freshly graduated newcomers exhibit a lack of testing skills. Begel et al. [2] shadowed 8 graduate students who just began working at Microsoft. The observed newcomers showed good efforts in testing their application and even wrote automated tests. As mentioned earlier, we suspect that certain company attributes influence the testing skill level of applicants who may enter the onboarding process. Microsoft has a sophisticated testing culture and is a large company. It may be that Microsoft has a high hiring bar, thus rejecting applicants with low testing skills.

The onboarding phase is difficult for the newcomer in several ways, not only regarding testing issues. Begel et al. [2] highlight five areas in which new hires often exhibit problems: communication, collaboration, technical, cognition, orientation. When to ask whom was a big issue for the new hires and often hindered their progress.

Steinmacher et al. [13] studied the onboarding process in an open source development setting. They describe a catalogue of barriers for onboarders of open source projects. Newcomers have trouble finding mentors for support.

However, active support while onboarding helps newcomers to contribute to the project more actively. Fagerholm et al. [6] hypothesizes that developers, who were supported during their onboarding phase, have a higher chance to "be exposed to, select, and perform tasks in a proactive and self-directed manner". Supporting onboarding with mentoring seems to be most influential in this regard.

The benefits and caveats of mentoring have been assessed by several practitioners. Sim et al. [12] deem it inefficient and disruptive, yet very effective in conveying new information to onboarders. The mentor cannot complete other tasks [5] and the team's productivity suffers. Begel et al. [1] advocate mentoring, as the mentor acts as a role model from whom the novice can learn. In our ongoing study, mentoring seems to be the go-to choice for onboarding strategies. However, we see potential to make it more efficient by improving it with social transparency traits. Thus, enabling the new hire to find some answers or better-suited colleagues on her own and lessening the load on the mentor.

[2]Due to the 4 page limit, these proposals cannot be presented here, but can be handed in on demand.

The onboarding process in general in open source and commercial development has been the focus of many researchers. However, a focus on testing skills is still missing.

4. PROPOSED SOLUTION

The main contribution of this thesis will be the following: *The recognition of the influences of social transparency on systematic software testing and its targeted transferral onto testing issues in onboarding processes of software companies.* One result of this thesis is an actionable catalogue for improving the onboarding experience. It outlines different company environments, the testing problem area of the new hire and proposes an improvement method that employs social transparency mechanisms.

So far, we have described the issues that computer science students have with systematic testing. Currently, we are analyzing software companies take on these issues: Do they see this lack of testing skills in new hires? What do they do about it? Preliminary results show that companies *do* have problems with applicants' testing skills.

We have intensively studied the testing behavior of software development teams on a social coding site. This study yielded insights into the effects and influences that social transparency has on testing behavior of developers. We understood how a testing culture is adopted by new team members and how testing knowledge is diffused.

I want to use this knowledge of the effects of social transparency to improve testing related problems that new hires bring into the onboarding phase. Therefore, I will outline onboarding strategies and analyze which ones are successful and why — and under which circumstances. I will use my knowledge about the effects social transparency on testing behavior to adapt the successful strategies and enrich them with social transparency mechanisms. This will result in the aforementioned catalogue of environments, issues and methods. Companies can use this catalogue to tackle specific testing knowledge needs of their new hires according to the environment they provide.

In all three studies mentioned above, a grounded theory approach was used [7, 15]. Each time, a general area of interest was defined and the appropriate population was addressed: computer science students who were about to acquire their bachelor's degree (testing problems of undergraduates [10]), active users of the social coding site GitHub.com (testing culture on a social coding site [11]) and software development practitioners (testing issues in onboarding). Except for the first study with students, open source and commercial practitioners were self-chosen and came from all over the world, working in projects of differing sizes and criticality.

Every population was repeatedly interviewed in a semi--structured manner and a theory of their most pressing problems was build incrementally. We validated the core elements of our testing culture theory [11] through sending an online questionnaire to another set of active github users and asking them to verify or reject our findings. We plan on validating the findings of our current study about testing issues in onboarding in the same way.

After three grounded theory studies, I will have collected information about the problem space and a possible solution space: testing problems of inexperienced developers, industry's take on these problems and social transparency mechanisms. The next step is the transferral of those social transparency mechanisms onto onboarding strategies.

It starts as soon as we have gained insights into the current onboarding strategies. I will evaluate some of my proposed methods with a population of inexperienced developers.

5. EVALUATION

This thesis deals with two populations: inexperienced developers and software companies. Ideally, I can cooperate with a software company and improve their onboarding process. I would assess the testing issues and environment of this company, apply an improvement method and measure the impact of my changes. However, measuring the impact of my improvements in numerical values appears difficult: The assessment of testing skills and testing issues is multifacetted and non-numerical. Therefore, a survey or questionnaire with closed questions is inappropriate and rather leading. In order to capture such qualitative traits, I will use an adapted grounded theory approach: During the first stage, I will interview members of the company in an open manner and focus on the testing issues with inexperienced onboarders. I will then apply my proposed improvements to the onboarding process. An inexperienced developer will then be onboarded this way. Eventually, I will re-interview company developers. This will result in a comprehensive theory about the impacts of my changes.
Additionally, I would interview the new hire and inquire about the perceived impact of the social transparency traits of the new onboarding process. This could — for example — include questions about whether or not the new hire accessed other developer's written tests or *followed* discussions and how that helped her. Alternatively, I could shadow the new hire altogether and document her progress and problems. Her progress would then be compared to other new hires who completed the normal onboarding.

6. PUBLICATIONS

Testing and Social Transparency [11]: Grounded theory study, conducted in August 2012. Investigated influence of social transparency on testing behavior of software teams on social coding site. Conducted 33 interviews with active github.com users, several online questionnaires; yielded 569 questionnaire answers. Presented at ICSE 2013.
Testing Issues of Inexperienced Developers [10]: Grounded theory study, conducted in January 2014. Uncovered problems and constraints of systematic testing for computer science students (just before receiving bachelor's degree). Interviewed 97 students who had developed software product for four months. To be presented at FSE 2014.
Testing Issues in Industrial Onboarding Processes Ongoing grounded theory study, finished by November 2014. We want to understand if software companies recognize a lack of testing skills in inexperienced new hires and what they do about it.

7. ACKNOWLEDGEMENTS

Prof. Schneider, head of the Software Engineering Group at Leibniz Universität Hannover, is supervising this thesis.

8. REFERENCES

[1] A. Begel and B. Simon. Novice software developers, all over again. In *Proc. of the Fourth Int. Workshop on Computing Education Research*, pages 3–14. ACM, 2008.

[2] A. Begel and B. Simon. Struggles of new college graduates in their first software development job. In *ACM SIGCSE Bulletin*, volume 40, pages 226–230. ACM, 2008.

[3] E. Brechner. Things they would not teach me of in college: what microsoft developers learn later. In *Companion of the 18th annual ACM SIGPLAN Conf. OOPLSA*, pages 134–136. ACM, 2003.

[4] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in github: transparency and collaboration in an open software repository, 2012.

[5] T. DeMarco and T. Lister. Peopleware: Productive people and teams. *Dorseth House*, 1987.

[6] F. Fagerholm, P. Johnson, A. S. Guinea, J. Borenstein, and J. Munch. Onboarding in open source software projects: A preliminary analysis. In *Global Software Engineering Workshops (ICGSEW), 2013 IEEE 8th Int. Conf. on*, pages 5–10. IEEE, 2013.

[7] B. Glaser and A. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Observations (Chicago, Ill.). Aldine de Gruyter, 1967.

[8] R. L. Glass, R. Collard, A. Bertolino, J. Bach, and C. Kaner. Software testing and industry needs. *IEEE Software*, 23(4):55–57, 2006.

[9] S. Ng, T. Murnane, K. Reed, D. Grant, and T. Chen. A preliminary survey on software testing practices in australia. In *Software Engineering Conf., 2004. Proc.. 2004 Australian*, pages 116–125. IEEE, 2004.

[10] R. Pham, S. Kiesling, O. Liskin, L. Singer, and K. Schneider. Enablers, Inhibitors, and Perceptions of Testing in Novice Software Teams. In *to appear: 22th Int. Symposium on the Foundations of Software Engineering (FSE 2014), Hong Kong, China*, 2014.

[11] R. Pham, L. Singer, O. Liskin, F. Figueira Filho, and K. Schneider. Creating a shared understanding of testing culture on a social coding site. In *Int. Conf. on Software Engineering (ICSE)*, pages 112–121. IEEE Press, 2013.

[12] S. E. Sim and R. C. Holt. The ramp-up problem in software projects: A case study of how software immigrants naturalize. In *Software Engineering, 1998. Proc. of the Int. Conf. on*, pages 361–370. IEEE, 1998.

[13] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, and D. Redmiles. The hard life of open source software project newcomers. In *Proc. of the 7th Int. Workshop on Cooperative and Human Aspects of Software Engineering*, pages 72–78. ACM, 2014.

[14] M. Storey, C. Treude, A. van Deursen, and L. Cheng. The impact of social media on software engineering practices and tools, 2010.

[15] A. Strauss and J. Corbin. *Grounded Theory in Practice*. SAGE Publications, 1997.

[16] H. C. Stuart, L. Dabbish, S. Kiesler, P. Kinnaird, and R. Kang. Social transparency in networked information exchange: a theoretical framework, 2012.

[17] J. A. Whittaker, J. Arbon, and J. Carollo. *How Google tests software*. Addison-Wesley, 2012.