Designing Engineering Onboarding for 60+ Nationalities

Julian Harty
Commercetest Ltd/The Open University
High Wycombe, U.K.
julianharty@gmail.com/julian.harty@open.ac.uk

ABSTRACT

A large international engineering office in Germany needed to double in size in 12 months. We designed an onboarding programme within 3 months to help it do so efficaciously. We wanted to optimize for: fast iterations in the programme rollout, to keep the 'flywheel spinning' by reducing drag on current staff, rapid acceleration where new hires contributed quickly, and smooth integration where new hires adapted to the engineering, company, and country cultures.

To reduce drag we onboarded in cohorts and involved existing practitioners in the design and discussion. To encourage contributions quickly we built contributions into the sessions, we also streamlined IT Support. To help new hires adopt the culture we encouraged help and mentoring within and across cohorts.

For fast iterations, we incorporated existing islands of onboarding, involved local technical staff in design and delivery of hands-on training, and applied analytics to help improve the practice. And we launched early to bootstrap our learning and evaluation.

Our approach worked; new hires were able to make meaningful contributions within a week and they scored the onboarding programme positively (8.5 NPS).

CCS CONCEPTS

• **Software and its engineering → Collaboration in software development**; • **Applied computing → Collaborative learning**.

KEYWORDS

Collaborative learning, Collaboration in software development, Mob programming, Onboarding

ACM Reference Format:
Julian Harty. 2020. Designing Engineering Onboarding for 60+ Nationalities. In *15th IEEE/ACM International Conference on Global Software Engineering (ICGSE '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3372787.3390504

1 INTRODUCTION

An international company needed a structured onboarding programme that enabled new hires to become productive, engaged with the ethics and vision of the company, while reducing drag on current employees and preserving the positive aspects of the company and culture. The programme had to work at scale and be effective as the workforce doubled and cope with growth of 200+ staff per calendar month across engineering globally.

The company had thousands of people working in technology and wished to design a structured onboarding programme that would scale and work well even if the company doubled the staff in the technology organisation within 12 months. New hires in both engineering locations had taken months to become sufficiently productive to provide a return on the cost and effort of recruiting, hiring and paying them while they were getting up to speed.

Business objectives were to onboard newly hired technology staff from people starting their career to experienced director level hires. We needed to reduce both the drag and the elapsed time needed to enable people to become competent team members who would be able to contribute productively *even while the team was still growing and acquiring more recently hired members.* (As Fred Brooks [7] indicated, adding people is hard to do productively.)

The author of this paper was recruited to lead and help design and establish the onboarding programme for the German Office and transition the operations to a newly hired manager.

New hires had to work in a global environment in three senses of the concept of global:

1. nearly all the new hires came from different countries, climates, cultural expectations, and work environments (they also spoke tens of languages natively);
2. they needed to acclimatise to working in Germany where the official language is German and they needed to adapt to the cultural and legal aspects of working in a foreign land; and
3. the technology organisation was split between the USA and Europe so they needed to be able to work effectively across regions and timezones.

In essence they needed to learn how to work with each other in a global melting pot and be effective.

The company had learned the importance of onboarding new hires and had already had significant success recruiting for their main US engineering office. They wanted to develop ways to improve the efficacy of their onboarding into the engineering organisation in both the USA and Germany. It also needed to scale so it would be viable for hiring an average of 200 staff per calendar month, and also handle larger peaks of engineering interns who were onboarded once or twice a year.

They estimated 4 to 6 months were needed for a new hire to be net productive, and there were significant costs to the established staff who had to devote substantial effort to integrating recently hired team members.

ICGSE '20, October 5–6, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7093-6/20/05…$15.00
https://doi.org/10.1145/3372787.3390504

We designed the onboarding program to help new hires learn how to work with the extensive codebases (including technical debt and areas of radioactivity where the code was dangerous to approach unprotected or unprepared). For many recent joiners the engineering was at least ten times the size of anywhere they had worked at previously, and for many it was over 100 times the size and complexity.

2 BACKGROUND TO THE PROJECT

The technology organisation described in this paper is an intrinsic part of an international online product sales business with over 10,000 employees. They recruit globally to hire software engineers and related roles including DBAs, DevOps, security engineers, and so on. They have two primary engineering centres, in the USA and in Germany. The German team was established for various business reasons, including supporting planned business expansion into Europe. Reporting lines and engineering teams had hybrid reporting lines, both to the local Head of Engineering and also to respective senior managers in the USA. They had successfully recruited people from over 60 countries at the start of the programme.

The company hires people with diverse backgrounds and varied experience and expertise into the technology organisation. In Germany they recruited mainly experienced staff from across the globe. In the USA they hired experienced people from across the USA and also fresh graduates of software development bootcamps, who often lived nearby.

Several years before this project the company had established a three-month programme. It was designed to bring non-traditional new software developers, including those from bootcamps, up to speed in commercial software development based on the company's codebases. It included two distinct one-month projects to help the new programmers learn how to work in key codebases. This was perceived as so useful they wished to explore ways to expand the programme in two dimensions: 1) internationally *i.e.* in Europe, and 2) to include a wider range of experiences, including more senior hires who had prior experience in technology and software development and managers up to director level.

3 RELATED WORK

3.1 Global working

Researchers have realised the importance and necessity of software engineers working in a global environment. For instance in an ACM article in 2017 [4] the authors discussed ways to help students learn how to work in a global environment. The article covers some of these topics e.g. working asynchronously across timezones, albeit in an academic microcosm which lacks the richness, history and challenges of running a multi-billion USD business. Research has also explored some of the other challenges of working globally, such as the effects on team members working in remote 'agile' teams [5], and in ways to best distribute work across international teams [15]. The technology organisation needed to consider these factors generally and address them in the onboarding materials.

3.2 Approaches to Learning and Practicing

From a business leader's perspective, the first 90 days are considered crucial while the newcomer has to: make sense of the organisation, work out how they can be effective, and start delivering significant value [23]. In effect, new recruits to a technology organisation may have to achieve similar results in their domain. Therefore it would be in everyone's interests for new hires to ramp up quickly through effective onboarding.

We considered several concepts including Grovo's concept of Microlearning®, a series of short lessons only several minutes long, often delivered by video [12]. Grovo provided a short 'create a new lesson' microlearning lesson that proposes three steps 1) introduce a topic, 2) explain a topic, 3) teach a skill. In [13] Grovo encourage showcasing the company's culture and values, and finding out *"what kinds of onboarding activities might they enjoy most"*. However their material was in orthogonal areas and outside the scope of the onboarding we designed so we did not end up using their material.

The work of Edwards, on using software testing to move students from trial and error to reflection in practice [10] influenced the design of the onboarding; where recent hires (and sometimes also longer serving employees) would change their perspective from the immediate problem and how to 'fix' the problem, to improving how they understood, developed and tested the software they were working on.

Various books in the education domain, including *'Making Every Lesson Count'* [1], and in the personal improvement domain, such as *'The Practicing Mind'* [20] and *'Peak Performance'* [21] influenced the concepts we wished to foster in the new hires. In [1] the author proposed an approach of: challenge, explanation, modelling, practice, feedback, questioning, and embedding the ethos.

According to an article in HBR [14] the authors recommend a *'Learning Loop'* of four stages: *'gain knowledge; practice by applying that knowledge; get feedback; and reflect on what has been learned.'*. Our peer-led learning helps small teams to apply these concepts.

3.3 Pair and Mob programming

Mob programming has been studied in several contexts. As we discovered, it can help less experienced programmers get up to speed [6]. And a long-term study of one team using mob programming over 18 months identified various benefits including: more consistent coding and use of tools, and increased the understanding of, and confidence in, working with the codebase [8].

3.4 Onboarding

Onboarding is well described in a clear, well researched article [3]; for instance, which identifies key concepts including: "working mastery", "interactions", and the concepts of "outsiders" and "insiders". Newcomers (outsiders), insiders, and potentially the organisation, all adapt to varying degrees throughout the onboarding process. As an example, onboarding of a new CEO may have a greater impact on the organisation than an intern. Our onboarding programme included even senior hires - directors - to help them spend time gaining experience with the real, live production codebase to help ground them in organisational reality.

Onboarding is a hot topic in business circles, with several fast growing young global businesses investing in long, rigorous onboarding [17]. Several businesses identified the value of longer onboarding of up to a year, and focus on" *"three key dimensions: the organizational, the technical, and the social"* [9]. Some teach specific skills, for instance, Booking.com ensure employees learn about experimentation [22].

Early work in 1998 described onboarding experiences of four recently hired software developers for a team who created a software compiler in an aptly titled paper on *"... how software immigrants naturalize"* [18]. Despite a 21-year gap in our respective experiences there were several aspects in common including the relevance of what is needed to become productive software developers. One of the challenges the paper identified was that mentoring alone was an inefficient use of time *"... because it [mentoring] results in a net decrease in team productivity in the short term."*. We chose to learn in groups and as groups (cohorts) to help reduce the drag on the current teams.

There were three main stages to the assignment: 1) scoping the needs for the desired onboarding programme 2) designing the programme 3) performing iterative micro-pilots to learn and refine through experience.

4 SCOPING THE NEEDS

Scoping took a couple of weeks and included the discovery work.

4.1 Understanding the needs

The author visited the headquarters in the USA for a week to experience aspects their onboarding programme, before working on site in Germany for just under 3 months. The author attended the existing onboarding in the two main engineering locations, first in the USA at the headquarters, then two weeks later in Germany, as if the author was a newly hired software engineer. In the USA the onboarding programme was 2 $\frac{1}{2}$ days for non managers with an extra $\frac{1}{2}$ day for managers. In Germany it was just 1 day long.

4.2 Discovering existing practices and results

In both locations a wide range of people were interviewed; ranging from the technical co-founder to recently hired bootcamp graduates, to obtain a broad range of perspectives and views on the current onboarding and potential objectives and approaches for any future programme. There were several common themes that emerged from these interviews:

- Everyone was keen to improve the onboarding for new hires, and to extend it so more senior technology would also take part in extended onboarding of at least a week. One aim was for new hires to 'touch' real code and understand the practical aspects of delivering software in whichever group or system they would join.
- The business was willing to invest up to several months in new hires to help them settle in and become productive and motivated employees.
- It took days and sometimes weeks for new hires to be setup with working computers that had access to the relevant source code repositories, permissions, network connections, etc. IT support was split into pre- and post- hire teams where pre-hire performed provisioning of accounts, access, permissions and computers and post-hire dealt with whatever was needed once a new hire started. The two behaved as a split brain with little communication or collaboration between them which lead to a lack of corporate learning and improvements were infrequent.

We realised the current three-month onboarding programme used in the USA for junior programmers was not very relevant for the German office. In particular, the vast majority of new hires for the German office already had significant commercial software development experience; therefore we agreed to devise a fresh approach that would suit these engineers.

5 DESIGNING THE ONBOARDING

We established the onboarding programme needed to incorporate and align three complementary facets: 1) the company ethos and culture we wished to engender, 2) establishing healthy and productive working practices, and 3) the technology and software skills that new hires would need to become proficient in. We also wanted to remove as many roadblocks as practical; for instance to provide computers pre-configured so they could be used productively to work with the codebases (and/or whatever the individual needed for their role such as database access for DBAs) on their first day of employment.

We made a conscious choice to encourage those who were more able to help those who were still learning and getting up to speed. For example, if someone was already proficient in a topic and able to complete an exercise quickly was encouraged to work alongside someone who was in their early stages of learning.

5.1 Three complementary threads

To be effective our onboarding needed to address three complementary threads: competence, collaboration, and culture. New hires needed to become competent in their role in terms of technical skills, of course. Equally relevant are competence in working practices, by learning to collaborate within their team, with other teams, and across timezones. They also needed to internalise and apply the ethos of the engineering organisation.

5.2 Building on existing, successful practices

We discovered one manager had developed a well defined and structured onboarding for his team where a post onboarded employee was paired with a newly hired employee. The existing employee was responsible for actively mentoring and working with the recent hire for up to three months. The manager controlled and revised the training iteratively based on the experiences with each of the new hires. As a result, the onboarded hires appeared to integrate well with their teams.

Another team had a self-taught programmer (who originally worked in far removed jobs) who devised several learning modules. These modules were to help small groups learn various programming languages and techniques such as unit testing. Groups of peers self-formed. One of the peers led a short prepared lesson (illustrated in Figure1). Their work was influenced by two books in particular, *The Practicing Mind* [20], and *Peak Performance* [21] where the authors focus on ways to improve one's performance. The learning became popular and grew both across the technology organisation and within the team who ended up creating a group of related lessons on several programming topics.

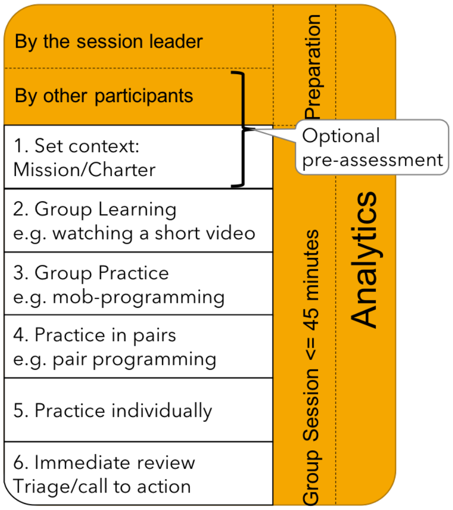

Figure 1: Lesson Structure

There were three distinct aspects to each lesson: 1) design, 2) preparation by the peer leading the session (and sometimes also by the rest of the peers), and 3) the actual 45 minute session together as a group. This concept was codified and enhanced, for instance to set the context for each lesson by adding a mission or charter (concepts introduced by Jonathan Bach in Session-Based Test Management [2] which proved effective in helping establish credibility for the concept of Exploratory Testing).

We also developed a guide for new lesson contributors, with acceptance criteria, that explained the underlying concepts *e.g.* that everyone involved can learn from being involved in the lesson, and practical aspects (such as ensuring external materials allowed use for training and learning in corporate, group settings).

5.3 Cohorts

Where practical, new hires were arranged to start on a fortnightly basis i.e. every 2 weeks. This meant that for a three-month onboarding process there would be roughly six groups of new hires within the onboarding process of any particular person being onboarded. We realised their experiences could help more recently hired employees *and also help them recognise they were learning and could contribute effectively even while they were getting up to speed.* Furthermore, by involving and engaging the people being onboarded the programme didn't solely rely on the current engineering base which reduced the drag on the teams who had to support the business, the projects, etc.

6 ITERATIVE PILOTS

As new hires were hired on an ongoing basis, we decided it was better to grow the onboarding iteratively too. We started before we believed we were "ready" in order to maximise learning and reduce the risk of developing irrelevant or unproductive content. In parallel a team in the US office were developing a three-day build-test-deploy course that was expected to be ready within a month of our launch. Their material was being reviewed by engineers in Germany and meanwhile they were aware of our pilot courses so we could learn from each other and adapt the overall work.

Key elements in the pilots: We chose to integrate our pilot course so it immediately followed the corporate onboarding, albeit the new starters would receive correctly configured machines capable of being used for development on their first day. The mainstream onboarding covered various aspects of being an employee in Germany (including payroll, health insurance, workers rights, etc. registering with the local authorities.) welcome to the company and to the office, getting started with their desktop or laptop computer, *etc.*

The engineering onboarding started with a welcome from senior engineering leadership to set the context for the rest of the onboarding. We used Slack (a very popular collaborative communications tool [11]) with several 'channels', including one for the organisers of the onboarding and another shared across everyone involved.

The rest of the first day covered foundational topics: setting-up and testing Virtual Machines for development, and a mob programming session on arrays in JavaScript. The second day included: setting up the development IDE for php programming, VPN connectivity and learning how to access various key database servers.

7 RESULTS

7.1 Initial results

Early feedback and results were qualitatively positive, both mentors (existing post-onboarded staff) and new hires rated the pilot onboarding sessions extremely well. Furthermore the mentors started to establish healthy and supportive working relationships with the recent hires. One of the mentors developed ad-hoc training material within days to explain the various main system databases that many of the developers would need to work with.

7.2 Updates, 13 months later

The initial engagement finished in January 2019, during the pilot phase of the implementation of the onboarding programme. An interview with the Head of Engineering at the company in Germany in February 2020 provides a useful perspective on the maturation of the onboarding programme. By February 2020 the duration of the onboarding had changed slightly, corporate onboarding was much improved and now comprises the first 2 days; immediately followed by 3 days to cover build, test and deploy activities for the core php codebase. This happens in a dedicated classroom environment in the first week of their employment.

The technical training consists of a mix of pairing and groups of up to 5 people working collaboratively. At the end of the week participants completed a Net Promoter Score (NPS) questionnaire, the mean score is 8.5 and positive[1]. From the organisation's perspective the main aims of the onboarding are: establishing supportive cohorts of highly connected people and everyone understanding how to build, test and deploy through first-hand, personal experience.

Onboarding occurs every two weeks. Everyone hired into the technology organisation, up to and including directors, completes the training. During this period the technology organisation in Germany grew by over 50%, a net increase of over 110 people. The new hires have been absorbed effectively into their various teams;

[1] Discussions about flaws and alternatives to NPS include [19]

and overall the onboarding programme is considered to be useful and valuable by participants, existing staff, and the company.

8 DISCUSSION AND LESSONS LEARNT

Harvard Business Review identifies several of the concepts we applied as being valuable to businesses and other organisations. In [14] they discuss ways employees can learn effectively from each other using what they call a *learning loop* of four stages: 1) gain knowledge, 2) practice by applying that knowledge, 3) get feedback, and 4) reflect on what has been learned.

We learned the organisation already had some isolated, local, positive practices where the original team actively helped us incorporate their work in the onboarding programme. We did not threaten their approach, instead we nurtured and encouraged it.

We discovered disconnects that had a major adverse effect on providing new hires with correctly configured computers, delays of several weeks were common. We connected IT support staff so pre-hire could learn from what happened post-hire. We also encouraged IT support to provide a dashboard and tracking of new hires from the hiring request in the system until the new hire was correctly provisioned with a suitable computer and working environment.

Launching lightweight, onboarding pilots provided useful insights which enabled the team to improve the onboarding quickly and iteratively. We also piloted material developed predominantly in the parent location first, so they received feedback to help them improve that larger body of course material. We discovered previously unknown talents; for instance a programmer originally from Egypt who has a flair for engaging new hires and helping them help each other to learn and solve collaboratively.

Onboarding is insufficient by itself, integration into the teams and organisation is also important and an area worth considering for future research [16]. Our work delivered material improvements through an intensive 1-week programme; far less than other industry leaders. We are still keen to learn what optimal onboarding looks like and what results it could deliver.

9 CONCLUSION

We designed a structured onboarding program for newly hired developers and technologists that enabled new hires to work collaboratively and productively within one week of joining the company. This program was able to scale to cope with ambitious goals: the organisation wished to double their engineering organisation within 12 months with a particular focus of growing the organisation in Germany to serve the European market.

The onboarding helped foster collaborative working across engineering disciplines and across seniorities where a director might be learning a technical skill with a junior programmer. We identified and applied good practices found in pockets of the engineering organisation. Examples include peer led training 1 to 1 within the DBA team to peer-led short 45 minute mob-programming sessions on a series of related exercises.

As a single case study for a particular organisation our approach has not been proven elsewhere yet. We hope others will find valuable aspects that work for them, please contact the author with your experiences of onboarding.

ACKNOWLEDGMENTS

Tony Clear shepherded me through writing this paper. Isabel Evans, Joseph Reeve and Yijun Yu also suggested improvements.

REFERENCES

[1] Shaun Allison, Andy Tharby, and Doug Lemov. 2015. *Making Every Lesson Count: Six principles to support great teaching and learning.* Crown House Publishing, Wales, UK.

[2] Jonathan Bach. 2000. Session-based test management. *Software Testing and Quality Engineering Magazine* 2, 6 (2000), 32–37. https://www.satisfice.com/download/session-based-test-management

[3] Talya N Bauer and Berrin Erdogan. 2011. Organizational socialization: The effective onboarding of new employees. *APA handbook of industrial and organizational psychology, Vol 3: Maintaining, expanding, and contracting the organization.* 3 (2011), 51–64.

[4] Sarah Beecham, Tony Clear, John Barr, Mats Daniels, Michael Oudshoorn, and John Noll. 2017. Preparing tomorrow's software engineers for work in a global environment. *IEEE Software* 34, 1 (2017), 9–12.

[5] Pernille Bjørn, Anne-Marie Søderberg, and S Krishna. 2019. Translocality in global software development: The dark side of global agile. *Human–Computer Interaction* 34, 2 (2019), 174–203.

[6] Karel Boekhout. 2016. Mob programming: Find fun faster. In *International Conference on Agile Software Development.* Springer, Springer, Edinburgh, UK, 185–192.

[7] Frederick P Brooks Jr. 1995. *The mythical man-month (anniversary ed.).* Addison-Wesley Longman Publishing Co., Inc., USA.

[8] Jim Buchan and Mark Pearl. 2018. Leveraging the mob mentality: an experience report on mob programming. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018.* ACM, Christchurch New Zealand, 199–204.

[9] Ron Carucci. 2018. *To Retain New Hires, Spend More Time Onboarding Them.* HBR. https://hbr.org/2018/12/to-retain-new-hires-spend-more-time-onboarding-them

[10] Stephen H Edwards. 2004. Using software testing to move students from trial-and-error to reflection-in-action. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education.* ACM, Norfolk Virginia USA, 26–30.

[11] Brian Elliott. 2019. *Not all Daily Active Users are created equal: Work is fueled by true engagement.* Slack Technologies Ltd. https://slackhq.com/work-is-fueled-by-true-engagement

[12] Grovo. 2016. *What is Microlearning? - Grovo.* Grovo. https://blog.grovo.com/microlearning/

[13] Grovo. 2020. *Create a Great First Day For New Hires.* Grovo. https://www.grovo.com/lessons/create-a-great-first-day-for-new-hires

[14] David Blake Kelly Palmer. 2018. *How to Help Your Employees Learn from Each Other.* HBR. https://hbr.org/2018/11/how-to-help-your-employees-learn-from-each-other

[15] Sajjad Mahmood, Sajid Anwer, Mahmood Niazi, Mohammad Alshayeb, and Ita Richardson. 2017. Key factors that influence task allocation in global software development. *Information and Software Technology* 91 (2017), 102–122.

[16] Michael D. Watkins Mark Byford and Lena Triantogiannis. 2017. *Onboarding Isn't Enough.* HBR. https://hbr.org/2017/05/onboarding-isnt-enough

[17] Daniel McGinn. 2020. The Power of These Techniques Is Only Getting Stronger. *HBR* 98 (2020), 54–55. Issue 2. https://hbr.org/2020/03/productive-innovation#the-power-of-these-techniques-is-only-getting-stronger

[18] Susan Elliott Sim and Richard C Holt. 1998. The ramp-up problem in software projects: A case study of how software immigrants naturalize. In *Proceedings of the 20th international conference on Software engineering.* IEEE, IEEE, Kyoto, Japan, 361–370.

[19] Christina Stahlkopf. 2019. *Where Net Promoter Score Goes Wrong.* HBR. https://hbr.org/2019/10/where-net-promoter-score-goes-wrong

[20] Thomas M Sterner. 2012. *The Practicing Mind: Developing Focus and Discipline in Your Life Master Any Skill or Challenge by Learning to Love the Process.* New World Library, Novato, CA, USA.

[21] Brad Stulberg and Steve Magness. 2017. *Peak Performance: Elevate Your Game, Avoid Burnout, and Thrive with the New Science of Success.* Rodale, New York, NY, USA.

[22] Stefan Thomke. 2020. Building a Culture of Experimentation. *HBR* 98 (2020), 40–48. Issue 2. https://hbr.org/2020/03/productive-innovation#building-a-culture-of-experimentation

[23] Michael Watkins. 2013. *The first 90 days, updated and expanded: proven strategies for getting up to speed faster and smarter.* Harvard Business Review Press, USA.