Onboarding Tasks

Effective Onboarding in Software Development Teams

Tim Wellhausen
kontakt@tim-wellhausen.de
https://www.tim-wellhausen.de

v1.0, October 20, 2024

Onboarding a new developer to an existing software development team is a process that aims to introduce them to and familiarize them with the project, the team, and their expected role in the team. This process poses several challenges: the new colleague must familiarize themselves with the specific business needs of the software system, adapt to the existing code base and technical infrastructure, and ultimately integrate personally into the team.

Software development projects are often carried out in small teams. Sometimes, teams are co-located, while at other times, team members are distributed across multiple places. For an efficient and enjoyable collaboration, it is important that all team members know and respect each other.

While extensive resources are available on onboarding new employees at a company-level, less guidance can be found on onboarding new team members to software development projects, in particular in form of patterns ([1], [2]). In this paper, I present the concept of Onboarding Tasks, a team effort to create a series of tasks that facilitates their learning of the software system and fostering connections within the team.

The Onboarding Tasks pattern is derived from personal experience in multiple projects where it was applied in various forms. This paper aims to distill all lessons learned from those projects into a cohesive form.

The Onboarding Tasks Pattern

As a team, you are working on a software project. The software system you are developing is complex, with incomplete technical documentation and a fragmented understanding of business requirements.

A new team member is about to join your team.

You want to welcome and integrate the new member and make them an effective and fully integrated team player as quickly as possible. However, the existing team is busy working on their own tasks, leaving little time to connect with the new member. Additionally, it is difficult to find proper tasks that the new member could work on to systematically learn the system under development.

There are two different aspects to this problem: how to integrate the new member into the team on a personal level and how to help them learn the software system.

Integrating a new member into an existing team is often difficult for a number of reasons:

- For a productive working setting, it is essential that trust is established between all team members, but for this, you need to get to know each other well.
- All team members might be willing to help with onboarding the new colleague, but project constraints could hinder the team from devoting sufficient time and energy to the onboarding process.
- Face-to-face communication is very helpful for building relationships, but in a distributed team, members have limited opportunities to encounter each other spontaneously.
- The new member might want to reach out to all existing team members, but some of them might be reluctant to get to know the new colleague.

- The existing team members might try to reach out to the new colleague, but the new member might hesitate to engage with other people.

At the same time, learning the software system is also a challenge for the new colleague:

- The new team member might have strong technical skills in general, but may not know all the products and libraries in use.
- The new member might have strong coding skills, but they also need to learn sufficient details about the project's technical infrastructure.
- The software system might be a publicly available application for end users, but the new colleague might lack familiarity with it.
- The new member might be willing to quickly take over development tasks, but gaining a good overview of the code base takes a considerable amount of time, and for a new team member, it is challenging to determine where to begin.
- For many developers, it is more rewarding to work on code than to read code or documentation, but suitable real development tasks might be rare.
- Tasks to fix bugs are typically good candidates to start with, but you cannot rely on the existence of such bugs at the time the new member joins.

* * *

Therefore:

As a team, write up a series of exploratory onboarding tasks for new team members to help them familiarize themselves with the system. Each task should focus on a specific technical or business issue of the software that a new member should learn about. Allow time for them to work on each task and, at the conclusion of every task, encourage them to discuss their learnings with other team members.

Writing proper onboarding tasks is a time-consuming effort that cannot be done spontaneously the day before the new member joins the team. Rather, treat these tasks as a team asset that you start building at some point to have them ready when the next colleague joins. Such a collection of tasks grows over time but needs constant attention and refinement. By regularly revising and updating the tasks, you should be able to reuse the tasks in the future.

It should be the responsibility of all team members to write and revise onboarding tasks. During the preparation of onboarding tasks, it is useful to have one person in the lead who focuses on the quality and consistency of the tasks. However, it is very desirable to peer-review all tasks.

Design each onboarding task with a focus on a distinct part of the software system. Each task should aim to provide the new member with valuable insights into isolated, specific aspects of the system. As the new member completes each task, their understanding of different areas within the system will broaden. It is optimal for these tasks to be arranged in a sequence so they gradually build upon each other, allowing the new member to gain a comprehensive and deeper knowledge of the software.

When writing an onboarding task, consider the perspective of a novice end user interacting with the system. Encourage the new member to understand the system from a user's point of view. In parallel, provide them with guidance on the code base that is relevant to implementing those user interactions.

In each onboarding task, guide the new member towards relevant resources for documentation and introduce them to team members who are knowledgeable about specific areas of the system.

After completing each onboarding task, ask the new member to contact the assigned teammate to discuss and evaluate their findings. By assigning every team member as a contact person for at least one onboarding task, you provide opportunities for the new member to interact with all team members during the onboarding process.

Due to the finite number of onboarding tasks that you can create, it is crucial to prioritize and concentrate on the most essential aspects of the system. Make sure that the most relevant parts of the system receive sufficient coverage.

Besides writing a collection of individual tasks, also set up a guide on how a new team member is meant to follow the onboarding process. Make it clear to a new colleague what your expectations are, whom they can ask in case of any general questions, and what they should do if they get stuck on a problem that was not considered when the task was written.

Depending on the position of the new team member, you probably need to choose a suitable subset of all available onboarding tasks. A database developer might not need to dive into UI details, and vice versa.

Make sure that every task is meaningful. Don't waste your new colleague's time and energy on artificial tasks. If there are open bugs that can be fixed easily, add custom tasks for those bugs. As an alternative, propose optional improvements or suggest adding missing test cases. For many developers, making real code changes is more rewarding than just reading and understanding code.

Also, the order of the tasks matters. You should start with easier tasks and slowly increase the complexity of them.

Feel free to add fun parts. You could, for example, use your creativity to transform these tasks into a treasure hunt, where each task brings the new colleague closer to some final reward.

* * *

Creating and assigning Onboarding Tasks to a new member brings a couple of *advantages*:

- With a full set of tasks, your team is well-prepared to onboard new developers. Chances are high that the new developer will also get a good first impression of the team.
- While working on the tasks, the new colleague will gain many insights into the system's implementation and locate primary entry points within the source code.
- The balanced approach of exploring the system both as a user and as a developer will help the new member develop an understanding of both the most important business requirements and the technical foundation of the system.
- The collaborative approach will enable the new team member to access valuable information from both written documentation and from experienced teammates.
- After completing all assigned tasks, the new team member should have reached out to each teammate for discussions, ideally going beyond just task-related topics alone.
- While the creation of onboarding tasks consumes both time and effort, it pays off quickly if new members join the team regularly. If you regularly revise and update the tasks, you are always prepared to welcome new members even when tight deadlines are ahead.
- As a team, you will have acquired valuable information about the new member's aptitude for understanding your system and their overall compatibility with the team. Should there be any gaps in technical skills, for example, prompt action could be taken to address these deficiencies.

However, this approach also has *liabilities*:

- Crafting well-designed onboarding tasks requires a lot of time and effort. Given the dynamic nature of most software systems, it is necessary to periodically update the tasks. To ensure consistency and accuracy, it's essential to consider these tasks as part of the system documentation.
- Discussing the outcome of a task can be done in a few minutes. If neither participant is willing to talk about topics beyond the actual task, the opportunity to foster a meaningful relationship is limited.
- Deep connections with others are not a given for every person. While it is essential to encourage interaction among team members, you cannot force anyone to participate in the process fully as expected.
- For some developers, onboarding tasks without making real code changes might be boring. Those developers might be tempted to shorten those tasks in favor of working on real tickets.
- Projects are often faced with constraints such as limited time for any nice-to-have tasks. In the case of a tight deadline, a project team might not be able to spend time on any proper onboarding at all, let alone writing onboarding tasks.
- Judging the overall abilities of a new team member from their performance in some artificially designed tasks might be misleading.
- Onboarding tasks are best suited for experienced developers who are used to learning new systems on their own. Less experienced developers might need more support. A mentor, i.e., a colleague who continuously supports the new developer, might be a better option.

* * *

If you plan to apply this pattern, please be aware that it requires a joint effort of the whole team. Setting up a set of onboarding tasks should be handled as any other task in a project: it needs time, resources and dedication.

When a new person joins the team, it can also be helpful to assign a single team member as the main point of contact. That team member would explain the whole process to the new colleague and help in case of any questions about the process itself.

Example

The following examples present a template for onboarding tasks.

Let's assume that an existing team develops an application for end users. There's a new team member onboarding. Setting up the working environment can be an excellent starting point for the very first task:

Task	**Set up your local working environment**
Steps	Please consult our project wiki and read and follow the readme file in our git repository. In a nutshell, you need to perform these steps: • Clone the git repository. • Set up the local database and configure it properly. • Set up your favorite IDE and import the code. • Start the application. • Connect to the frontend via this link.
Contact persons	Ask Michael for any problems with the database setup. Sarah knows best how to work around any problems setting up the IDE and importing the code.
Discuss with Emily	• How do we manage database schema changes? • What is our strategy to branching and merging in git? • What functionality is included in the project's build script?

Once the system runs locally, the new member should create both a new account like an end user would do and an admin account:

Task	**Create new users in your local environment**
Steps	Our system distinguishes between end users and admin users. Please create both a new end user account and a new admin account. Please also learn the basics about security in your system. You can find a (slightly) outdated introduction to our security related implementation here. Look around in the application, both as end user and as admin and find out what you can do. For end users, we've got some help pages that will help you as well.
Contact persons	Jennifer knows best all about security. William wrote the help pages and is keen to get your feedback about missing bits and pieces.
Discuss with Ken	• There are roles and permissions. How do they relate? • You can create a purely local account or you can use a social login. How did we try to minimize the impact of those different account types on our account logic? • What is the main functionality that a new user can initially do? • How does the system behave differently when you log in as an admin?

Most systems need to execute some tasks periodically in the background and run jobs:

Task	**Investigate into scheduled jobs**
Steps	So far, you've started and looked into our API service. There's another service that is solely responsible to launch background jobs. Please take a close look at that code. Unfortunately, there's little documentation available. The code is located inside the `scheduled-jobs` sub folder, and there's a small readme file. Try to run a few jobs, for example, the "old registration clean up" job and the "send e-mail for due payments" job.
Contact persons	Jessica wrote most of the infrastructure to handle jobs. She should be able to explain the basics to you in detail. Andrew from the business department knows best about the functional requirements of our jobs.
Discuss with Lisa	• How do we deploy the job scheduler in general and how many instances do we deploy? • Why do some jobs process data in batches while others do not? • How do we make sure that jobs are repeated in case of errors? • How do we record the outcome of each job?

Finally, let the new member work on a real task:

Task	**Fix formatting issues on our main product page**
Steps	There is an open bug in our ticketing system (link). It seems that there is a corner case when we format the product description from our internal representation. Please look into the ticket and try to reproduce the issue in your local environment. Create a new git branch and try to fix the issue. Do not forget to add another unit test. Once you succeed, please create a pull request.
Contact persons	We don't have an up-to-date guide on our git conventions. Please contact Joshua before you start coding. He will shortly explain our conventions to you. Olivia created the product page. Ask her if you need any support on the product description logic.
Discuss with all of the team	Congratulations! You've finished your onboarding tasks by making some real code changes. Now, please provide feedback to us how we can improve these tasks. :-)

Conclusion

Onboarding Tasks can be an effective means to start the onboarding process of a new team member. By combining both understanding of business requirements and exploration of technical details, these tasks serve as foundation for a successful integration into the project.

Furthermore, these tasks offer opportunities for both the new colleague and the existing team members to connect to each other. This often leads to a positive working environment.

Acknowledgments

I'm very thankful to Uwe van Heesch who gave me a lot of concise and insightful feedback that helped me improve the paper! This paper has also been discussed at the EuroPLoP 2024 conference. The participants of my workshop group (Luciane Adolfo, Adrian Schuckart, Michael Weiss, Francisca Almeida, Takako Kanai) gave me many more excellent suggestions to further revise my work!

References

[1] Veli-Pekka Eloranta: Organizational patterns: Creating an on-boarding experience, VikingPLoP 2016: Proceedings of the 10th Travelling Conference on Pattern Languages of Programs, https://dl.acm.org/doi/abs/10.1145/3022636.3022638

[2] Veli-Pekka Eloranta: Patterns for making entrance to a new organization culture a pleasant experience, VikingPLoP 2017: Proceedings of the VikingPLoP 2017 Conference on Pattern Languages of Programs, https://dl.acm.org/doi/10.1145/3158491.3158498