ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea Triennale in Informatica

Integrating Agile development models with User Experience methods

Relatore:
Chiar.mo Prof.
FABIO VITALI

Presentata da:
AGNESE SGARBI

Sessione II
Anno Accademico 2022-23

Table of Contents

1. Introduction

Companies need to deliver excellent customer experience to stay competitive in their market space. Companies that have adopted Agile development models can encounter difficulties when they try to integrate user experience (UX) research and design tasks in the development process.

What if there is a better way to integrate user experience design in an Agile development process?

Since the release of the Agile Manifesto in 2001, many companies have switched from a traditional development model to an Agile process.
Agile processes have many benefits. They reduce the risks a company takes by removing the long research and design phases done in traditional software models before the software development step.
Instead, Agile processes focus on creating and delivering a minimal viable product (MVP) and use the MVP to gather feedback and validate assumptions, hypotheses, and market fit. Once the input is collected, an Agile development team will iteratively improve the product through a series of releases. The Agile Manifesto encourages the developers to do a lot of smaller releases, allowing the team to continuously gather feedback and improve the product, keeping the risks low. A suggested timeframe for a release is two to four weeks.

This short release cycle brings many challenges for teams that want to integrate UX research and design since these activities have evolved to fit the longer timeframes the traditional software models allow. Companies and experts face many issues when adapting UX activities to fit into the agile release cycles.
This thesis intends to propose a process to help companies who want to focus on delivering a great user experience while using an Agile development process.

To support companies and developers, engineers created software models. The first frameworks developed are the ones we call “traditional models” or “heavyweight models.”
The traditional software models are sequential. They comprise a list of steps, where each operation starts after the previous one is completed. When using a traditional model, the requirements of a project are well-defined and are not supposed to change. A lot of work

goes into defining the prerequisites and ensuring they are correct because changing the requirements after they are set is very expensive.
All traditional frameworks have the following phases: requirements, research, design, implementation, testing, and release.
Between the requirements, research, and design phases, the user experience experts have ample time to work on creating the UX guidelines for the developers.
It’s important to note that many user experience tasks require ample time, especially those involving the users, like user research and user testing. Chapter 2.2 offers an in-depth examination of traditional software models.

On the opposite side of traditional models are Agile systems.
Agile frameworks are iterative and flexible. One of the core objectives of Agile is to enable companies to respond to change quickly, without wasting any time. These attributes of Agile methods are essential when the markets you are developing for are subject to frequent changes.
Agile methods have short iteration times (usually between two and four weeks) and heavily rely on user feedback to respond to market variability quickly. The feedback is needed to decide what to work on in the next iteration. Agile development models are described in more detail in Chapter 2.3.

With such a short lifecycle and no dedicated time to user experience tasks, it’s hard to see how teams can focus on delivering great UX to their users.
While some Agile patterns have a dedicated design phase, the design tasks must fit into the current iteration while also leaving time for its development, leaving the designer almost no time to conduct the appropriate research and create UX designs.
Another issue that stems from only doing UX design feature by feature is the risk of having the final product with a non-cohesive user experience.
After reviewing academic articles and documents written by professionals in the UX field, I have divided the issues a company or team can face into three main categories.

Team Coherence refers to the alignment of all the team members to a common purpose. The research of Kashfi et al. [KNF16] revealed that it’s common to have power struggles between UX professionals and the other team members.
These issues can be related to ownership of responsibilities, division of tasks, decisional power, and timing of the UX tasks.
Furthermore, a team can have a lack of consensus about the value of UX [KNF16]; this is more common in functionality-based teams where UX is done “on the side.” Having such a strong focus on functionalities can be an issue when doing sprint planning, as this can turn into a deprioritization of UX tasks. [DE10] For a more comprehensive exploration of this topic refer to Chapter 3.3

The second category is **scheduling**. One of the issues that stands out more prominently is the length of some of the UX activities, which can easily span more than one Agile iteration.

Another problem part of this category is the perceived length of UX stories. From the research of Singh [SI08], we know that developers tend to overestimate features with UX requirements compared to projects without them.
While it's true that following these requisites can lengthen development, some of this extra length is tied to the lack of experience of the developers.
It's also essential to avoid falling into a development-focused approach as this can prevent UX tasks from being scheduled in favor of carrying out development tasks [PW19] [SI08]. This situation can directly result from overestimating the implementation costs of UX stories. Chapter 3.4 offers an in-depth examination of the scheduling issues.

Knowledge base is the last category, and it encompasses all the issues related to the lack of knowledge developers have about UX and UX experts have about development.
Kashfi et al. found out that people's understanding of the concept of UX is inconsistent; developers tend to have less knowledge about UX [KNF16], while product owners often discard user experience because they are overwhelmed and focus on other, more pressing tasks. [SI08]
There are also many misconceptions about Agile patterns; people can think that Agile is about being fast and forget that its goal is delivering software of high quality. In these situations, setting UX tasks to the side can be easy. [HV16]
Furthermore there be a lack of knowledge about what responsibilities the other roles in the team have.
It is complex to evaluate the impact that working on UX has, and this can lead to a lack of consensus on the value UX tasks have [KNF16]. This lack of agreement can lead to a deprioritization of UX tasks, creating a snowball effect where these tasks are not done or only done on the side. [KNF16] [HV16] The knowledge base issues are examined in depth in chapter 3.5.

The documents I have examined propose some solutions, which are often incomplete since they primarily focus on giving the user experience experts enough time to carry out their tasks. Finding a way to do UX tasks is only one of the steps to improve a product's UX successfully. I have compiled a list of steps to go hand in hand with the methodologies proposed by the articles.
There are three main patterns proposed by the articles.

The "**one sprint ahead**" model suggests having two parallel tracks, with the UX team working on research and design at least one iteration ahead of development.
There are two separate backlogs, one for development and one for UX tasks, but the two backlogs are kept in sync.
It was initially proposed by Miller [MI05] and Sy [SY07], and it's the most recommended process.

The "**three tracks UX Research**" mainly focuses on UX research, as the name implies. It was proposed by Handa and Vashisht [HV16].
In this method, the research is divided into three tracks of varying lengths:

- The strategic track aims to shape the product's future and is the longest. Each cycle is supposed to span several months.
- The tactical track is more short-term and spans several weeks. Its goal is to research specific features and to help prioritize the product backlog.
- The validation track covers the evaluative research to verify designs, check for usability or accessibility issues, and measure user satisfaction. This track has the same length as a sprint.

The three tracks UX research model aims to help teams conduct research.

Lean UX is another appreciated methodology.
Deepak defines Lean UX as a process that ensures equal participation of design and development professionals.
Gothelf and Seiden [GS21] explain that Lean UX works on three principles:

1. Design thinking. Using the direct observations of people’s needs to fuel innovation.
2. Agile software development.
3. Use the “build – measure – learn” feedback loop proposed by Eric Ries in the Lean Startup Method to minimize project risks by delivering a minimum viable product and continuously improving it.

The issue is that these patterns alone can not guarantee a successful integration of Agile development and UX design tasks.
These methodologies are insufficient to successfully integrate UX design within an Agile development environment because they only solve part of the issues. Please refer to Chapter 3.7 for more information of the proposed methodologies.

Thanks to my work experience and through the analysis of our current and past development patterns, I have created a list of steps to follow to ensure a better transition from either a heavyweight model or an Agile model that doesn’t integrate well with the UX design requirements.

Balsamiq is a small company that has grown organically over the course of 15 years, growing from being a single-employee company to having 35 employees. I’ve worked there since 2015, and I’ve had the chance to see us grow and tackle many of the issues cited by the research papers.
As the years went by, we’ve become more organized, and we’ve gotten better at handling and prioritizing our backlog.
Back when I started, we didn’t have a coded process. Sure, we used to do releases once a month, and we had a one-year and three-year roadmap. However, the roadmap was vague, and the developers were in charge of deciding the sprint backlog.
Since then, we have improved our processes a lot.

We now have a 6-month roadmap. Each quarter, we plan which features will be worked on by the UX team and which are ready to be implemented.
The team managers ensure all the scheduled projects have a lead developer assigned and help with the quarter schedule, ensuring everyone has a reasonable workload.
The team leaders meet weekly to decide on that week's backlog and evaluate if anything needs urgent attention.
One coworker got specialized training in performing UX research, and since then, we've introduced user experience research in our development process.
All this organization allows us to deliver quality features in a timely manner. For a comprehensive explanation of the process we follow at Balsamiq you can refer to Chapter 3.

Through the years, we've faced many challenges, and by analyzing them, I have compiled a list of steps to help teams that wish to integrate UX activities into their development process better.

- **Make sure everyone is on board**. Collaboration is of the utmost importance during this transition phase.
- **Create a feedback culture**. Getting and receiving feedback has many benefits, but the most important one is that it allows you to understand where the friction points are and to catch any issues early on.
- **Align the knowledge base**. As it emerged from the academic articles, people in the same team will have different assumptions and levels of knowledge on UX and Agile. We want to ensure everyone has a base level of knowledge.
- **Choose a development approach**. Whether you're already Agile or switching from traditional development models, you will need to change how you work. Pick an already established model; there are no added benefits from starting from scratch, and slowly tweak it to your team's needs.
- **Define roles**. It's essential that roles and responsibilities are clearly assigned and that this information is available to everyone. Well-defined roles prevent conflicts and ensure no work is left behind because no one owns it.
- **Keep a loose schedule**. Initially, allowing your team more time to carry out tasks and implement features is crucial. Significant changes can temporarily slow a team down; don't fret; it should resolve naturally. Developing new features will naturally take a little bit longer since you have added the UX tasks to them, but you will gain value in the quality of the features.
  Unfortunately, there is no "one size fits all" approach and immediate success is not guaranteed even when following directions.

The challenges we have encountered and the suggestions for a successful transition are expanded upon in greater depth withing Chapter 5.

Because of this transition's nature and its limitations, it's impossible to guarantee its success.

I have created an expanded model to help teams make a successful transition. In the future, the model should be tested in numerous and varied work environments.
Ideally, feedback from the participants would be collected often, alongside any documents produced. Through the observation of the teams using my proposal and the feedback, the method can be expanded and improved.
I also expect to find differences between small and big companies. So perhaps my findings will not apply to bigger and more structured companies, or they will need to be adapted to their environment.
It would be interesting to gather knowledge about how bigger workplaces with many teams work and see if my solution can be applied to them or how it needs to be changed.

2. UX design and software engineering approaches

2.1 Introduction

“Software engineering is a growing set of disciplines and procedures for the dependable development and maintenance of software” [MI80] is the definition given by H. D. Mills in 1980. Since then, several models have been created to develop and maintain software more reliably over the years.

A software development model provides a framework to plan, develop, and release an application. All models include the same basic operations: requirement research, design, implementation, testing, and release. The main difference between models is in the timing and execution of the phases.
These approaches can be divided into two categories:

- Traditional systems, also known as heavyweight,
- Agile systems.

Traditional systems are sequential; each step starts after the previous one and should only be done once per project, except for the spiral model, which has cycles like Agile processes. Furthermore, the requirements are fixed and are not expected to change; changes are expensive because many, if not all, steps must be redone. Agile systems, in contrast, are iterative and flexible.
I will illustrate the most common traditional development models and analyze how UX Designers do design tasks in these models. Then, I will review the most common Agile systems and highlight the issues in adapting UX Design tasks to a more dynamic environment.

2.2 Traditional Software Models

I will explain how 3 of the most common methodologies structure their software development process, their strengths and weaknesses, and when they perform the UX Design tasks.

2.2.1 The Waterfall Model

The Waterfall model proposed by Royce in the 1970s [RWW87] is sequential, and each activity must be completed before starting the next one.
However, when I analyzed multiple papers, I found that each author might identify different activities. However, the following phases are commonly found in the literature: requirement analysis, design, implementation, testing, and operation and maintenance.

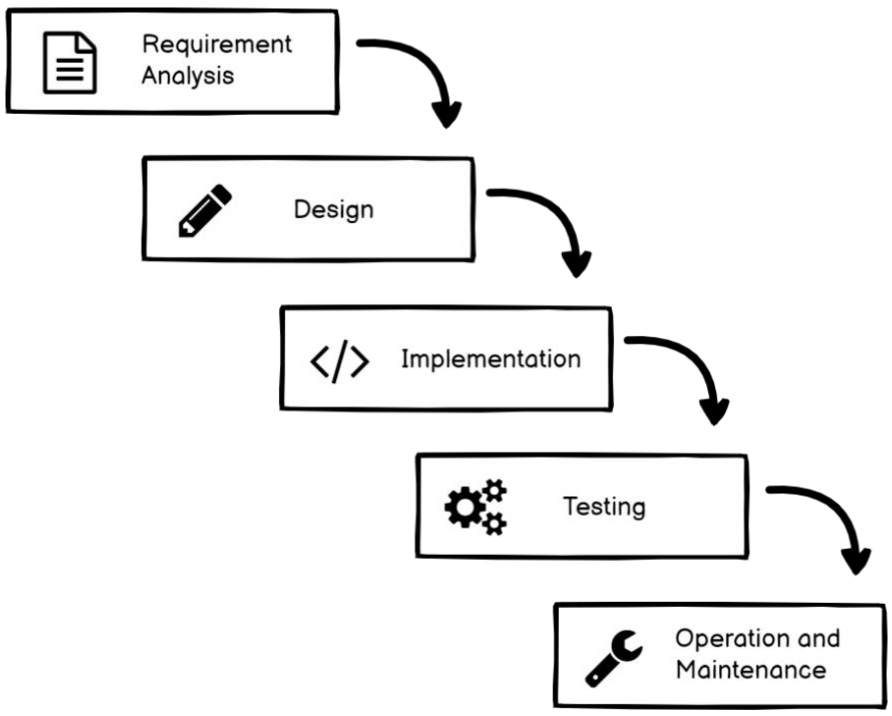


Figure 1: The phases of the Waterfall Method

In the Waterfall method, there is ample space to carry out all the User Experience Design tasks during the “Requirement Analysis” and the “Design” phases before development begins.
The development phase starts once all the requirements are gathered and documented.
The Waterfall model relies on extensive documentation, so developers work autonomously without interaction with UX experts.
If a requirement changes, the current work has to stop, and everything starts from the beginning. Redoing all the steps means wasting a lot of work (and money).
The likelihood of a requirement changing depends on the target market and the kind of end-user. For example, developing a new social network app is quite different from creating software to fly airplanes.

2.2.2 The Spiral Model

The spiral model was first proposed by Bohem in 1986 [BB88], this model combines an iterative process with elements from the Waterfall method. Its main goal is to manage the

risks encountered using the Waterfall method by iterating on four phases and doing more minor releases.
The spiral model is best suited for a larger, more complex project with high stakes.
In this model, each iteration is more expensive but refines and improves the work of the previous cycle.
It might seem like the spiral model should be an Agile process. Still, it is in the "traditional development" category because it focuses on risk reduction rather than collaboration and efficiency.

The spiral model has four main phases:

- **Identification** is when the requirements are described. For example, they can be business, system, or sub-system requirements.
- **Design.** In this phase, any design can be done, from architectural to physical to UX Design. The Design phase is the only moment where UX Design can be done.
- The **build** phase is where the production of software happens. The Spiral model allows multiple software releases, one for each Build phase. Finally, the customers are sent these builds for feedback.
- **Evaluation and risk analysis.** During this phase, the team collects user feedback to plan the next iteration. It is time to execute the risk analysis and risk management tasks.

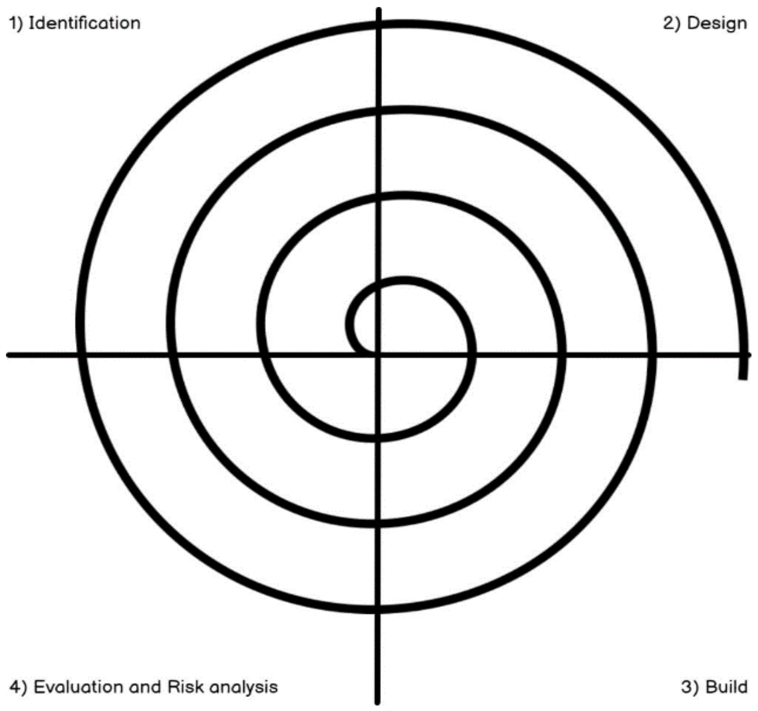


Figure 2: The phases of the Spiral Method

Since the first few iterations are short, there is little time to carry out UX Research, which is time-consuming. However, UX Research is crucial for the success of a project; consequently, it is almost impossible to do research during the first few iterations. In the beginning, during the design phase, the team members have the time to integrate feedback from one cycle into the subsequent ones.

For some projects, this might be enough to balance the small to non-existent upfront UX Research even though this might waste some time by needing to refactor or change part of the features.”
The multiple releases and design phases allow more flexibility compared to the Waterfall method.

2.2.3 The Unified Process Model

The Unified Process model attempts to unify the best features of traditional development with the best practices from Agile methods. This model was proposed by Ivar Jacobson, Grady Booch, and James Rumbaugh in the late 1990s in a book titled “The Unified Software Development Process”. [JBR99]
This change means that the Unified Process is both iterative and incremental, and each phase can be composed of multiple, timeboxed iterations.

The model is divided into 4 phases:

- **Inception phase.** During this phase, the model requires the involvement of customers and stakeholders to define the project requirements and plan activities. The result of this phase should be an initial architecture of the application, a plan for the project, and some preliminary use cases.
- **Elaboration phase.** The team refines and expands the documents produced in the inception phase. As a result, most requirements are captured, use cases are well-defined, and the architecture is defined.
- During the **construction phase**, the developers implement the project following the documentation produced during the previous stages.
- **Transition phase.** In this phase, the release happens, and users receive the software. Afterward, the team can plan changes to implement during the various transition phase iterations based on the feedback received from the users.

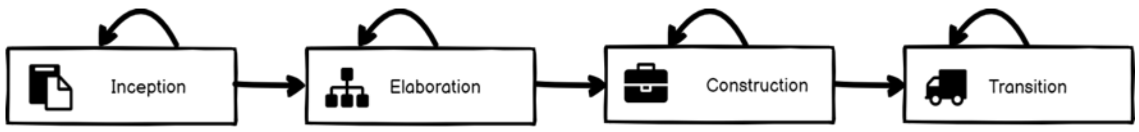


Figure 3: The phases of the Unified Process Model

This model was developed with a focus on architecture and is use-case-driven. While it does not focus on UX Design, the process is adaptable to the team’s needs. There are many variations that integrate UX Design during the inception and elaboration phase.
The Unified Process is easy to adapt to the needs of different projects, but this flexibility makes it very complex to learn and use and adds much overhead on top of the development.

2.2.4 Advantages and disadvantages of traditional development patterns

There are many differences between these three models, and I could analyze even more models, each with its unique characteristics. However, all traditional development models have shared characteristics that differentiate them from Agile models.

All heavyweight methodologies aim to be predictive and efficient to achieve a good outcome across multiple projects. They do so by imposing a structured process, which has the following common characteristics:

- Predictive approach: All the requirements are well defined, and all prerequisites are ready before starting the development phase. The expected outcome is clear and will not change.
- Documentation: Traditional models rely heavily on documentation; teammates write all relevant information in the knowledge base, especially requirements and design artifacts, as the developers refer to these documents during the development.
- Process-oriented: The procedures are clearly defined, with multiple steps or phases, and everyone involved in the project follows the process. However, a fixed process is required to improve the development model’s reliability and get consistent results.
- Tool-oriented: When using a heavyweight model, the team uses specific tools for project management, user requirements, and software development. It is either the process that defines which tools to use or the company that picks them.

Since traditional patterns are predictive, they all have time to do User Experience Design before development happens since it is part of the requirements definition.

However, traditional methods also have various drawbacks.
Having the team specify all the requirements at the beginning and the inability to accommodate changes throughout the project can be extremely complicated for more extensive projects or when the target market is highly variable.
Traditional development patterns have a big focus on documentation. As a result, a lot of effort is put into writing and maintaining it, which might be excessive for some projects, leading to wasted time and extra stress for the employees who have the task of keeping the documentation up to date.
Because the software release happens at the end of the project, and the project does not involve customers before the release, feedback from users might arrive too late. Even worse, full features that are not useful for end-users might have been developed. [SMG13]
These concerns led to the creation of Agile models.

2.3 Agile Development Models

In February 2001, a group of software developers met and wrote the “Agile Manifesto,” a proposal for an alternative to documentation-driven, heavyweight software development processes.
These are the four core values of Agile:

“Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan.” [BJR01]

These software developers were not anti-methodology or anti-documentation, but they felt that the traditional development models were too focused on those aspects.
They acknowledged that documentation and processes are essential but writing them might be a waste of time. It is not helpful to write hundreds of pages of documentation and then not update it or use it, and it is not valuable to plan carefully in a turbulent environment. Planning is helpful but should be done with the flexibility to follow the changes in the environment. [HI01]

Alongside the four core values, the Agile Manifesto also lists twelve principles. These principles describe a culture where change is welcome, and the customer is the focus of the work. They also highlight that one of the goals of the manifesto is to improve collaboration between developers and stakeholders to keep the business and the development teams’ goals aligned.
With all this information, we can imagine an agile model’s look. Iterations should be short; 2-4 weeks is a reasonable timeframe, and the team should do a software release at the end of each iteration. Continuous communication and feedback with users should help the team adapt to the requirements and decide what to work on in the following iterations. Finally, the team should meet regularly to see if they can do anything to improve the development process. The people involved in the project write the documentation, but unlike heavyweight methods, this is lighter and constantly maintained.

I will introduce three Agile development patterns and briefly explore how UX Design can be integrated. In the following chapter, I examine the issue of integrating UX Design and Agile methods and how to mitigate known problems in more depth.

2.3.1 Scrum

Scrum is the most popular Agile methodology, developed in the 1990s by Sutherland and Schwaber. [SS07]
A Scrum team comprises three central figures: the product owner, the Scrum master, and the developers.
The product owner represents the customers and the stakeholders and drives the product backlog. The product backlog is a prioritized list of features containing a description of all the required functionalities, and each project feature is called a story.

The team will work on one or more sprints during a release cycle. The goal of a sprint is to produce value for the stakeholders.
"Value" represents everything that improves the product. Scrum does not differentiate between bug fixing, feature work, or UX improvements.
Here is a picture illustrating the phases of a Scrum Sprint (image taken from scrum.org).

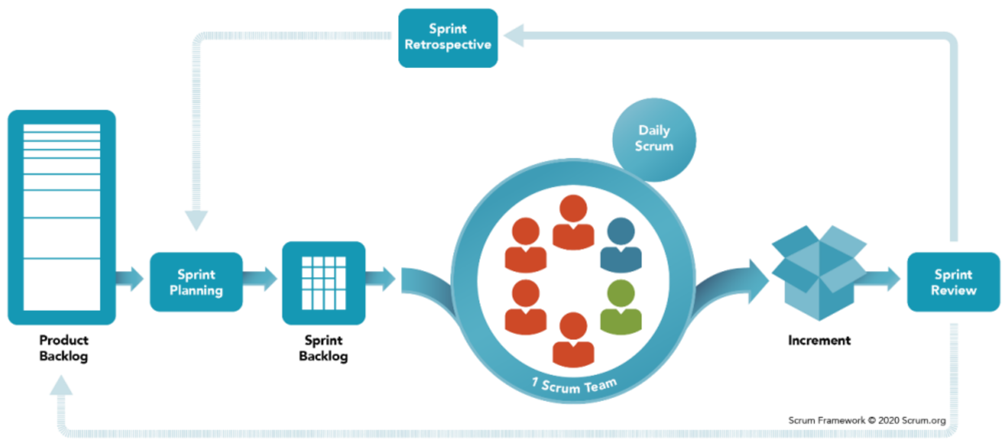


Figure 4: The phases of a Scrum sprint

During the sprint planning, the Scrum team picks stories from the product backlog to add to the sprint backlog based on the sprint goal.
The team works together to deliver value; to do so, the team must agree on the definition of done: the criteria to consider a backlog story completed and ensure that the results meet the organization's quality criteria.
During the development phase, the Scrum team meets daily for 15 minutes for the "daily Scrum." During this time, the team members share feedback on what they have done and the issues they have encountered. With all this information, the Scrum master might adapt the sprint planning to ensure the team reaches the sprint goal.
At the end of the sprint, the stakeholders review the work done during the sprint with the team. Then, the team members use the collected feedback when planning the following sprint.
Scrum does not have a phase where UX Design would be a natural fit, and the Scrum team definition does not include a Designer.

2.3.2 Extreme Programming (XP)

Extreme Programming was developed in 1996 by Kent Beck [BK00]; it focuses on customer satisfaction and changing requirements. It also emphasizes teamwork, where all parties are considered equal: stakeholders, managers, and developers.
The XP framework consists of 5 phases that the team iterates on continuously:

- **Planning**. Using the customer requirements, the team creates stories and makes a release plan, divided into iterations.
- **Designing**. While designing should be part of the planning phase, it has its own step to highlight its importance.
- **Coding**. Where the developers build the feature following XP practices like pair programming and continuous integration.
- **Testing**. This is the Core of Extreme programming. The team has to write and run automated tests (like unit tests) and acceptance to check that the requirements have been satisfied and that the changes made do not introduce regressions.
- **Listening**. The team listens for feedback to integrate into the next cycle.

But XP is not just a list of steps to follow; it has principles and practices to keep in mind while using this framework.
The five values of XP are communication, simplicity, feedback, respect, and courage. These values tell us that team members should be highly communicative, give and receive feedback often, and aim to keep things to avoid wasting time. The principles are built on top of the values. They clarify what is most important for a team. I will not go into more depth since they are irrelevant to this thesis.

2.3.3 Feature Driven Development (FDD)

Feature Driven Development is the last agile methodology I am going to illustrate. FDD works best on large-scale projects, and it organizes development around features. [PF01] Features are functions that bring value to the customer and are similar to user stories in SCRUM.
In FDD, first, you gather data to clearly understand the overall project goals and the customers' needs. The FDD process begins once this first step is complete and all requirements are understood.

- Develop an overall model: The team develops detailed domain models to outline the whole system. The models are then merged to obtain one cohesive and comprehensive model. As development progresses, the team updates the model with more details.
- Build a feature list: Use the information gathered to create a list of features, the model, and the requirements. Each story should be small enough to be completed in two weeks; otherwise, the FDD model states that we should divide it into smaller-sized features.

- Plan by feature: A chief programmer analyzes the complexity of each story and how they relate to each other and determines the order of implementation and what team or team members work on it.
- Design by feature: With the team's help, the chief programmer designs and models the feature. After the design is approved, the developers can implement the story.
- Build by feature: Implement the feature following the design. After the developers build it, they write tests; the feature is inspected and approved. Once approved, it moves to the main build, releasing it to the users when appropriate.

2.3.4 Advantages and disadvantages of Agile development Models

All the 3 Agile methods I have illustrated have some common principles: the development cycle is short, and the requirements might change at any point during the product development. Therefore, the upfront research and design phases are much more limited to accommodate changing needs. Instead, they prefer to outline the requirements of the product and proceed better to define them when a particular feature is ready to be implemented.
But Agile methods also have their drawbacks. Involving customers at the end of each cycle, every 1-4 weeks, can create so much feedback that could overwhelm developers and generate a lot of overhead of fixes and features to be tracked and discussed.
Having more barebone documentation means that new developers take longer to get up to speed on the project and might need more help from experienced developers, slowing down the project.
Complex features might not fit into the shorter iteration phases, especially later in the project when the system is more elaborate. Developing features that do not fit the iteration length increases the risk of delays, which can stress the developers and upset the customers. [SMG13]
UX Design suffers due to the shorter release cycle; even when the model has a design phase, like the Extreme Programing or the Feature Driven Development models, the time constraint makes integrating UX Design a complex task.

3. Integrating UX Design into Agile Teams

3.1 Introduction

As illustrated in the previous chapter, Agile Development strategies and UX Design have some shared approaches. However, they also have differences that can create friction between UX designers and developers, resulting in a sub-optimal experience for both parties.
In this chapter, I will analyze some of the literature published over the years, highlighting the main issues and what is most important for teams that want to improve the collaboration and quality of work produced by UX professionals and developers.

3.2 Analysis of the current literature

I have analyzed various documents, most of which are academic, but some are articles written by professionals in the UX field. Much literature has been produced over the years, so I have decided to organize the information into three sections: team coherence, knowledge base, and scheduling. These are general topics where a team might encounter difficulties or friction points.

3.3 Team Coherence

According to Beck, team coherence – the alignment of every team member to a common purpose - is essential in Agile teams as one of the key goals of the planning game in extreme programming is to bring the team together [BK00].
Team coherence and collaboration are what can make or break a project. A highly collaborative team that has clear objectives will produce better work faster.
The common ground to build team coherence is the user. Agile methods greatly emphasize the user; for example, Schwaber advocates for user involvement during the development process [KS97]. Similarly, UX also requires early and continual user involvement.
However, many obstacles can arise in a newly formed team of this kind.

3.3.1 Power struggles

Kashfi et al. [KNF16] conducted an explorative qualitative study investigating the challenges of integrating UX practices into software development processes. The findings reported that there often are power struggles between non-UX professionals and UX professionals. These power struggles can be many issues, from the division of tasks and responsibilities to making decisions such as when to work on UX tasks.

3.3.2 Lack of consensus on the value of UX

Another issue is again highlighted by Kashfi et al. [KNF16]: the lack of consensus on the importance of UX work.
When looking at market-driven products, the user experience represents a core aspect of the program. Usually, companies place a greater value on the UX of said products than functionality-based companies, where UX is done “on the side” and not a fundamental value.
According to the practitioners interviewed by Kashfi et al., when a product is more “technical,” the business focuses on functionality, and the UX is often left behind. We can call this kind of software development approach “development driven,” which is a challenge the designers can encounter during Agile sprint planning [DE10].
Even if users now expect programs with good user experience, the data Kashfi et Al. have analyzed shows that not all stakeholders perceive UX work as important, especially when we look at market-driven products.
When UX Design tasks are considered less important than development tasks, teams might develop “tunnel vision,” focusing on delivering as many backlog stories as possible to the detriment of the user experience. [DE10]

3.4 Scheduling

Scheduling tasks, projects, and meetings is at Agile development's heart. Fortunately, both Agile development and UX Design are iterative by nature, and they use user feedback to iterate and improve the design or the application [FSM08].
What can prove difficult is adapting some of the lengthier UX activities to fit into the Agile iteration cycles.

Kuusinen et al. list many solutions to the most common scheduling problems. They suggest involving a UX specialist during the roadmap ideation to avoid starting UX work too late into the project [KMP12]. To give the UX specialists enough time to carry out their tasks, multiple researchers suggest the designers work one or two sprints ahead of development [KMP12] [PW19].
Working one sprint ahead gives the designers more time to carry out the research and design tasks and the developers more time to implement the designs.

Working a sprint ahead does not imply working separately; the more communication, the better. However, the UX specialists and the developers should still interact for feedback, to keep each other in the loop about what work is being done, and to discuss issues they are encountering. For example, Kuusinen et al. suggest involving developers in design meetings and UX experts in architectural meetings; sharing information will prevent a mismatch of expectations between the software developers and the UX designers. Moreover, they suggest synchronizing the development and UX roadmaps [KMP12]. I will give a more in-depth overview of the "one sprint ahead" method later in this chapter. To prevent UX development tasks from being set aside in favor of working on other development issues, Kuusinen et al. suggest setting criteria for minimum (or desired) UX design realization. This standard is meant to prevent stories from being accepted/delivered when it is not met.

According to Handa and Vashisht, UX research should be carried out iteratively and fit in an agile iteration, except for the initial UX research that must be done before the beginning of the project [HV16].
Pillay and Wing suggest carrying out the initial UX design tasks as a separate pre-development process [PW19]. This pre-development step can be longer than one iteration but will still be shorter than the research phase when adopting heavyweight methodologies. Handa and Vashisht suggest the developers work on architecture while they wait for the initial UX Research phase to be completed. The developers could even implement some basic sketches produced by the UX team during this phase while considering that they will probably need to make changes to them in the future [HV16].
However, for this strategy to work, the team needs a high level of open communication and collaboration.

3.4.1 Perceived length of UX stories

As stated by Singh [SI08], developers often estimate the cost of implementing a user-friendly story to be significantly higher than implementing it without following the UX requirements. While it can be true that implementing UX requirements can lengthen the development, it is primarily a lack of experience working on user-centric features that leads developers to overestimate the effort needed to implement a feature.

3.4.2 Development focused approach

One of the things that can happen during the project is that the main focus is on development tasks, aiming to release software as quickly as possible and neglecting UX tasks such as research or evaluation[PW19][SI08]. This situation can result from overestimating the time costs of implementing UX stories. A development-focused approach is detrimental to the quality of the UX in the product, which will be lower, and when the team decides to focus on UX, more time and effort will be needed to improve things.

Moreover, the data analyzed by Kashfi et al. shows that the main focus of the organizations interviewed is on testing functionality.
When the organization is limited by budget or time, UX evaluation is either non-existent or rarely done compared to other testing activities. In around 50% of the survey responses, Kashfi et al. UX evaluation was done without involving users, and some organizations even replaced UX evaluation with usability testing. [KNF16]
These two issues contribute significantly to creating software with low-quality UX.

3.5 Knowledge base

Lack of knowledge is one of the reasons why UX activities are set aside or only partially done. When an Agile team does not know how to properly implement a UX strategy while developing the software, it becomes hard to recognize the benefits those UX activities have and their impact on the product.

When merging UX activities and Agile development, it is essential to consider the possibility that not all team members have the same level of knowledge about UX and Agile.
Regarding UX, Kashfi et al. noted people's understanding of the concept of UX could be inconsistent and even contradictory. Practitioners with more technical backgrounds generally have less knowledge about UX, and that knowledge is mainly limited to usability or is based on previous work experiences [KNF16].
Traditional product owners often lack the skills and motivation to design compelling user experiences. In addition, product owners frequently do not pay adequate attention to usability and user experience because they are overwhelmed with other concerns, such as marketing and sales [SI08].

Regarding Agile methods, many misconceptions can cause friction and slow down the team. Integrating UX activities becomes harder when it is unclear what the goal of agile development strategies is.
Handa and Vashisht explain that it is easy to make the mistake of thinking that Agile methods are about speed when they are about delivering quality software.
The timeboxing of the development efforts ensures that teams do not stray too far in the wrong direction and allows teams to make quick corrections and early adjustments if requirements change.
However, the short time available in each iteration can lead to teams that devote little to no time to UX activities [HV16].
The Agile manifesto also suggests keeping the documentation minimal in favor of delivering working software. However, this lack of documentation can lead to confusion about UX deliverables and objectives [PW19].

3.5.1 Understanding of roles

When many different figures work together in a project, it is crucial to understand what each role entails. Knowing each other responsibilities and the expectations everyone has of each other prevents misunderstandings and promotes collaboration.
Deepak Arasu's research brings to light that team members and UX Designers might have issues understanding their roles within an agile team. [DE10] Unfortunately, he does not expand on this issue. Still, when people are brought together to work on a project, there should be time dedicated to discussing and assigning roles and responsibilities.

3.5.2 Perceived Value of User Experience

One of the challenges in integrating a UX workflow in an agile environment identified by Kashfi et al. [KNF16] is the lack of consensus on the value of UX tasks. Stakeholders have different opinions on the importance of UX; some believe it is an essential quality of software, while others still consider it as something "on the side" or an added bonus. Treating UX tasks as an afterthought will create a snowball effect where they are not prioritized, leading to a worse result for the user.
This issue is also highlighted by Handa and Vashisht's article [HV16], which states that UX Research is frequently viewed as a lower-priority task compared to development and design. Thus, UX Research does not get prioritized in the backlog appropriately.
They also state that UX Research blocks development tasks only if your approach to agile development is a series of "mini-waterfalls." If the team is truly Agile, it will work in parallel, and UX Research will not block development or design tasks.

3.6 Literature Analysis Summary

To recap this lengthy analysis, Agile development and UX design methodologies have some common aspects that a team can leverage to make working together smoother and some traits that make the merging of the two complex.
These are the typical aspects that should be used to make the merging of the teams smoother:

- **Team coherence**. Agile methods can succeed only when all the team members work together towards a shared objective. In comparison, UX also requires the team to keep the user in mind when developing a product to ensure coherence in the produced documentation.
- **Scheduling.** Both UX Design and Agile development are iterative by nature. They rely on user feedback to plan and improve their design or application [FSM08].
- **Knowledge base**. Agile methods greatly emphasize the user; for example, Schwaber advocates for user involvement during development [KS97]. Similarly, UX also requires early and continual user involvement. The shared understanding of the importance of the user can be used as a starting point to fill gaps in the knowledge base of the team members.

And these are the differences that must be kept in mind to prevent issues:

- **Team coherence.** In this category, we find all the problems related to miscommunication and competition between developers, UX experts, and stakeholders.
- **Scheduling**. This category contains all the problems related to when to start working on UX, when and how to schedule UX stories, and the priority UX should have in the project. Issues with how to adapt UX tasks to fit into an Agile iteration are also part of this group.
- **Knowledge base.** This category represents all the issues that relate to the lack of knowledge of team members or differences in knowledge (for example, on the definition and importance of UX or the goal of Agile development methods).

Here are the most relevant advice proposed in the literature:

1. **Align the team members' expectations.** These expectations include each team member's domain of responsibilities, setting an expectation for the level of communication and documentation needed by the project, and anything that can reduce friction and competition during the project.
2. **Share knowledge**. Everyone in the team should understand how Agile works and its core principles, what UX is, its priorities, and tasks.
3. If it is a new project, **have a "sprint 0"** to give UX experts the time to do the initial research and design tasks.
4. **Work on staggered cycles**. Remember that Agile development is not a series of "mini-waterfalls." UX designers should work at least one sprint ahead of development.
5. **Keep some documentation**. While Agile discourages extensive documentation, some documentation will be necessary. Take the time to decide with the team what needs to be documented and how.
6. **Involve everyone** in the creation of the roadmap and the creation of the sprint backlog. The involvement in the roadmap creation process will keep everyone's expectations aligned and will help with the prioritization of UX tasks.
7. When something is not working for your team, do not be afraid to **try new strategies** or change things up. The team should aim for continuous improvement.

3.7 Proposed methodologies

Aside from my tips, you can find many new or adapted methodologies to integrate UX activities in agile models. Here are the three most noticeable ones.

3.7.1 One Sprint Ahead

The "one sprint ahead" approach is reported as one of the most recommended processes for integrating UX work in an Agile context [KMP12]; it was initially proposed by Miller [MI05] and Sy [SY07].

When using this method, the UX team should work one to two sprints ahead of the development team. The UX team has its own backlog but is kept in sync with the development backlog.

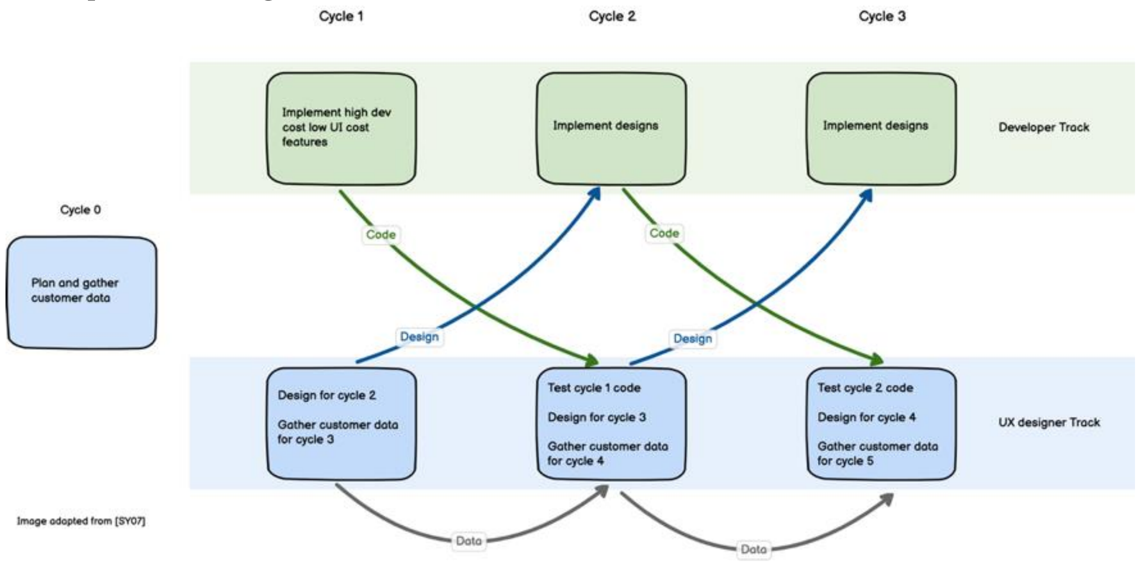


Figure 5: One Sprint Ahead model

The process starts with “cycle 0”; during this sprint, the UX team activities resemble their waterfall counterparts but are done in weeks rather than months [SY07]. These activities include defining product and release goals, interviewing or conducting contextual inquiries for market validation, preparing high-level designs, and developing user personas and workflows.

During cycle 1, the UX activities can include:

- Designing prototypes for Cycle 2 and conducting rapid usability testing to refine the design.
- Conducting contextual inquiry and interviews to investigate designs for Cycle 3.

During the first few early cycles, developers will work on software architecture or important features that need only minor designs to allow the UX team to finish their research and the initial design.
This pattern of gathering requirements two cycles ahead and designing one cycle ahead of the development will continue until the product is released.
This method still poses some challenges for UX designers. For example, the problem they are investigating might be too big to fit into an iteration. Still, big issues can usually be split into smaller problems that can fit into an iteration. Dividing a design task into smaller tasks to fit into an iteration is called “Design chunking” by Sy [SY07].
But when a UX team implements design chunking, testing workflows with users becomes harder since testing just part of a feature is more complex. Sy advises doing tests with external users once the design is at a later phase. For earlier design chunks, they suggest using in-house users (people who are not developers and have the same domain knowledge as the actual users).

Sy explains how they changed the kind and quantity of documentation produced. Their team tried to align to the agile value of "working software over comprehensive documentation" [BJR01]. The UX experts joined the daily scrum meetings to stay up to date with development and to update the developers on their progress. Some of this information, like the progress of a project, did not need to be recorded in an official document.

They decided that the information needed to organize work was written down as features or bugs and then moved to the correct backlog (UX or development).

Their UX team still keeps some documentation, mainly a record of the decisions taken for each design chunk, a record of the design iteration, and artifacts that the developers will need to implement the backlog item.

3.7.2 Three tracks UX Research

If the team's priority is carrying out thorough research, a good option is to divide UX research into three tracks, as suggested by Handa and Vashisht's article [HV16].

They suggest dividing UX research into the following categories:

- The **Strategic track** comprises any generative research that can help shape the product vision, strategy goals, and product roadmap and identify the right target users. This research is usually ongoing and exploratory. The duration of the cycle of this track is several months.
- The **Tactical track** comprises short-term, exploratory research that helps the UX team define specific product features and desired experience outcomes. In addition, it helps product owners prioritize the work in the product backlog. This research is done iteratively, and each iteration lasts several weeks.
- The **Validation track** entails evaluative research and testing to verify assumptions, validate design decisions, determine usability and accessibility issues, and measure user satisfaction and emotional response. Each iteration of the validation research lasts 2-3 or the same length as an agile sprint.

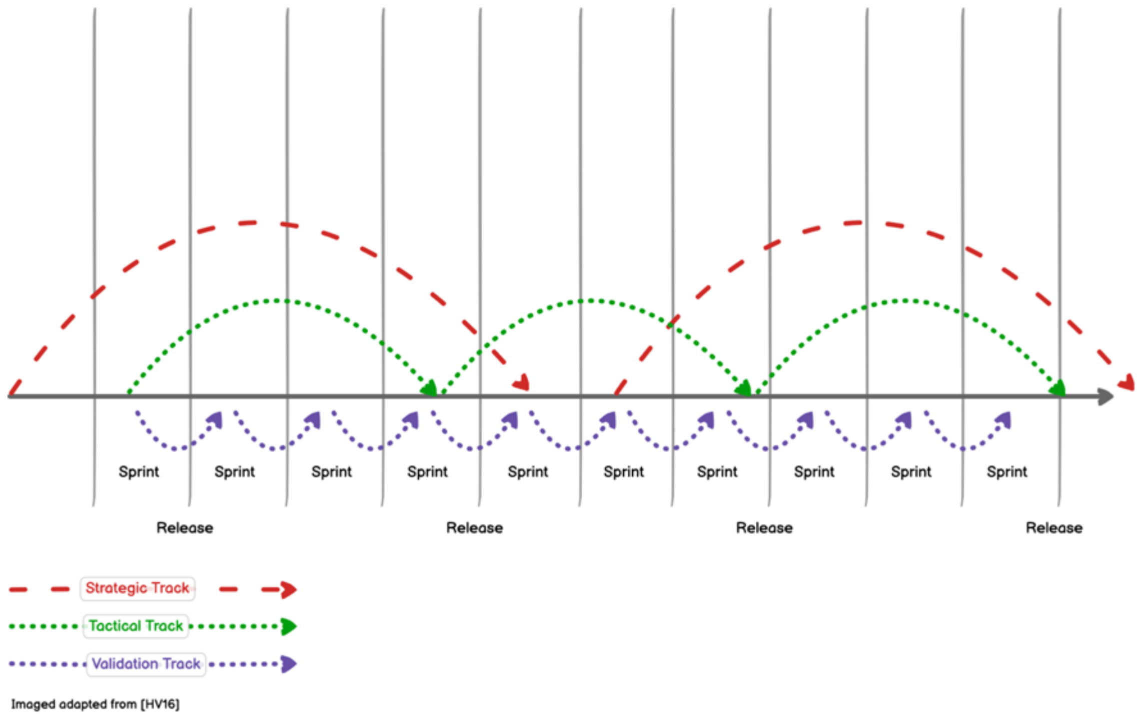


Figure 6: The three tracks of the UX Research method

Tasks from the three tracks can be carried out at the same time.
Handa and Vashisht explain in detail how to organize the product backlog. The main takeaway is that if a research item needs more than one sprint to be completed, it should be added to the backlog as an "epic" story and then broken down into smaller tasks; otherwise, it can be added as a single story. Furthermore, they do not suggest having two backlogs, one for UX and one for development, but instead using only one.
During backlog grooming, the stakeholders review the UX research items and add any relevant information, such as expectations, acceptance criteria, and dependencies. At the end of the backlog grooming phase, the stories should be appropriately prioritized.
During the sprint planning meeting, discuss and estimate the UX stories with the team, non-UX experts included, to allow everyone to understand UX research activities better.

3.7.3 Lean UX

Another approach is the Lean UX methodology.
Deepak defines Lean UX as an iterative, participatory design process that ensures equal participation by design and development team members. [DE10]
In their book "Lean UX," Gothelf and Seiden [GS21] explain that Lean UX stands on three foundations:

1. **Design thinking** means direct observation of what people want, which leads to innovation. Direct observation is studying people to understand what they like or dislike about a product, from how it is used to how it is packaged or supported. It encourages teams to collaborate across roles and consider product design holistically.

2. **Agile software development.** While Agile methods can pose process challenges for designers, Lean UX applies the four core principles of Agile development to product design.
3. Eric Ries founded the **Lean Startup method**. The lean startup uses a feedback loop called "build-measure-learn" to minimize project risk. Teams build minimum viable products (MVPs) and ship them quickly to gather feedback early on and use that feedback to improve the product continuously.

The work is organized in a "build, measure, learn" loop that lets the team iterate and improve the designs based on user feedback. Because the entire team is involved in the process, there is a lot of learning and sharing of information, preventing the emergence of silos and gurus.
The Lean UX method follows a list of 12 principles; through them, we understand that a Lean UX team is cross-functional and focuses on user research to set expectations and achieve the team's objectives. These teams also strive to avoid silos and rockstars; all the knowledge and information should be shared between team members.
But what is the best way to integrate Lean UX and Agile development? Gothelf's team tried using the "one sprint ahead" method but preferred a different approach. His team opted to use a pattern that integrates UX activities with Scrum rather than having them separated on two different tracks.

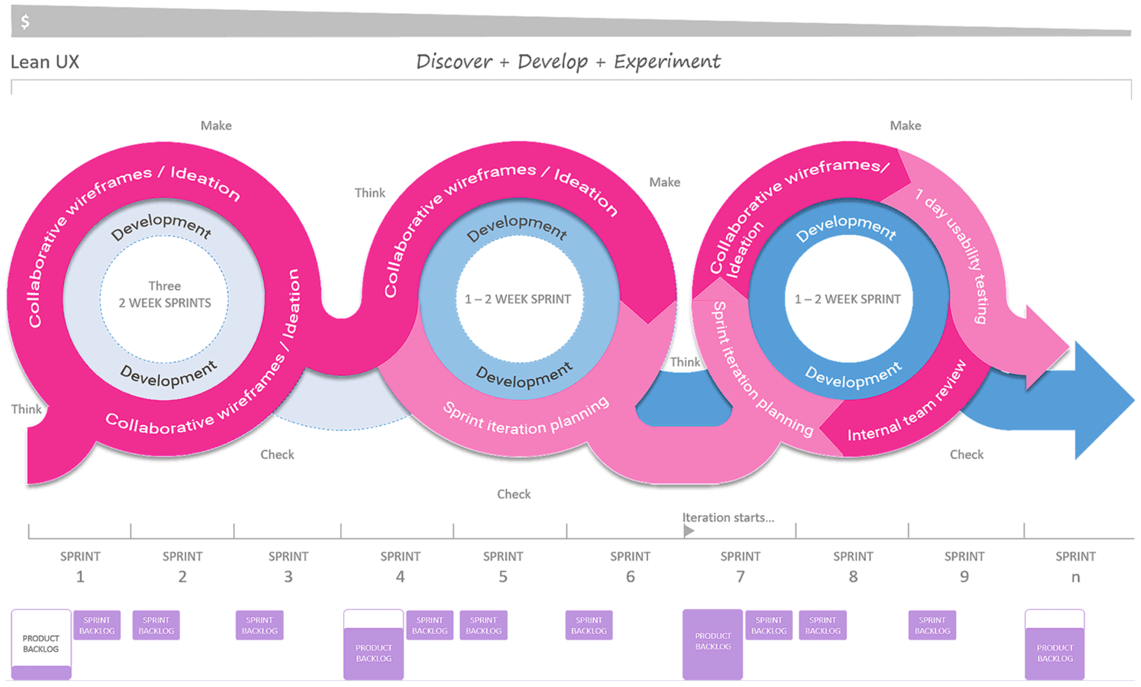


Figure 7: Lean UX and Agile development

Here is a picture from Deepak's article of how this integration is achieved [DE10]. From the picture, we can see that from the UX point of view, sprints either concentrate on the collaborative creation of wireframes or usability testing and team review. While the team, developers included, participate in creating the wireframes, the developers will work on implementing the selected sprint backlog stories.

This integration still presents some issues, mainly the focus required by the team to carry out many concurrent tasks and the high level of communication needed.

3.8 Conclusion

I have highlighted the most common issues and solutions proposed by researchers and experts in the field. There is no “one size fits all” method; the best approach for a team or company would be to start where they feel comfortable and confident working and apply small incremental changes to adapt their chosen method to their needs.
A team and the processes they adopt should not be seen as unchanging and static through time, but they should strive to improve their operations constantly.

4. The Balsamiq Way

In this chapter, I will explain how we work at Balsamiq, and following an actual feature project, I will illustrate the process in depth, analyzing the pros and cons of it. I will also bring the point of view of some of my colleagues I interviewed.

4.1 What is Balsamiq?

Balsamiq is a company comprised of 35 employees. We develop and sell one software, Balsamiq Wireframes, a design tool for creating wireframes or low-fidelity prototypes of user interfaces. Our motto is:
"Life is too short for bad software."

To help rid the world of bad software, we take many initiatives, including creating content for the Balsamiq Wireframing Academy, a website full of articles and videos on how to improve the UX of a product.

4.1.1 My role at Balsamiq

I started working at Balsamiq in 2015, and I work on Balsamiq Wireframes, but I have also worked on the old version of Wireframes, called Mockups.
Regarding Balsamiq Wireframes, I have worked on the Cloud and Windows desktop applications, at the same time if a feature needed to be implemented on both platforms.
I am part of an eight-person team: one manager, six developers, and one quality assurance expert.
I mainly interact with people from my team during my daily work, but if I am working on a feature that requires me to change the user interface, I will interact with the designer assigned to that feature.

4.2 How we work

In this section, I illustrate our current process for design and development.
An integral part of our process is "kaizen," a philosophy that encourages continuous improvement. Kaizen means we constantly gather feedback on our operations and work

on improving them. You can see a kaizen-related project in the quarterly organization picture (section 4.2.3) titled “Adjust how we work based on new Desktop strategy (…).” Most of the improvements are about the process we have in place for developing new features. But we have also worked on improving the roadmap creation process.

4.2.1 Overview

Balsamiq does not have external investors, so we can choose what features or issues we want to implement. Instead, the owner is in charge of deciding the company’s focus.

While we develop one software product, it is available through multiple platforms, from our Cloud application to a Mac OS and a Windows desktop application to various plugins (Atlassian, Google Docs, and more).
We have divided the codebase into two parts to avoid writing the same logic for each software version. The “Core” contains all the platform-independent logic, all the different product versions use it, and the “Natives” have all the platform-dependent code.

I will now illustrate how a development project is planned and executed at Balsamiq. First, I will explain at a higher level how we design and implement a feature, and then I will delve deeper into the organization of a project.
Employees are divided into groups based on their area of expertise:

- The “Editor” group includes all the developers working on the product’s Core, Mac OS, and Windows Native products.
- The “UX” team includes Design and User Research experts.
- Support: contains people who interact with the customers the most and can gather much feedback.

Project planning is done every three months. During this meeting, the managers from the different groups discuss what projects each group thinks are essential to work on during the quarter. At the end of the session, the participants create a tentative list of features and projects.

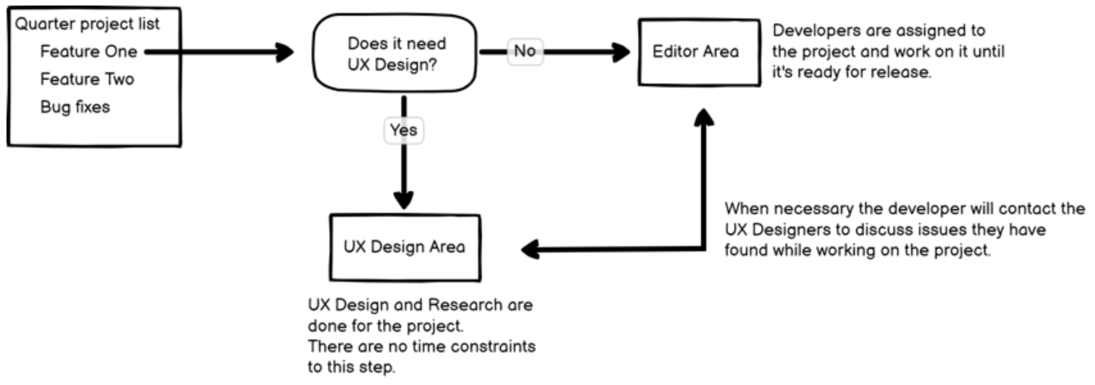

Quarter project list
Feature One
Feature Two
Bug fixes
Does it need UX Design?
No
Yes
Editor Area
Developers are assigned to the project and work on it until it's ready for release.
UX Design Area
When necessary the developer will contact the UX Designers to discuss issues they have found while working on the project.
UX Design and Research are done for the project.
There are no time constraints to this step.

Figure 8: Overview of the Design and Development steps at Balsamiq

For each new feature project, we have set a process to follow, composed of the following steps:

1. Preparation or project setup. The project wiki page is created and filled with the available information.
2. Discovery. This step aims to fully understand the problem we want to solve by looking at all the information sources (issue tracker, instant messages, and customer emails) to gather the requirements.
    a. The UX Research expert decides if we need to involve the users to validate the requirements.
    b. We researched possible architectural restrictions the project might have.
3. Design. The designer creates the needed documentation, which might be one or more of the following: user flow scenarios, wireframes, and visual design comps.
4. Compliance check. The project lead reviews security and GDPR to verify that we treat user information correctly.
5. Initial Development. The initial development phase is where the developers work on the feature.
6. Acceptance testing. During this phase, the feature is tested internally by developers, testers, and UX Designers. We use this time to gather feedback and requests for changes. If we introduce new requirements, the UX Area members must validate them before moving to the final development step.
7. Final Development. The development team implements the requested changes. If development for the desktop platforms is required, they do it during this phase as the primary logic resides in the Core shared codebase. It is best to work on the native platforms after the work on the Core is completed to avoid unnecessary refactorings.
8. Testing to ensure the new feature is solid and has no significant issues.
9. Release. During this phase, we train our support team, write documentation on the new feature, and then release the new product version.
10. Clean up. The goal is to ensure easy maintenance in the future. In addition, the project members update the internal knowledge base and fill out a retrospective to understand what worked well and what we can do to improve our process.

It can look like a long list of steps, but the workload of following them is shared between different people, which works best for us right now. In addition, it is part of the Balsamiq culture to improve and adapt our processes, so what works for us right now is not identical to what we did two years ago and is likely to change in the following years.

The documentation produced during the project is minimal, and it is only during the release and clean-up steps that we create more in-depth internal documentation.

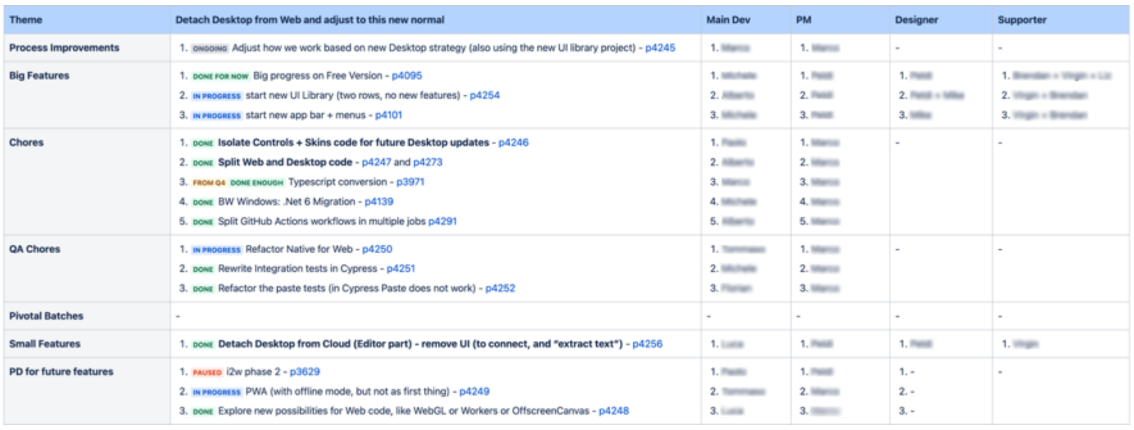
Due to the size and complexity of the code base, new features can require a longer development time compared to the standard 2-4 weeks cycles of SCRUM and other Agile patterns. Nevertheless, we still resonate strongly with the 12 principles in the Agile manifesto. I consider our development process Agile because the steps of Acceptance testing and final development are repeated until the feature has the required level of quality. We must remember that while Agile values being fast, its most important objective is delivering valuable features.

4.2.2 Long term vision

The owner of Balsamiq and product owner dictates the long-term product vision, an idea of what we want to achieve in the next few years. To do so, he looks at where the product is right now, customer feedback, and our competitors, and does market research to understand what the users need.
He shares an updated product vision twice a year, giving a general idea of his plans for the next three to five years and then diving more in-depth into the current year.

4.2.3 Quarterly organization

At the end of each quarter, each group (Developers, QA, Support, and UX) gets together, reviews what has been done in the last quarter and what is not yet finished, and makes a list of features they would like to see implemented in the next quarter.
After everyone has made their list, the team managers and the product owner meet to finalize what projects we will work on during the next three months.



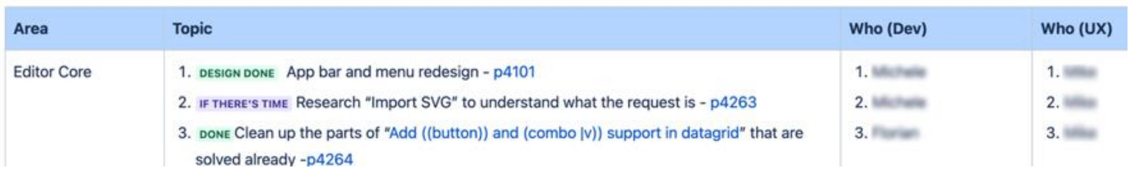
Theme	Detach Desktop from Web and adjust to this new normal	Main Dev	PM	Designer	Supporter
Process Improvements	1. ONGOING Adjust how we work based on new Desktop strategy (also using the new UI library project) - p4245	1. [redacted]	1. [redacted]	-	-
Big Features	1. DONE FOR NOW Big progress on Free Version - p4095 2. IN PROGRESS start new UI Library (two rows, no new features) - p4254 3. IN PROGRESS start new app bar + menus - p4101	1. [redacted] 2. [redacted] 3. [redacted]	1. [redacted] 2. [redacted] 3. [redacted]	1. [redacted] 2. [redacted] 3. [redacted]	1. [redacted] 2. [redacted] 3. [redacted]
Chores	1. DONE **Isolate Controls + Skins code for future Desktop updates** - p4246 2. DONE **Split Web and Desktop code** - p4247 and p4273 3. FROM Q4 DONE ENOUGH Typescript conversion - p3971 4. DONE BW Windows: .Net 6 Migration - p4139 5. DONE Split GitHub Actions workflows in multiple jobs p4291	1. [redacted] 2. [redacted] 3. [redacted] 4. [redacted] 5. [redacted]	1. [redacted] 2. [redacted] 3. [redacted] 4. [redacted] 5. [redacted]	-	-
QA Chores	1. IN PROGRESS Refactor Native for Web - p4250 2. DONE Rewrite Integration tests in Cypress - p4251 3. DONE Refactor the paste tests (in Cypress Paste does not work) - p4252	1. [redacted] 2. [redacted] 3. [redacted]	1. [redacted] 2. [redacted] 3. [redacted]	-	-
Pivotal Batches	-	-	-	-	-
Small Features	1. DONE **Detach Desktop from Cloud (Editor part) - remove UI (to connect, and "extract text")** - p4256	1. [redacted]	1. [redacted]	1. [redacted]	1. [redacted]
PD for future features	1. PAUSED i2w phase 2 - p3629 2. IN PROGRESS PWA (with offline mode, but not as first thing) - p4249 3. DONE Explore new possibilities for Web code, like WebGL or Workers or OffscreenCanvas - p4248	1. [redacted] 2. [redacted] 3. [redacted]	1. [redacted] 2. [redacted] 3. [redacted]	1. - 2. - 3. -	-

Figure 9: Organization of a quarter at Balsamiq

The above picture is the roadmap for our development team’s first quarter of 2023. As you can see, the list of projects is divided by kind. Each project has assigned the main developer, a project manager, a designer if the project requires one, and who is involved from the user support team, if necessary.
Not all projects need a designer; some focus on the backend of the application or on refactoring, like all the chores listed in the picture.

Each project has a dedicated project page, where you can read all the project updates, find links to the documentation, and see who is currently in charge of taking the project forward.
The research and design are already completed for all the projects listed in the picture.

Projects that involve changing the UI have a designer assigned to them. The designer will answer the developer's questions during development, update the designs if necessary, and review the changes during testing.
The UX and research team has a similar table for the projects they plan to do during the quarter.



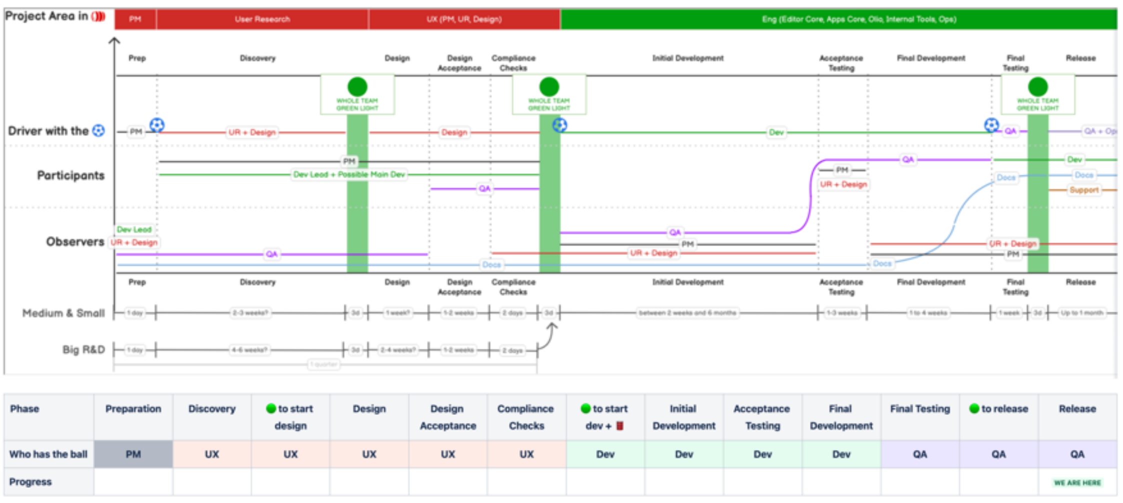
Area	Topic	Who (Dev)	Who (UX)
Editor Core	1. DESIGN DONE App bar and menu redesign - p4101 2. IF THERE'S TIME Research "Import SVG" to understand what the request is - p4263 3. DONE Clean up the parts of "Add ((button)) and (combo |v)) support in datagrid" that are solved already -p4264	1. [redacted] 2. [redacted] 3. [redacted]	1. [redacted] 2. [redacted] 3. [redacted]

Figure 10: Research and Design projects scheduled for a quarter.

As you can see, each project is assigned a developer. During the brainstorming and design phases, the developer will help the designer by highlighting possible problems and constraints created by the architecture.
After the roadmap is finalized, the development team meets again to determine project staffing and timing.

4.2.4 Project Organization

From the beginning to the release, I will illustrate how a feature project is done, focusing on the UX and development tasks. In the following picture, you can see a diagram of all the phases and who is involved.



Phase	Preparation	Discovery	🟢 to start design	Design	Design Acceptance	Compliance Checks	🟢 to start dev + 🟥	Initial Development	Acceptance Testing	Final Development	Final Testing	🟢 to release	Release
Who has the ball	PM	UX	UX	UX	UX	UX	Dev	Dev	Dev	Dev	QA	QA	QA
Progress													WE ARE HERE

Figure 11: Overview of the phases of a project

It can be a bit confusing when you see it for the first time; that is why I will follow the documentation we created for an actual project I have worked on to explain all the phases better. It is essential to notice that while the project is split into many steps, most are short and not complex. Discovery, Design, and Development are the most extended phases.

The project I chose had the goal of reducing the icons in the toolbar from this:

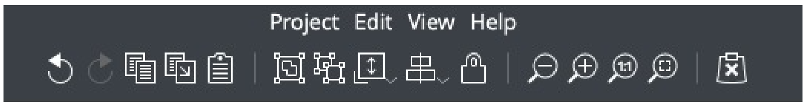


Figure 12: The old toolbar

To this:

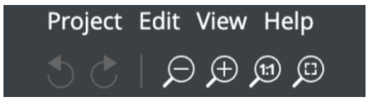


Figure 13: The new toolbar

This was a small change from a developer perspective, but from the user experience point-of-view, this is a significant change since the toolbar is visible and used by all users. Therefore, this project required careful consideration of which buttons we wanted to remove and which were necessary.

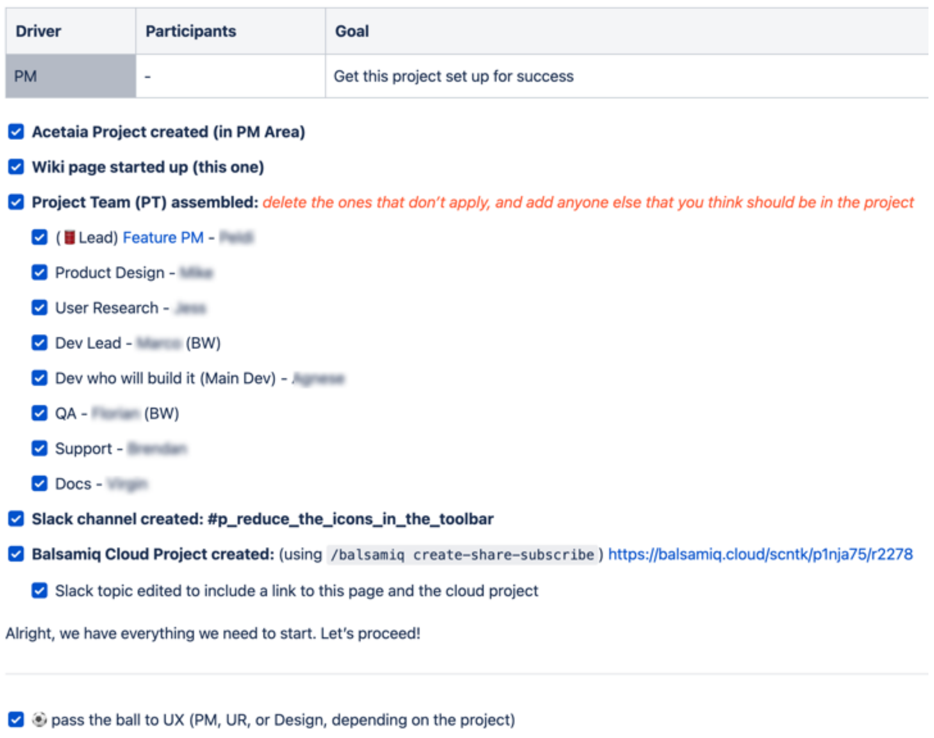
Preparation phase

During the preparation phase, the feature project manager (PM) creates the following:

- A page inside our project tracker application (called Acetaia).
- A wiki page that contains all the phases and steps we follow through a project. You will see screenshots of this page during this chapter.
- A channel in our messaging app (Slack) for ease of communication.
- A Balsamiq wireframes cloud project to iterate on the wireframes.

Then, they will assemble the project team (PT), which will include the following:

- Feature project manager.
- Product design: The UX and UI designer.
- User research expert: They will decide how much and what kind of research is necessary for the project, do the research, and take the results back to the team.
- Dev lead: This is the manager of the development group involved. The dev lead is not necessarily the developer that will implement the feature.
- The main developer: The main developer will oversee taking the development forward. For smaller projects, they might be the only developer involved, but for a more complex project, there will be multiple developers on the project team.
- QA: quality assurance.
- Support group: which support groups (tech, sales, docs) must be involved in the project and trained on the feature.



Driver	Participants	Goal
PM	-	Get this project set up for success

- [x] **Acetaia Project created (in PM Area)**
- [x] **Wiki page started up (this one)**
- [x] **Project Team (PT) assembled:** *delete the ones that don't apply, and add anyone else that you think should be in the project*
  - [x] (Lead) Feature PM - [redacted]
  - [x] Product Design - [redacted]
  - [x] User Research - [redacted]
  - [x] Dev Lead - [redacted] (BW)
  - [x] Dev who will build it (Main Dev) - [redacted]
  - [x] QA - [redacted] (BW)
  - [x] Support - [redacted]
  - [x] Docs - [redacted]
- [x] **Slack channel created: #p_reduce_the_icons_in_the_toolbar**
- [x] **Balsamiq Cloud Project created:** (using `/balsamiq create-share-subscribe`) https://balsamiq.cloud/scntk/p1nja75/r2278
  - [x] Slack topic edited to include a link to this page and the cloud project

Alright, we have everything we need to start. Let's proceed!

- [x] ⚽ pass the ball to UX (PM, UR, or Design, depending on the project)

Figure 14: Preparation phase

When everything is set up, we move to the next phase and give the project ownership ("pass the ball") to the appropriate UX expert.

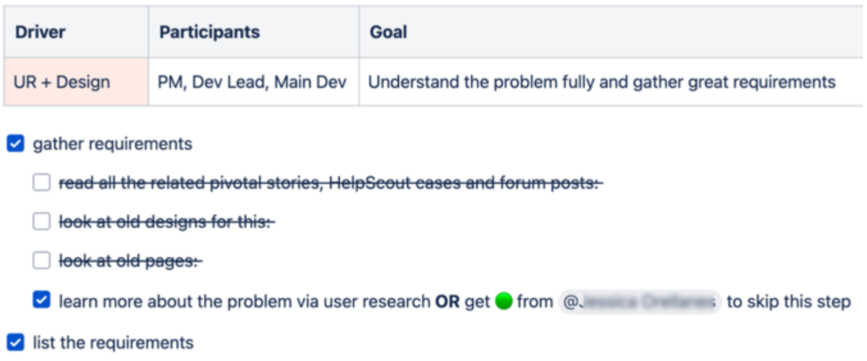
As you can see, this step is administrative. Its goal is to set up all the necessary documents and channels and to involve all the right people.
I want to highlight that the team size can look big, especially for a more straightforward feature, but not all team members are involved in all the steps. For example, eight people are part of the team assigned to this feature, but not all of them are involved all the time.

Discovery phase

The goal of this phase is to understand the problem fully and to gather all requirements. The steps include looking at all the information we have about the issue. Our knowledge sources are tickets, emails, and our forums. If we have any old design or research documentation, we will also review that.

The user research expert and the UX designer lead this phase. The UR and UX experts analyze the problem from the user's point of view while the developers bring up any constraints caused by the architecture.

When all the available information has been gathered, the User Research expert decides if further research is necessary to clarify the specifications.

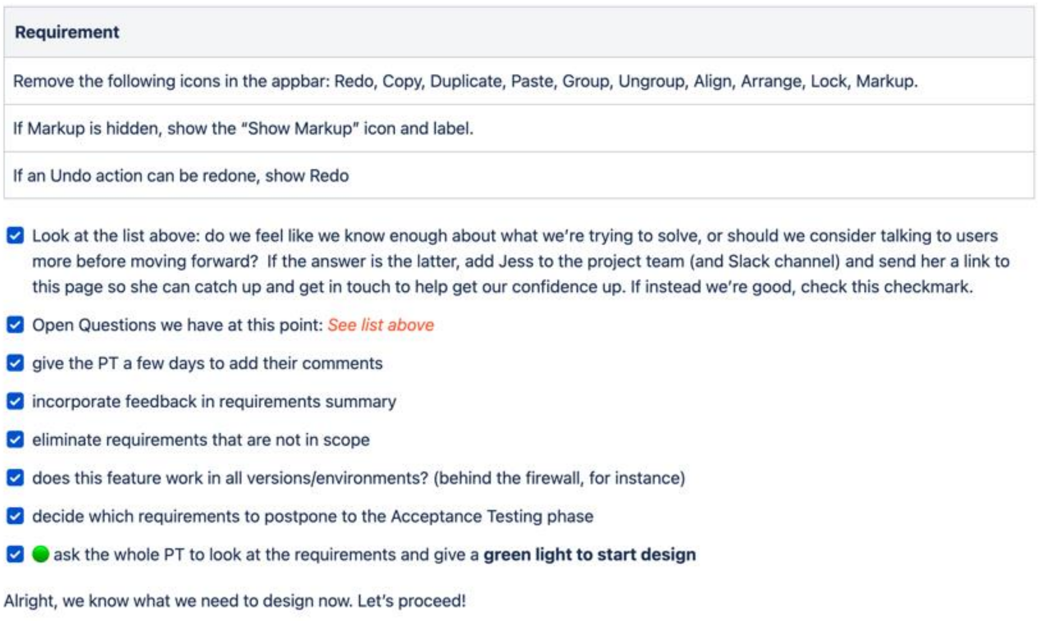


Driver	Participants	Goal
UR + Design	PM, Dev Lead, Main Dev	Understand the problem fully and gather great requirements

- ☑ gather requirements
  - ☐ ~~read all the related pivotal stories, HelpScout cases and forum posts:~~
  - ☐ ~~look at old designs for this:~~
  - ☐ ~~look at old pages:~~
  - ☑ learn more about the problem via user research **OR** get ● from @[redacted] to skip this step
- ☑ list the requirements

Figure 15: Discovery phase preparation

The result of this phase is a list of requirements. Once the list is ready, the Driver gives time to the project team to provide feedback before moving to the research step.



Requirement
Remove the following icons in the appbar: Redo, Copy, Duplicate, Paste, Group, Ungroup, Align, Arrange, Lock, Markup.
If Markup is hidden, show the "Show Markup" icon and label.
If an Undo action can be redone, show Redo

- ☑ Look at the list above: do we feel like we know enough about what we're trying to solve, or should we consider talking to users more before moving forward? If the answer is the latter, add Jess to the project team (and Slack channel) and send her a link to this page so she can catch up and get in touch to help get our confidence up. If instead we're good, check this checkmark.
- ☑ Open Questions we have at this point: *See list above*
- ☑ give the PT a few days to add their comments
- ☑ incorporate feedback in requirements summary
- ☑ eliminate requirements that are not in scope
- ☑ does this feature work in all versions/environments? (behind the firewall, for instance)
- ☑ decide which requirements to postpone to the Acceptance Testing phase
- ☑ ● ask the whole PT to look at the requirements and give a **green light to start design**

Alright, we know what we need to design now. Let's proceed!

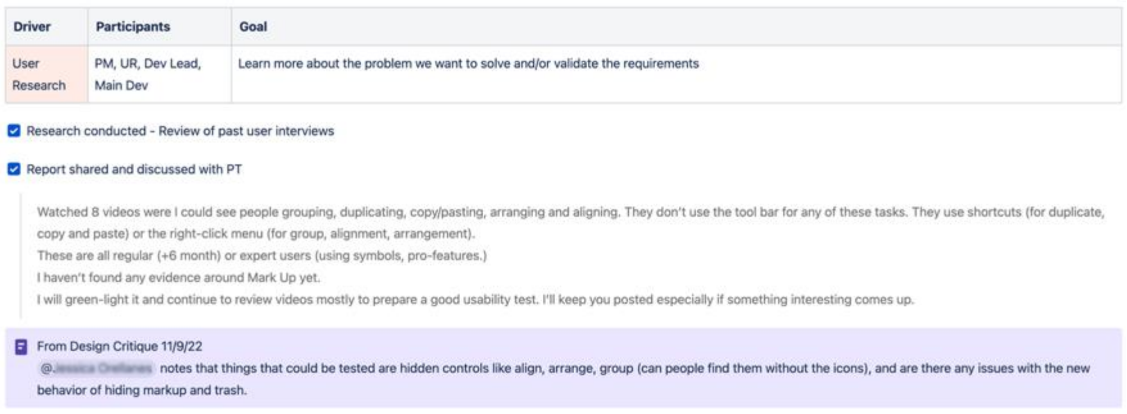
Figure 16: Discovery phase results

Research phase

This phase aims to learn more about the issue we are trying to solve and validate the requirements produced in the previous step.

The user research expert will carry out the needed research. It is up to them to decide the appropriate approach to this step.

Based on the research results, the requirements are either validated or need to be updated. Afterward, the project will move to the design phase.



Driver	Participants	Goal
User Research	PM, UR, Dev Lead, Main Dev	Learn more about the problem we want to solve and/or validate the requirements

- [x] Research conducted - Review of past user interviews
- [x] Report shared and discussed with PT

> Watched 8 videos were I could see people grouping, duplicating, copy/pasting, arranging and aligning. They don't use the tool bar for any of these tasks. They use shortcuts (for duplicate, copy and paste) or the right-click menu (for group, alignment, arrangement).
> These are all regular (+6 month) or expert users (using symbols, pro-features.)
> I haven't found any evidence around Mark Up yet.
> I will green-light it and continue to review videos mostly to prepare a good usability test. I'll keep you posted especially if something interesting comes up.

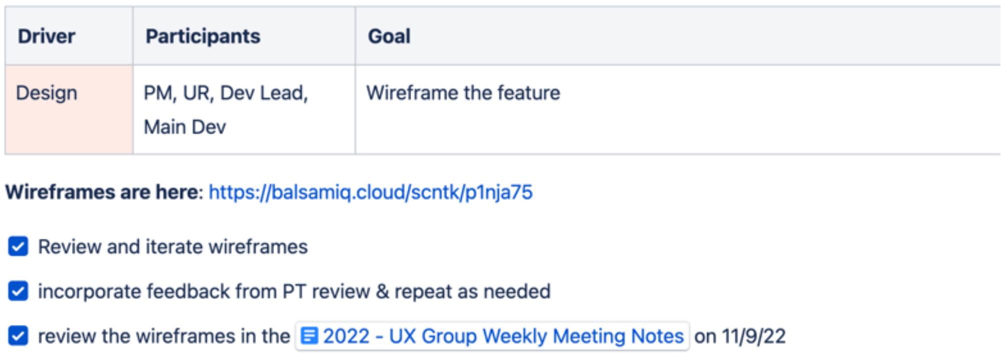
From Design Critique 11/9/22
@[redacted] notes that things that could be tested are hidden controls like align, arrange, group (can people find them without the icons), and are there any issues with the new behavior of hiding markup and trash.

Figure 17: Research phase

Design phase

During the design stage, the designer will create a wireframe of the feature, and they will iterate on it with the help and input of the other participants.

If the feature requires higher fidelity prototypes, they are usually done once the iteration on the wireframe is finished.



Driver	Participants	Goal
Design	PM, UR, Dev Lead, Main Dev	Wireframe the feature

Wireframes are here: https://balsamiq.cloud/scntk/p1nja75

- [x] Review and iterate wireframes
- [x] incorporate feedback from PT review & repeat as needed
- [x] review the wireframes in the 2022 - UX Group Weekly Meeting Notes on 11/9/22

Figure 18: Design phase

As shown in the picture below, we only needed a simple wireframe for this feature. It has the old version of the toolbar and the proposed new version where most icons have been removed or hidden.

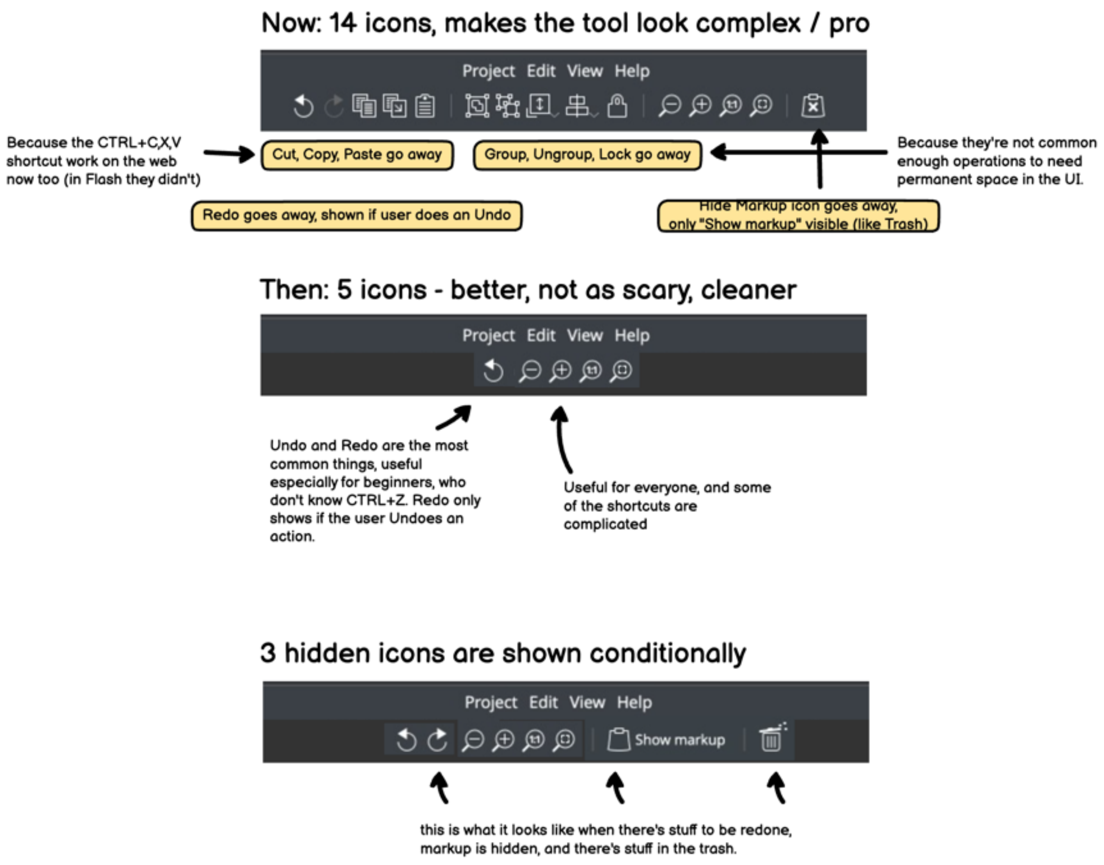
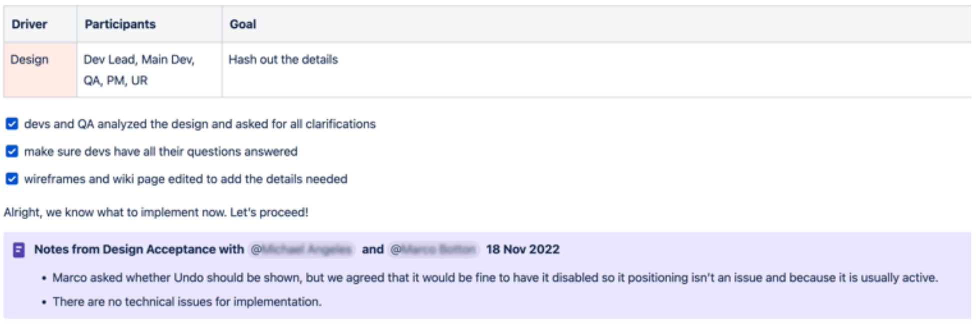

Now: 14 icons, makes the tool look complex / pro
Project Edit View Help
Because the CTRL+C,X,V shortcut work on the web now too (in Flash they didn't)
Cut, Copy, Paste go away
Group, Ungroup, Lock go away
Because they're not common enough operations to need permanent space in the UI.
Redo goes away, shown if user does an Undo
Hide Markup icon goes away, only "Show markup" visible (like Trash)
Then: 5 icons - better, not as scary, cleaner
Project Edit View Help
Undo and Redo are the most common things, useful especially for beginners, who don't know CTRL+Z. Redo only shows if the user Undoes an action.
Useful for everyone, and some of the shortcuts are complicated
3 hidden icons are shown conditionally
Project Edit View Help
Show markup
this is what it looks like when there's stuff to be redone, markup is hidden, and there's stuff in the trash.

Figure 19: The Wireframe created during the design phase

Design Acceptance phase

During this phase, the team analyzes, reviews, and approves the wireframe produced. The study is usually done asynchronously, and then we meet for the final review and acceptance.

If there are any notes from the meeting, they are recorded on the page, like in the picture below.



Driver	Participants	Goal
Design	Dev Lead, Main Dev, QA, PM, UR	Hash out the details

- [x] devs and QA analyzed the design and asked for all clarifications
- [x] make sure devs have all their questions answered
- [x] wireframes and wiki page edited to add the details needed

Alright, we know what to implement now. Let's proceed!

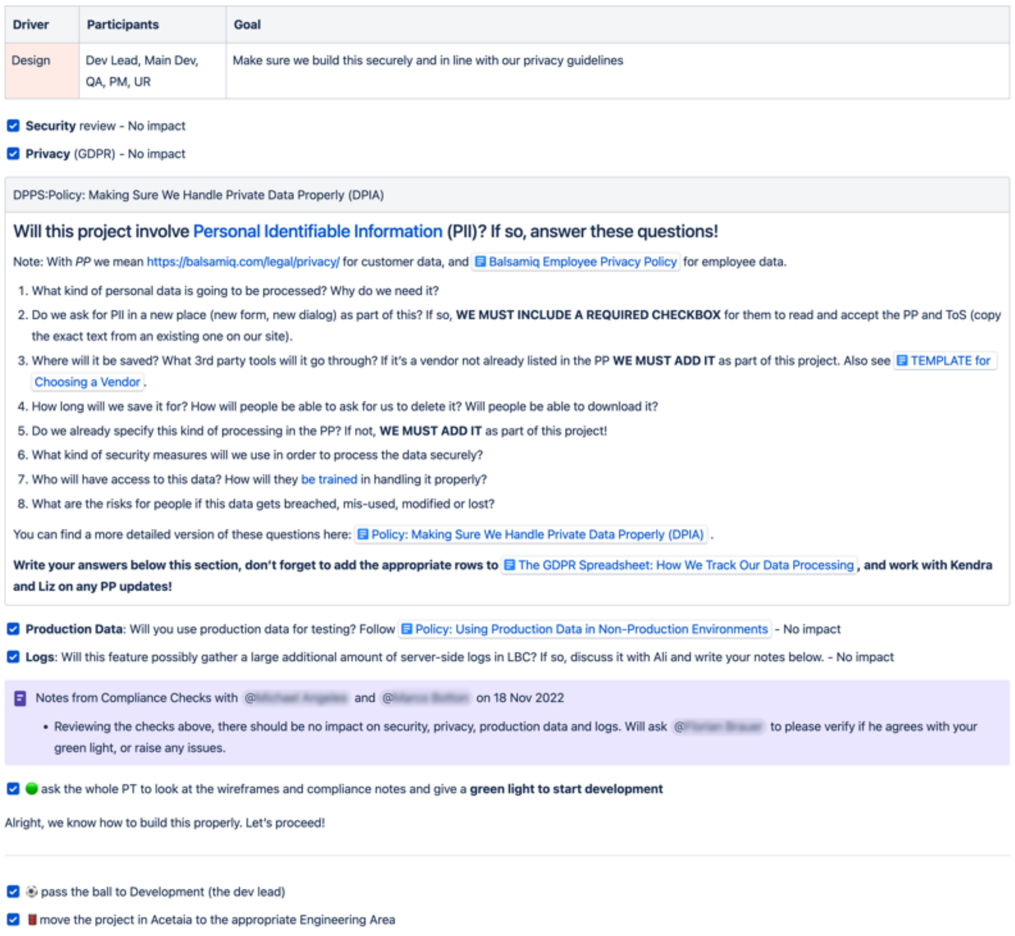
Notes from Design Acceptance with [redacted] **and** [redacted] **18 Nov 2022**

- Marco asked whether Undo should be shown, but we agreed that it would be fine to have it disabled so it positioning isn't an issue and because it is usually active.
- There are no technical issues for implementation.

Figure 20: Design Acceptance phase

Compliance Checks phase

This step is administrative. We ensure we align with our security standards and privacy guidelines (GDPR).



Driver	Participants	Goal
Design	Dev Lead, Main Dev, QA, PM, UR	Make sure we build this securely and in line with our privacy guidelines

- [x] **Security** review - No impact
- [x] **Privacy** (GDPR) - No impact

DPPS:Policy: Making Sure We Handle Private Data Properly (DPIA)

Will this project involve Personal Identifiable Information (PII)? If so, answer these questions!

Note: With *PP* we mean https://balsamiq.com/legal/privacy/ for customer data, and Balsamiq Employee Privacy Policy for employee data.

1. What kind of personal data is going to be processed? Why do we need it?
2. Do we ask for PII in a new place (new form, new dialog) as part of this? If so, **WE MUST INCLUDE A REQUIRED CHECKBOX** for them to read and accept the PP and ToS (copy the exact text from an existing one on our site).
3. Where will it be saved? What 3rd party tools will it go through? If it's a vendor not already listed in the PP **WE MUST ADD IT** as part of this project. Also see TEMPLATE for Choosing a Vendor.
4. How long will we save it for? How will people be able to ask for us to delete it? Will people be able to download it?
5. Do we already specify this kind of processing in the PP? If not, **WE MUST ADD IT** as part of this project!
6. What kind of security measures will we use in order to process the data securely?
7. Who will have access to this data? How will they be trained in handling it properly?
8. What are the risks for people if this data gets breached, mis-used, modified or lost?

You can find a more detailed version of these questions here: Policy: Making Sure We Handle Private Data Properly (DPIA) .

Write your answers below this section, don't forget to add the appropriate rows to The GDPR Spreadsheet: How We Track Our Data Processing**, and work with Kendra and Liz on any PP updates!**

- [x] **Production Data**: Will you use production data for testing? Follow Policy: Using Production Data in Non-Production Environments - No impact
- [x] **Logs**: Will this feature possibly gather a large additional amount of server-side logs in LBC? If so, discuss it with Ali and write your notes below. - No impact

Notes from Compliance Checks with [redacted] and [redacted] on 18 Nov 2022

- Reviewing the checks above, there should be no impact on security, privacy, production data and logs. Will ask [redacted] to please verify if he agrees with your green light, or raise any issues.

- [x] 🟢 ask the whole PT to look at the wireframes and compliance notes and give a **green light to start development**

Alright, we know how to build this properly. Let's proceed!

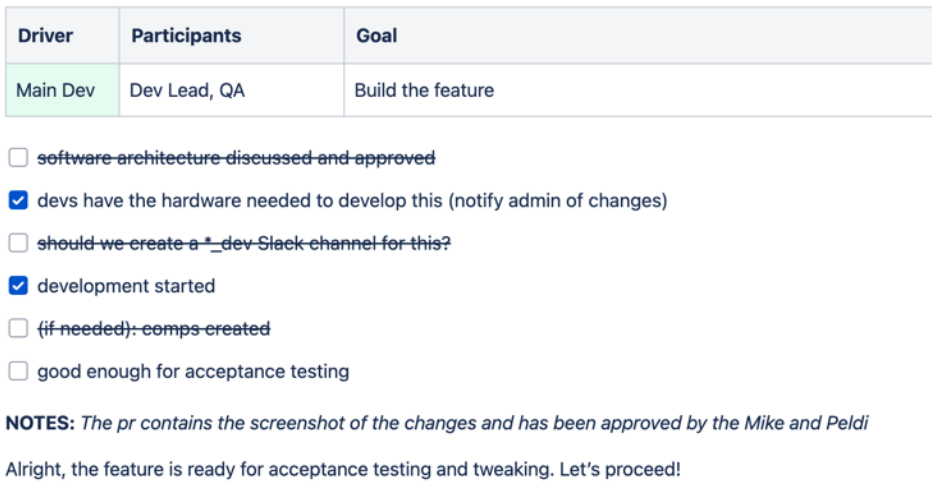
- [x] ⚽ pass the ball to Development (the dev lead)
- [x] move the project in Acetaia to the appropriate Engineering Area

Figure 21: Compliance Check phase

At this point, the "Research and Design" section of the project is complete. Therefore, the project is put on hold until scheduled for development in the following quarter.
While a finished design can be several weeks old before development starts, it will never be so old that the feature is no longer necessary or needs to be redesigned. Therefore, we always aim to do the research and design phases only when we know we will implement the feature as soon as possible.

Initial Development phase
The "Initial Development" phase is where the feature is developed.
We have a checklist to ensure everything has been discussed before development starts and to facilitate communication between developers, QAs, and design.
The first thing to do is evaluate if the developers need to change the architecture to implement the feature.
Once we are ready, the developers will create a Git branch and use it to implement the feature. We use Git and GitHub as our versioning systems.

Design is not listed as a participant in this phase but will be involved if necessary. What might happen is that some corner case has not been considered, or there are questions about the design. The developers will ask the designers to clarify or update the design. To ensure better communication, questions that arise during this step are usually shared via a messaging application; in our case, we use Slack.



Driver	Participants	Goal
Main Dev	Dev Lead, QA	Build the feature

- [ ] ~~software architecture discussed and approved~~
- [x] devs have the hardware needed to develop this (notify admin of changes)
- [ ] ~~should we create a *_dev Slack channel for this?~~
- [x] development started
- [ ] ~~(if needed): comps created~~
- [ ] good enough for acceptance testing

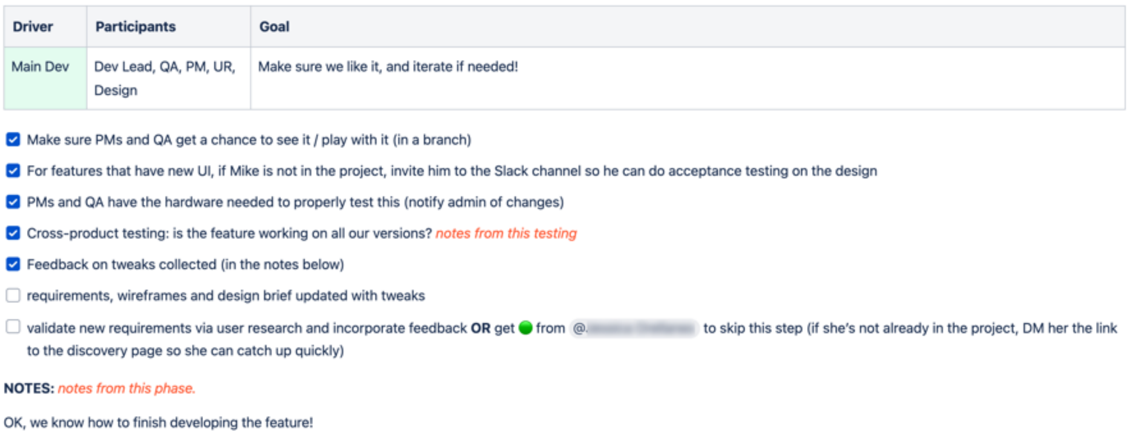
NOTES: *The pr contains the screenshot of the changes and has been approved by the Mike and Peldi*

Alright, the feature is ready for acceptance testing and tweaking. Let's proceed!

Figure 22: Initial development phase

Acceptance Testing + Tweaking phase
The lead developer ensures the project manager and QA can test the feature during this step. If they have set up a development environment, they can try the feature directly on their machine; otherwise, the developer can release a feature preview version of our cloud application.
We call a feature preview a custom staging build of our Balsamiq wireframes application created using the code from a GitHub branch; this allows our stakeholders to test the feature before we merge it into the master branch.

The PM and QAs test the feature to see if they like the UX and UI, and at the same time, they will try to find bugs.
They file a bug report in our bug tracker application for each issue they find. Sometimes, if there is time, they might require the developer to fix related bugs found and reported before the project started.



Driver	Participants	Goal
Main Dev	Dev Lead, QA, PM, UR, Design	Make sure we like it, and iterate if needed!

- [x] Make sure PMs and QA get a chance to see it / play with it (in a branch)
- [x] For features that have new UI, if Mike is not in the project, invite him to the Slack channel so he can do acceptance testing on the design
- [x] PMs and QA have the hardware needed to properly test this (notify admin of changes)
- [x] Cross-product testing: is the feature working on all our versions? *notes from this testing*
- [x] Feedback on tweaks collected (in the notes below)
- [ ] requirements, wireframes and design brief updated with tweaks
- [ ] validate new requirements via user research and incorporate feedback **OR** get 🟢 from @[redacted] to skip this step (if she's not already in the project, DM her the link to the discovery page so she can catch up quickly)

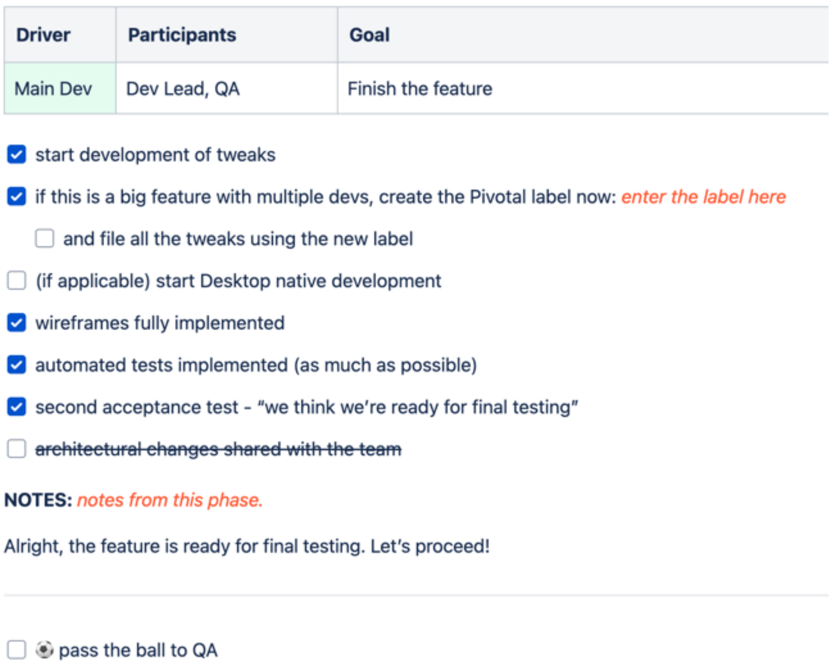
NOTES: *notes from this phase.*

OK, we know how to finish developing the feature!

Figure 23: Acceptance Testing and Tweaking phase

Final Development phase

During this phase, the developers implement all the requested changes and fix all the known issues. Afterward, another “Acceptance Testing + Tweaking” phase is done. The project keeps iterating between the “Final Development” and the “Acceptance Testing + Tweaking” phases until everyone is satisfied with the results.



Driver	Participants	Goal
Main Dev	Dev Lead, QA	Finish the feature

- [x] start development of tweaks
- [x] if this is a big feature with multiple devs, create the Pivotal label now: *enter the label here*
    - [ ] and file all the tweaks using the new label
- [ ] (if applicable) start Desktop native development
- [x] wireframes fully implemented
- [x] automated tests implemented (as much as possible)
- [x] second acceptance test - “we think we're ready for final testing”
- [ ] ~~architectural changes shared with the team~~

NOTES: *notes from this phase.*

Alright, the feature is ready for final testing. Let's proceed!

- [ ] ⚽ pass the ball to QA

Figure 24: Final development phase

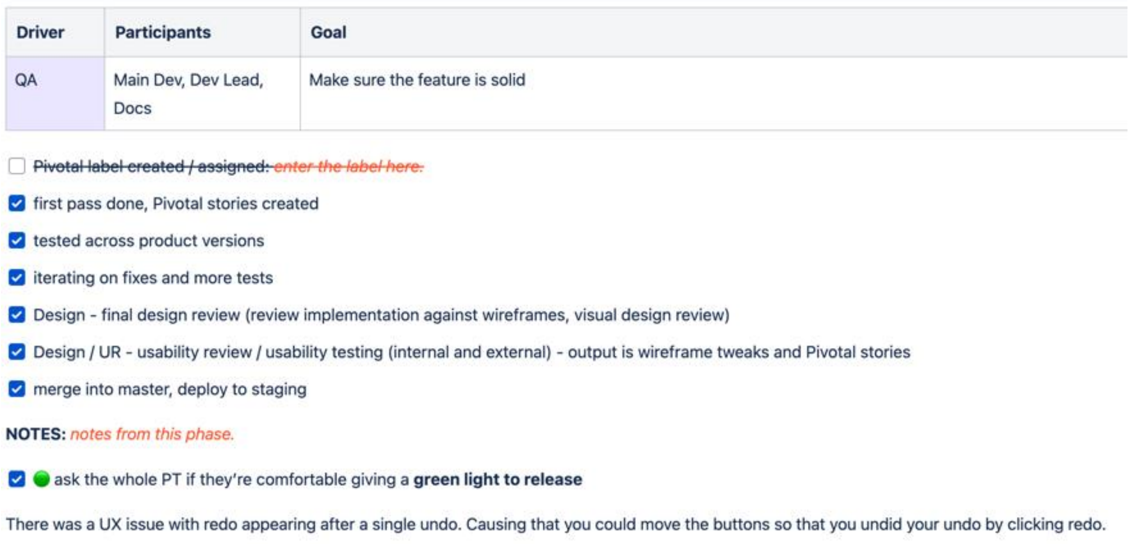
Then, a pull request is created, and developers who have not worked on the project will review and leave comments if necessary. Once the pull request is accepted, the code is merged into the master branch.

From my experience developing new features, the acceptance testing and final development phases are not strictly separate. Instead, they happen simultaneously and repeat until the team is satisfied.
Developers will work on improving the feature while the PM and QA are still testing the previous feature preview. Then, when the team feels it is the right moment, a new feature preview is released to allow the team to test the latest improvements.

Once the developers fix all the issues, the git branch is merged into the master branch, and the project moves to the next step.

Final Testing phase
Once the code has been merged on the master branch, the QA will perform all the needed tests across all the product versions. Finally, the design team checks that the feature matches the design.

The feature is ready to be released at the end of this phase.



Driver	Participants	Goal
QA	Main Dev, Dev Lead, Docs	Make sure the feature is solid

- [ ] ~~Pivotal label created / assigned: *enter the label here.*~~
- [x] first pass done, Pivotal stories created
- [x] tested across product versions
- [x] iterating on fixes and more tests
- [x] Design - final design review (review implementation against wireframes, visual design review)
- [x] Design / UR - usability review / usability testing (internal and external) - output is wireframe tweaks and Pivotal stories
- [x] merge into master, deploy to staging

NOTES: *notes from this phase.*

- [x] 🟢 ask the whole PT if they're comfortable giving a **green light to release**

There was a UX issue with redo appearing after a single undo. Causing that you could move the buttons so that you undid your undo by clicking redo.

Figure 25: Final Testing phase

Release phase

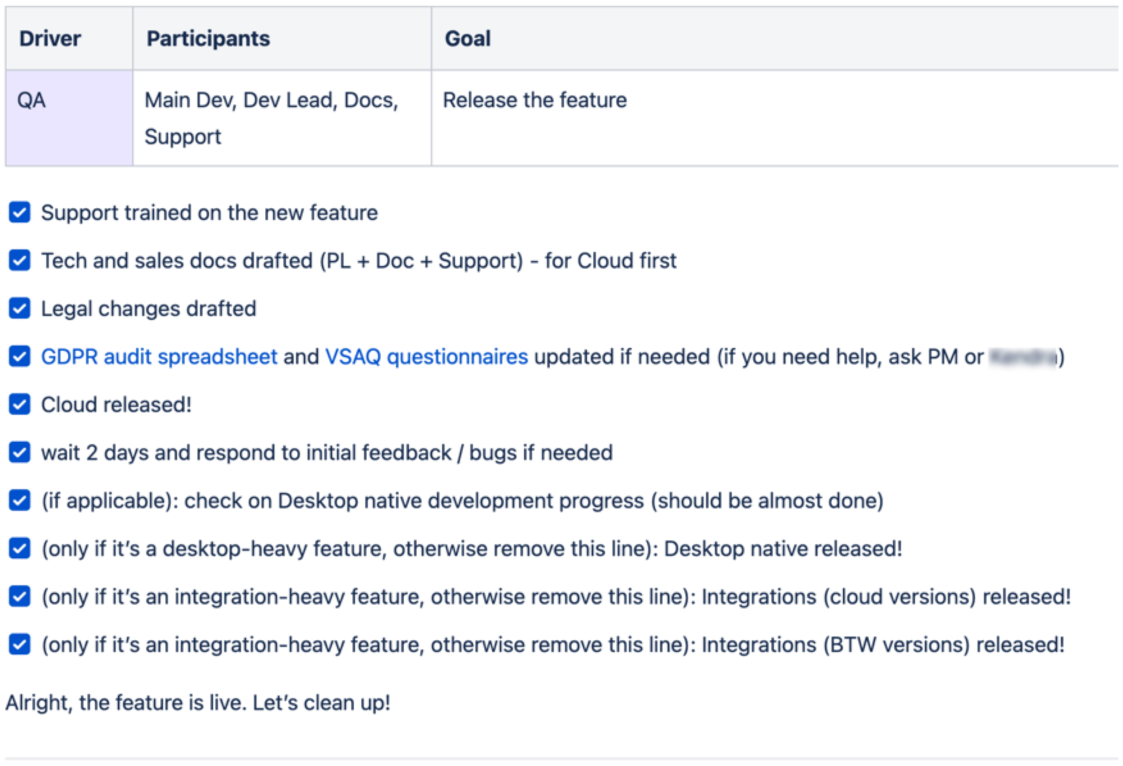
During the release phase, we get everything ready for the release day. Which means:

- Checking that any new documentation (for the customers) is prepared,
- Checking that all legal and privacy documents have been updated if needed.
- Checking that support is prepared and trained on the new feature.

We have a detailed list to avoid forgetting any essential steps.
Usually, we release once a month, so a new feature might be on the main git branch for a couple of weeks before it is deployed to the users.

We do not release all the versions of Balsamiq Wireframes on the same day; instead, we do staggered releases.

We release Balsamiq Wireframes for Cloud first, then wait a couple of days to ensure the release has no issues, and then we proceed with releasing the other products. We have chosen to do staggered releases because we can patch or revert the release faster on Cloud than on the other products if a release has issues.



Driver	Participants	Goal
QA	Main Dev, Dev Lead, Docs, Support	Release the feature

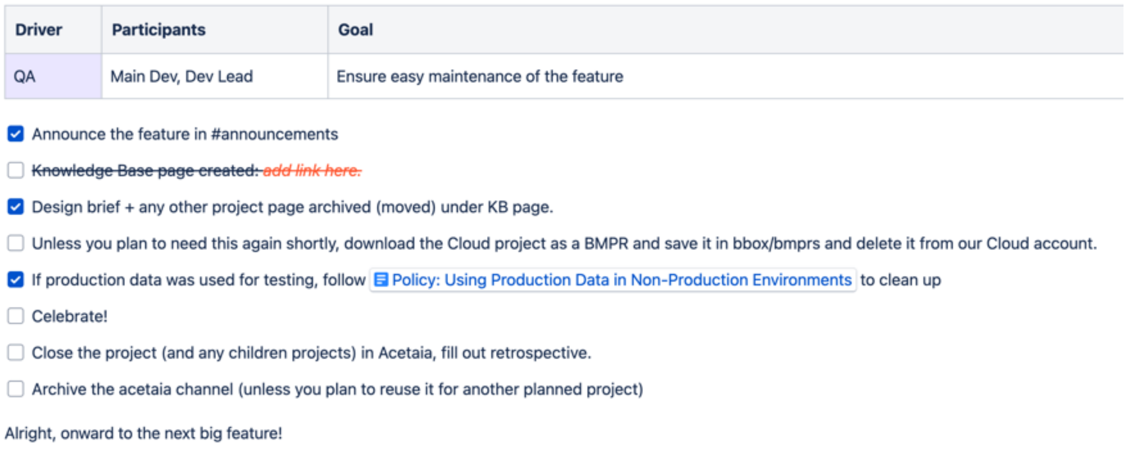
- ☑ Support trained on the new feature
- ☑ Tech and sales docs drafted (PL + Doc + Support) - for Cloud first
- ☑ Legal changes drafted
- ☑ GDPR audit spreadsheet and VSAQ questionnaires updated if needed (if you need help, ask PM or [redacted])
- ☑ Cloud released!
- ☑ wait 2 days and respond to initial feedback / bugs if needed
- ☑ (if applicable): check on Desktop native development progress (should be almost done)
- ☑ (only if it's a desktop-heavy feature, otherwise remove this line): Desktop native released!
- ☑ (only if it's an integration-heavy feature, otherwise remove this line): Integrations (cloud versions) released!
- ☑ (only if it's an integration-heavy feature, otherwise remove this line): Integrations (BTW versions) released!

Alright, the feature is live. Let's clean up!

Figure 26: Release phase

Clean Up phase

After the release, we have some clean-up tasks, like creating internal wiki pages, closing the project (in the wiki and in our internal project tracker), and closing the slack channel.



Driver	Participants	Goal
QA	Main Dev, Dev Lead	Ensure easy maintenance of the feature

- [x] Announce the feature in #announcements
- [ ] ~~Knowledge Base page created: add link here.~~
- [x] Design brief + any other project page archived (moved) under KB page.
- [ ] Unless you plan to need this again shortly, download the Cloud project as a BMPR and save it in bbox/bmprs and delete it from our Cloud account.
- [x] If production data was used for testing, follow Policy: Using Production Data in Non-Production Environments to clean up
- [ ] Celebrate!
- [ ] Close the project (and any children projects) in Acetaia, fill out retrospective.
- [ ] Archive the acetaia channel (unless you plan to reuse it for another planned project)

Alright, onward to the next big feature!

Figure 27: Clean Up phase

The most relevant step of the clean-up is the retrospective. When we close a project in Acetaia (our internal project tracker), a retrospective page is automatically created on the wiki, and everyone is invited to give feedback.

We ask everyone four questions:

- What went well?
- What could we have done better?
- Can you think of anything we could do short-term to improve things?
- How about the longer term?

This last step helps us identify any friction points we can improve on.

4.3 Are we agile?

Before analyzing what works well in our environment, we need to understand if we can consider our process Agile or if it is traditional.

By reading the Agile manifesto [BJR01], I would conclude we are agile because:

- While we have a defined set of tools for our process, we have changed them multiple times when we found they no longer fit us. We continuously check in to analyze how to improve and adapt our strategies to our needs.
- While we have some internal documentation, we only keep what we need to do a good job. We recognize that our focus is to create a great product and help our users design the best wireframes. Therefore, our processes allow us to write as much or as little documentation as necessary.
- Having excellent customer support is one of our strengths and goals. Our support team is always looking at ways to improve themselves.

- As you have seen at the project's beginning, we have a long-term plan that is not set in stone or with precise deadlines. Instead, we use it more as a compass to guide us.

We also follow most of the 12 agile principles:

Agile principle	What we do at Balsamiq
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	From the description of our process, we do continuous delivery. But since we do not have strict deadlines and we strive to deliver quality software, we will not release an incomplete feature, even if it means the development step will take longer than expected.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	We do entirely agree with this principle. We are not afraid to change things even during late development when requirements change.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	While a single feature can take over a month to implement, we deliver features and bug fixes each month. This timescale allows developers and QAs to work without being rushed.
Business people and developers must work together daily throughout the project.	Business people and developers work together as needed. The frequency will depend on the project. Since we have frequent and open communication about goals and expectations, we might get away with fewer synchronous interactions.
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Every developer is involved with the staffing of the project; this allows us to pick projects we find interesting. Projects are mainly self-managed; we trust that if a developer needs help, they will ask for it. On complex projects, we often do pair programming. When we encounter intricate architectural issues, we set up meetings to evaluate and discuss possible solutions.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	While we agree with this statement, we are a remote-first company. Many of our team members live outside of Italy, and even the Italians rarely visit the office.

	What we do is video calls as needed. We have team retreats once or twice a year, where the development team gets together. We use this time to discuss architectural problems or improvements.
Working software is the primary measure of progress.	We agree with this statement.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	One of our mottos is pace over deadlines. We give ourselves deadlines because we need to estimate how long a project will take, but if the estimate is wrong, we will push back the deadline. We do not encourage working overtime or weekends to meet deadlines.
Continuous attention to technical excellence and good design enhances agility.	All the employees can use part of their work hours to do “professional development time,” which can be taking a course, researching new technology, attending a conference, and more. Having the time to invest in professional development allows us to keep up to date with technology and continuously improve ourselves. Not having to rush to meet deadlines allows us the time to create a quality product in terms of design and architecture.
Simplicity--the art of maximizing the amount of work not done--is essential.	Yes, we keep our internal documentation minimal. We also try to have only the necessary meetings. Often, chat messages are enough to clear up any questions and do not disrupt your day as much as a video conference.
The best architectures, requirements, and designs emerge from self-organizing teams.	We agree with this statement. Each team can self-organize as they please. However, with more complex architectural problems, we try to involve all the developers to get everyone’s opinion and keep everyone up to date.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	We often do this. We do a retrospective after each project, evaluating what can be improved. Then, during our weekly development team meeting, we discuss

	friction points and try to figure out how to improve them. Sometimes, it takes a while to change a process that is not working, but it is okay as long as the change happens.

I interviewed the developer's team lead, the product owner, and the UX expert and asked them if they thought we were agile. I have gotten mixed answers, mainly a "kind of." Our UX expert said: "It is mostly Agile, but I have heard a repeated desire in the past to have more certainty about design requirements BEFORE starting development."

The dev lead, states, "I would say we use some Agile practices, but we are not fully Agile. For example, we tend to break the projects in small parts, "shipping" to master as often as we can, but we also rely on the discovery, design, user research, for all the time it is needed, before touching the code."

I would conclude that we are Agile; we might not fit in the description 100%, but I cannot say we are implementing a traditional methodology either since we do not spend enough time and effort on the analysis, research, and design phases.
This difference in opinion is due to the different levels of knowledge of Agile and how strict we believe the 12 principles to be. Since Agile is a philosophy, it can be interpreted in different ways.

Furthermore, we are adopting the "one cycle ahead" process, with the UX research and design done one quarter ahead of the development. The timings are much more expanded, compared to the process described in the academic papers, but it is what we can afford to do with our staff. This process works very well for us, and the research and UX design does not get stale.

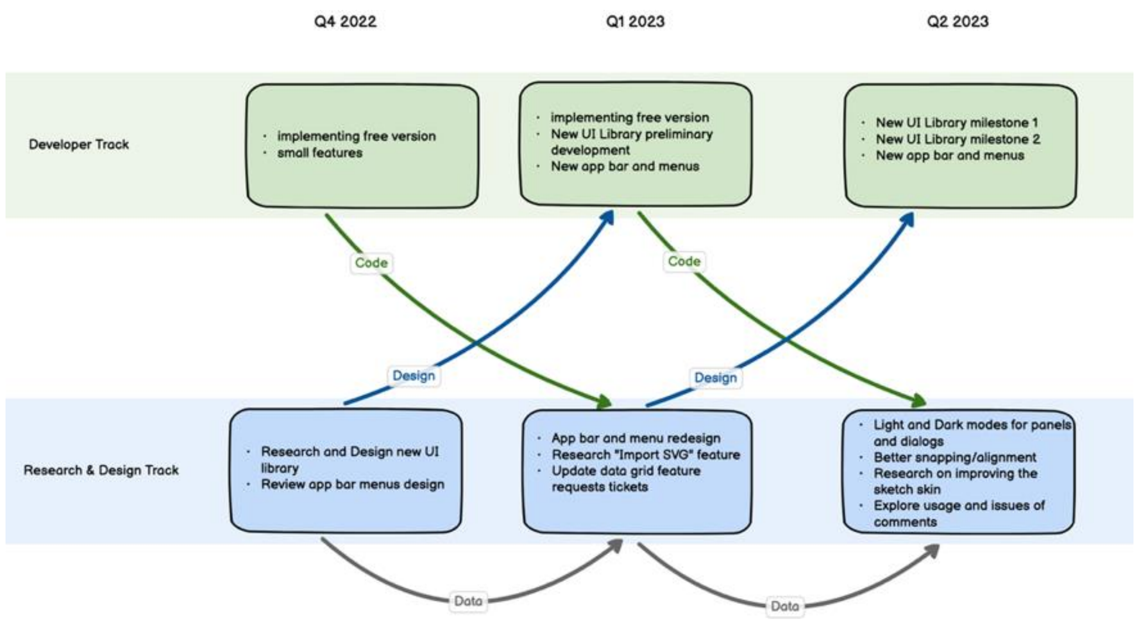

Q4 2022
Q1 2023
Q2 2023
Developer Track
implementing free version
small features
implementing free version
New UI Library preliminary development
New app bar and menus
New UI Library milestone 1
New UI Library milestone 2
New app bar and menus
Code
Code
Design
Design
Research & Design Track
Research and Design new UI library
Review app bar menus design
App bar and menu redesign
Research "Import SVG" feature
Update data grid feature requests tickets
Light and Dark modes for panels and dialogs
Better snapping/alignment
Research on improving the sketch skin
Explore usage and issues of comments
Data
Data

Figure 28: Three quarter development and design tracks overview

Having such time to carry out the UX tasks gives us a clear view and direction for the project, ensuring it fits well with the rest of the product. We do not want to rush and deliver half-baked features.

5. Assessing Our Development Process

We acknowledge that our development process is imperfect and work constantly to improve it. The version I have described works well for us, but we still have room for improvement. We genuinely believe in Kaizen, also known as continuous improvement, and we make a concrete effort to better ourselves and our processes.

5.1 Successes

Team coherence

We are not located in the same office, but having dedicated Slack channels allows for fast and asynchronous communication, which is what we need most of the time.
A written-down chat allows participants to ask questions and express opinions. The others can answer when they have time without getting distracted by a call or someone coming into your office to chat.
Designers and developers need deep focus to produce quality work, and even a tiny distraction can disrupt their attention.

At the same time, we don't avoid video calls; we use them as necessary. Sometimes, a "face-to-face" meeting can save time and frustration.
During meetings, we appoint a participant to take notes and add them to the project wiki page. These notes prevent the team from forgetting about the discussion and the conclusions reached, preventing us from re-discussing the same issue later on.
Having the same debate multiple times because we did not have notes has happened in the past, and it's a frustrating waste of time.

Scheduling

We have created our own project template that we follow for feature projects. We enjoy writing it down so we can easily follow the steps while ensuring we aren't forgetting anything.
At the same time, our process is flexible. We can propose changes when we find something that needs improvement. The ability to suggest tweaks reflects our Kaizen philosophy and decreases the friction that might arise when someone notices the process isn't working as smoothly as they'd like because they can propose changes.
Another positive side effect of having the template is that the template requests you design one person to fill each role necessary for the project. This means that the responsibilities

of each participant are clear, and we always know who is in charge of making sure the project moves forward (the person in charge changes through the different phases of the project).

Knowledge Base
We have created a culture of collaboration. When working on complex issues, the developers do pair programming. Pair programming has three main benefits: the programmers reach a better solution, two people are now experts in the piece of architecture they have worked on, and bugs and regressions are less likely.
When the UX experts work with the developers, there is no assumption from either side to be more important to the company than the other professional. This understanding allows for open and honest communication without managers acting as mediators.

5.2 Past challenges

The company has grown slowly through the years and has always hired professionals that would fit well with the company culture. This focus on maintaining our culture helped us avoid issues with team coherence since we were all aligned toward the same goals.
Instead, as we grew, we faced challenges related to our knowledge base.
We were developer-led for the company's first ten years when the founder would do UX design and development.
He did this without implementing UX research because he and two other UX experts interacted daily with our customers through multiple channels (e.g., support tickets, forums, Twitter…, etc.).
These interactions gave them enough confidence to plan and design the features the developers implemented.
But as time passed, they started noticing gaps in their knowledge, so they decided to become more intentional in doing UX research, first by conducting user interviews and then by having a team member get specialized training on UX research.
We also did not have a formal process to schedule features, so it happened that the UX research and design of a feature would be done too much in advance of implementation. In those cases, the UX design had to be discarded and redone because it would not fit anymore with the rest of our application.

We have developed the process I illustrated in the previous chapter to solve planning issues.
We have also introduced the Priority Review Board (PRB) meetings.
This meeting happens weekly with the following goals:

- Have a regular review of ongoing work with the right stakeholders to make sure everyone's in sync.
- Free up engineering time from making many priority decisions throughout the week.
- Keep the to-do list clean.

The developers lead, the QA lead, the support lead, and the project manager meet (via video call) to review the following:

- Review the size of the current Iteration / Backlog.
- Review new stories: Everyone in the company can add new stories to our bug tracker app. The PRB team reviews, sorts, and labels the latest stories every week.
- Pick stories for the next week based on resource availability, support lead input, and quarterly projects and goals.
- Review labels to make sure they are still clean. We use tags to categorize the issue tracker stories; they allow us to identify where we need to focus and to do "focus sprints" where we fix all the issues related to a specific label.

Before the priority review meeting was established, the developers had to pick what issues to work on. This is in the hands of the relevant stakeholders, who have all the available information to make informed decisions.

5.3 Current Challenges

Our company size, organization, and needs change over time, so it is natural that we still have many areas to improve.
These are the most pressing challenges we are facing.

Team coherence challenges
While everyone in the company is aligned toward common goals, we have high expectations and a long list of things we want to achieve. Still, our resources are limited regarding available staff and money.
We have clearly defined who is responsible for ensuring each phase of a project moves forward, but the expected outcome of a step or the desired level of involvement of a team member in each stage wasn't clear to everyone.

The developers recently highlighted that they used to work with the assumption that the requirements they get after the design phase is completed are final.
At the same time, the UX team viewed them as still flexible and not complete, as we do not have enough UX professionals with us to be able to spend the time to polish and refine a design to perfection.
The developers brought up the frustration of having to rework the implementation of a project multiple times due to the UX experts "changing their minds" on the design or not designing edge cases that later revealed the need to change the architecture.
Because the developers spoke up, the two teams realized that there was a mismatch of expectations: the developers expected the designs to be complete, while the UX team expected to be able to tweak and change the design through the development phase of a project.
We are currently testing a tweak in our process. The developers now have a more active role during the design phase by creating a "throwaway" implementation of the feature.

This throwaway version will implement the proposed design, allowing everyone to test it. Since this is not meant to be the actual implementation of the feature, writing low-quality code and making a mess is okay since we plan to re-implement the feature once the final design is tested and approved.

This new process has many benefits:

- We produce a testable feature quickly.
- Everyone can use the design and see how it feels in real use cases.
- It is easier to notice corner cases that have not been considered during design.
- The developers will notice where the current architecture needs to be altered and where any pain points are. And have the time to involve the rest of the developers if they think it is best to review the changes together.
- Since everyone knows this first implementation is meant to be thrown away, the developers are less frustrated when changes are requested, or new design parts are added.

We have just started using this new pattern, but it looks like everyone likes it, and it helps us to stay on the same page.
From a developer's perspective, knowing that the feature preview created during the design phase is not considered the actual implementation of the feature by the team is reliving and empowers us to try things out.

Another issue we face when working on a big project is that we have two people acting as UX experts: M. who is a trained UX designer and the company owner.
When the developers ask a question during such projects, the two UX experts might disagree on the answer, and to the developers, it is not always clear who has the power to make the final decision.
This issue is exacerbated by M. living in another time zone, meaning he will start answering questions towards the end of the developers' workday, while the owner can respond right away.
Because of this, the developers know that M. was not consulted prior to the answer and know M. might disagree with the answer given by the founder, leading to further discussions.
Unfortunately, we don't have a clear solution here.
Ideally, we would hire more people to work on the UX team, possibly in the same time zone as the developers; M. would be the coordinator as the senior UX designer, and the owner would act only as project manager, leaving the more nuanced details of the UX design in the hands of the designer.

Knowledge base challenges

We all agree about the value of UX; it is essential for our product:

- We aim to have a better user experience than our competitors.

- Balsamiq is a software used to design UX experiences, and it would be wrong if we had a bad UX.

But there is still room for improvement.
The UX team is small compared to the number of developers we employ.

Currently, the UX team is made up of 3 people:

- An UI designer, who is also a professional UX Designer.
- The owner who acts as UX Designer.
- A User Research expert.

That is a small team compared to the number of developers in our company: our 8-people team and another 7-people team.
This disparity stretched the UX team thin with the number of projects the developers could handle.
This issue seems to have improved since we changed the process, involving the developers more in the UX phases. As I said before, a solution would be to increase the number of UX experts in the company, but we do not plan to hire anyone soon.

Scheduling challenges
I believe we have gotten pretty good at scheduling features and estimating how long they'll take to implement.
We only start research and design on projects we plan to implement soon, and no one is left waiting for others. The developers do not spend time waiting on designs; they either work on other projects or participate in the design phases.

The only issue we are facing in this category is related to time zones. Unfortunately, our UX designer is located in the USA, while the developers are all in Europe. We only have one hour of overlap each day, which is often filled with other meetings. Face-to-face calls between the designer and the developers are rare, and that can be frustrating when the developers need clarifications on the design.
As I have said before, we ask questions in Slack, the instant messaging application we use, or create a wiki page. What usually happens is that the company owner answers the questions. Still, sometimes the UX designer has a different opinion, so the developers have to either start working before the UX expert can participate in the discussion or wait. The developers will choose to wait if they can work on other sides of the project (for example, on the architecture) or if they have other projects/bugs to work on; otherwise, they will start working with the information they have at that time and hope they do not have to go back and make changes.

5.4 Suggestions for a successful transition

In this thesis, I have analyzed the status of the Agile development patterns, with a focus on the integration of UX design methods.

Integrating UX research and design when developing in short iterations brings many challenges, but none of them are insurmountable.
Many companies have successfully merged UX design and Agile models, but true and seamless integration is always the result of hard work from everyone on the team.
Open, honest, and frequent communication is the stepping stone to achieving success.

In Chapter 3, I have highlighted the most important aspects of success according to the current literature.
Following those recommendations, these are the steps I would follow to ensure a successful addition of UX tasks to an Agile team.

Make sure everyone is on board.
Collaboration is the basis of success. Before you start making changes, set aside some time to gather requests, doubts, and questions from everyone. Then, address them to the best of your abilities.

Set up a feedback culture.
Ensure everyone knows how to give and receive feedback. And make sure the teammates know that feedback is a good and wanted thing. After all, it’s better to know that something isn’t working (or can be improved) sooner rather than later. But remember that feedback isn’t always negative; positive feedback is welcome too, and it boosts the morale of the team.
The best way to know if an aspect of your work needs work is through feedback.

At Balsamiq, we have several wiki pages about feedback, and we’ve also had some group training on it. Giving and receiving feedback is not easy (especially when it’s not positive feedback), so it’s essential to know how to do it. I would suggest having a resource page with resources the employees can use as a reference when they feel the need and for new employees to use as training.

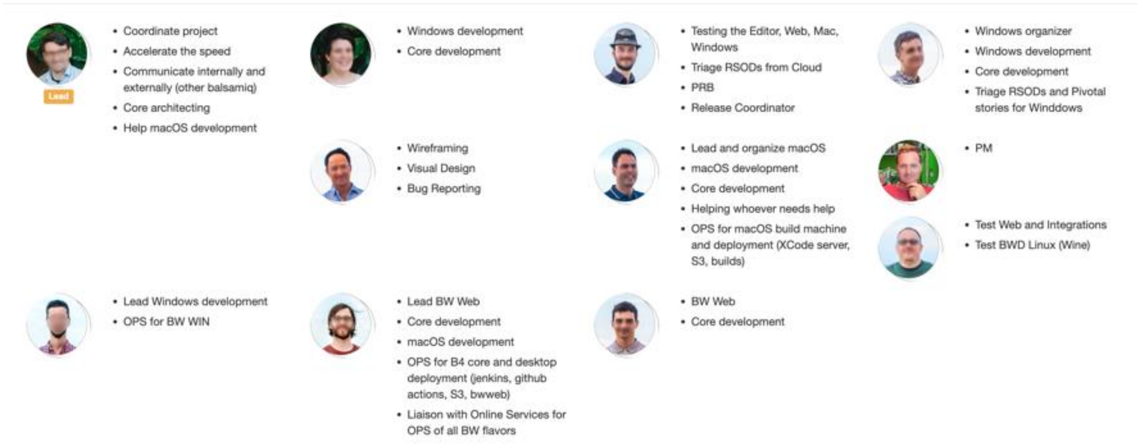
Align the knowledge base.
Ensure everyone understands the goals of Agile development and the importance of focusing on UX design. If there are experts on these topics in the team, ask them to give a short presentation and answer any questions that might arise.
Be sure to record the most important information in the wiki in case someone wants to revisit them or for future teammates.
Try to find ways to inspire employees to share information with eachother continuously. At Balsamiq, we have each team give a short presentation about what they’ve been working on to the rest of the company every two months. And we have company experts give more in-depth lectures at the company retreats.

Pick a software development approach.
Now that you have laid a solid foundation, you can start making changes.

Evaluate if your current Agile model can be adapted to ensure there is enough time to carry out UX tasks. At Balsamiq, we're using the one sprint ahead model, slightly edited to better fit our needs. But it was not always like this; we have reached our current model through many iterations and a lot of trial and error.
If you are switching from a traditional development model, involve your team when choosing a pattern to start from.
Whichever pattern you choose, you'll probably need to make some changes along the way. Avoid making the mistake of thinking you've found the final solution. Stay open to change. At the same time, don't rush into making a lot of changes at once. Try to improve your model incrementally and give the team time to test out the changes before you decide if they are working and before you make more edits.
Humans don't like change, so it can take your team some time to warm up to a new pattern.

Define roles.
Clearly defining roles in the team will increase its effectiveness and reduce conflict, leading to an overall more productive team.

Take the time to review who holds responsibilities and decisional power. Review your findings with the team and make sure the workload is properly distributed.
Once everyone agrees on the responsibilities distribution, make sure you record your decisions to help people know who has ownership of what.
The image below is my team's page. You can see every person and get an overview of what they work on and their responsibilities.


Lead
- Coordinate project
- Accelerate the speed
- Communicate internally and externally (other balsamiq)
- Core architecting
- Help macOS development

- Windows development
- Core development

- Testing the Editor, Web, Mac, Windows
- Triage RSODs from Cloud
- PRB
- Release Coordinator

- Windows organizer
- Windows development
- Core development
- Triage RSODs and Pivotal stories for Winddows

- Wireframing
- Visual Design
- Bug Reporting

- Lead and organize macOS
- macOS development
- Core development
- Helping whoever needs help
- OPS for macOS build machine and deployment (XCode server, S3, builds)

- PM

- Test Web and Integrations
- Test BWD Linux (Wine)

- Lead Windows development
- OPS for BW WIN

- Lead BW Web
- Core development
- macOS development
- OPS for B4 core and desktop deployment (jenkins, github actions, S3, bwweb)
- Liaison with Online Services for OPS of all BW flavors

- BW Web
- Core development

Figure 29: Development team overview

Keep a loose schedule.
Understand that when you make big changes, there will be a time when estimating the time it takes to implement and deliver a new feature is going to be less accurate.
Avoid setting hard deadlines and give more time to your team to work on the projects. This doesn't mean that the team will always be slower than before. But with a big change such as this one, people will have to put more effort into their workday to follow the new

guidelines; as time goes on and the pattern becomes second nature to them, their speed will improve.
You also must consider that adding UX research and design on top of development will make it look like a feature requires more time compared to traditional software models because each feature now includes design and development, whereas, in traditional software models, the design is already completed when the team starts working on it.
Don’t give up on estimating how long a feature will take to develop. Estimating the time and effort is a difficult skill to train, but it will help with building a solid quarterly and yearly timeline.

There is no magic formula to follow when deciding which development model will suit your team the best, as there are many factors that can influence the success of one approach over another. I would suggest researching the most common Agile development models (some I have cited in Chapter 2.3) and deepening your knowledge of the methodologies to integrate UX design and Agile development adopted by other companies (Chapter 3.7).
Compare those methods to the ones you are using now to see which one you think will suit your team the best. Reflect on who would cover each role and evaluate if you have enough staff with the correct expertise for each position required by the model.

In the end, the success of the transition is all up to the effort every team member is willing to put into it. Integrating Agile and UX design can be done and can be very successful for your team and your company.

6. Conclusions

The goals of the research were to evaluate the known strategies to integrate UX design tasks into Agile development and to investigate if there could be a better way to integrate them.
I also wanted to develop a pattern that alleviated the stress and frustration that can arise when merging two different disciplines, such as UX design and development.

To understand the issues that can arise while attempting to integrate UX tasks into an Agile team and to form an opinion to develop a process to better the work relationship between UX experts and developers, I had to answer a list of questions.

- What are the core differences between traditional software models and Agile patterns?
- Where and how do the UX design tasks fit into a traditional pattern? Where and how can they fit in an Agile pattern?
- What are the main issues encountered by teams trying to focus on UX design in Agile development environments?

Through the comparison between traditional and Agile patterns, one can easily identify the main issues a UX expert would have to face.

The issues are:

- Short development cycle. Having rapid iterations means that there is limited time to carry out UX tasks. Research and design activities can be lengthy and not fit into an Agile iteration.
- Changing requirements. The requirements can change at any point, so the research and design done might become obsolete fairly quickly.
- Involving customers. When iterations are short, it is difficult to involve clients in interviews and research at the correct moment. At the same time, if one manages to involve customers, the amount of feedback received can be overwhelming.
- Documentation: UX tasks require a significant amount of documentation, which can slow down the project.

After I reviewed articles about the topic of integrating UX work in Agile teams, I found out that besides the issues listed above, there are many more friction points but also some solutions to them.

I have collected these issues and divided them into three categories.

Team coherence issues regard all the problems caused by team members not collaborating or not agreeing on topics.
One of the biggest obstacles is power struggles inside the team. Authority disagreements between UX professionals and other teammates happen often. These conflicts can make working in a team nearly impossible.
A second set of issues can arise when coworkers value user-experience differently. The consequence is a team being focused only on development and on adding functionality where all the UX tasks are not prioritized or done on the side.

Scheduling issues are the problems that stand out the most when one wonders how to integrate UX tasks. As I said above, they are caused by the very short development cycle that Agile methods encourage.
These challenges vary from figuring out how to adapt the lengthier UX tasks to fit into one iteration to how to do UX research in a timely fashion.
These issues can be solved by adding a pre-development step before the start of a project to conduct the initial research and create the designs for the first feature. Then, the UX designers and developers work on parallel tracks. The designers work one or more sprints ahead of the development, allowing them more time to do their tasks.
The estimation of the time needed to implement a feature when it includes UX directives can be problematic. Oftentimes, developers overestimate how long such a feature will take when compared to the same feature done without the UX design requests.
While it’s true that a feature that takes UX design into consideration will take longer to implement, the overestimation comes from a lack of knowledge and training of the developers.
Falling back to a development-focused approach in order to deliver the software faster is another risk. This is more likely to happen when overestimating the time required to implement UX guidelines or when an organization is limited by budget or time.

A **lack of knowledge** is the last set of issues. When bringing UX design and development professionals together, they often have different levels of understanding of what each other work entails. This can cause developers and stakeholders to not give UX tasks the right value, and they might end up not prioritizing them correctly as a consequence.
Articles written in academia and by professionals propose a multitude of approaches to try and mitigate these issues.

The most relevant are:

- **One cycle ahead** pattern. After a first sprint dedicated to user research and UX design, the designers will work one (or more) sprint ahead of development. This gives the designers one full sprint to research and design a feature and gather feedback on the last released feature. The developers and designers will work on separate tracks while keeping synchronized via frequent meetings and open communication.

- **Three tracks UX research** pattern. This model focuses on scheduling three different research tracks of different lengths. The strategic track is the longest and is used to schedule generative research, and it will help define the product roadmap. The tactical track lasts several weeks, and it is comprised of generative research to define product features and prioritize the backlog. The Validation track lasts one iteration, and it is used to verify assumptions, validate designs, and gather user feedback.
- **Lean UX** is an iterative pattern that focuses on user feedback and aims to integrate UX design activities into Agile development by involving the developers in the design process. This favors knowledge sharing and prevents silos.

Unfortunately, while these patterns do tackle some of the issues, they do focus on only some of them. They mainly ensure that the UX designers have enough time to do their tasks, but they don't elaborate on the other issues a team is likely to encounter.
I proceeded to illustrate and analyze the development process at Balsamiq, my workplace. We are Agile; we have grown and changed throughout the years, and we have faced many challenges, solving many of them.

Through the years, we've faced issues in all three categories I have identified:

- **Team coherence challenges**. The developers believed that the directives provided by the UX designer were final and complete, while the designer considered them a work in progress and expected to be able to change them while development was happening.
- **Scheduling challenges**. It happened multiple times that the design was done too in advance of development, making it obsolete once development was ready to start implementing it.
- **Knowledge base**. We did not have a staff member trained in doing UX research; all the information we used to create the designs was collected through interaction with the users.

Using the knowledge we acquired through facing these problems, I have created a six-step guide to help teams make a successful transition.
I didn't propose a new pattern as I believe that the ones proposed in the literature work well. Instead, I focused on creating strategies to avoid the other challenges that might arise whilst transitioning. But even if your team has already implemented a UX-centric Agile environment and is facing issues, those ideas can still be applied.

These are my ideas and the order I'd follow them in if I were managing a team that is just about to introduce more focus on UX design tasks:

- **Make sure everyone is on board**. Without a culture of collaboration, you are setting your team up for failure. Find the time to make sure all the doubts have been addressed or that they are at least known to you. Find out the needs of your coworkers.

- **Create a feedback culture**. With change comes frustration. If your team knows how to give and receive meaningful feedback, you can lower the amount of stress they feel and improve the work environment.
- **Align the knowledge base**. You have professionals from different backgrounds trying to work together: stakeholders, developers, and designers. They have different understandings of what each other work entails. If their knowledge is not aligned, at least at a high level, they will have mismatched expectations of each other, grow frustrated, and the level of collaboration between them will be lower.
- **Pick a software development approach** if you don't already have a model that's working for your team. Examine the way you're currently working. Is there enough time to carry out UX tasks? Can you modify your model to allow the UX experts time to do their tasks? If not, I'd suggest researching different models and picking the one that best fits your team structure and needs. Keep reminding everyone that this new process is a "work in progress" and that together and with time, you will adapt it to the needs of the team. This will help your coworkers handle the stress they might be feeling and encourage them to give feedback on what's working and what isn't.
- **Define the roles** of every member of the team. With the pattern change, the roles and responsibilities every employee has might not be clear anymore. Take the time to define the expectations the team has of every member. Otherwise, some work might not be done because no one has ownership of it, or conflicts about who has decisional power might arise.
- **Keep your schedule loose**. Your team is going through a big change. Features will take longer to implement, especially in the beginning, as people need to adjust to the new pattern. Don't add frustration by having hard deadlines and making your team work overtime to catch up. Instead, allow for extra time to carry out tasks, and while everyone is adjusting, actively look for feedback. It's through feedback that you'll see what you should improve on.

By following these steps, the transition to an Agile and UX-focused team should happen successfully. I have put a lot of stress on feedback and collaboration because I have realized that they are what will make a team successful. They should be the focus of every team manager who cares about the success of their team and the well-being of their coworkers.

6.1 Limitations and Future Developments

A limitation of the solution provided is caused by the time constraints of this thesis. With more time, more in-depth research of the published documents could be carried out, and the list of issues encountered by the team could be expanded, both in the number of issues and in the description quality of them.

Afterward, more steps could be added to the solution. The provided steps could also be expanded on by creating further documents with a more in-depth formal process. These documents were not produced because to refine the steps, they need to be applied and tested in one or more work environments.

Regarding this future investigation, if there is the possibility, it would be carried out in multiple different companies, ranging from small to big ones and from young to established teams.
With the results from the studies, the process would be formalized and expanded, turning it into an actual, complete development pattern.

7. Bibliography

[BB88] B. W. Boehm, "A spiral model of software development and enhancement," in *Computer*, vol. 21, no. 5, pp. 61-72, May 1988, doi: 10.1109/2.59.

[BJR01] K. Beck, et al. (2001) The Agile Manifesto. Agile Alliance. http://agilemanifesto.org/

[BK00] K. Beck, “Extreme programming explained: embrace change,” 2000: addison-wesley professional.

[DE10] Deepak Arasu, “Choosing the Right UX Process for Your Software-Development Model,” 2019, https://www.uxmatters.com/mt/archives/2019/06/choosing-the-right-ux-process-for-your-software-development-model.php

[FSM08] D. Fox, J. Sillito and F. Maurer, “Agile Methods and User-Centered Design: How These Two Methodologies are Being Successfully Integrated in Industry,” Agile 2008 Conference, 2008, pp. 63-72, doi: 10.1109/Agile.2008.78.

[JBR99] I. Jacobson, G. Booch, J. Rumbaugh, “The Unified Software Development Process,” 1999: Addison Wesley Professional.

[GS21] Gothelf, Jeff, and Josh Seiden. “*Lean ux.* " O'Reilly Media, Inc., 2021.

[HV16] Atul Handa, Kanupriya Vashisht, “Agile Development Is No Excuse for Shoddy UX Research,” 2016, https://www.uxmatters.com/mt/archives/2016/11/agile-development-is-no-excuse-for-shoddy-ux-research.php

[KMP12] K. Kuusinen, T. Mikkonen, S. Pakarinen, "Agile User Experience Development in a Large Software Organization: Good Expertise but Limited Impac," in Winckler, M., Forbrig, P., Bernhaupt, R. (eds) Human-Centered Software Engineering. HCSE 2012. Lecture Notes in Computer Science, vol 7623. Springer, Berlin, Heidelberg.

[KNF16] P. Kashfi, A. Nilsson, R. Feldt, “Integrating User eXperience Practices into Software Development Processes: Implications of Subjectivity and Emergent Nature of UX,” https://arxiv.org/abs/1605.03783, https://doi.org/10.48550/arXiv.1605.03783, 2016

[KS97] K. Schwaber, “Scrum Development process,” in Business object design and implementation. 1997, Springer. p.117-134

[MI05] L. Miller, “Case study of customer input for a successful product,” Agile Development Conference *(ADC’05)*, 2005, pp. 225-234, doi: 10.1109/ADC.2005.16.

[PF01] S. R. Palmer, M. Felsing. “A practical guide to feature-driven development.” Pearson Education, 2001.

[PW19] N. Pillay and J. Wing, “Agile UX: Integrating good UX development practices in Agile,” 2019 Conference on Information Communications Technology and Society (ICTAS), pp. 1-6, doi: 10.1109/ICTAS.2019.8703607, 2019

[RWW87] W. W. Royce, "Managing the development of large software systems: concepts and techniques." Proceedings of the 9th international conference on Software Engineering. 1987, https://blog.jbrains.ca/assets/articles/royce1970.pdf

[SI08] M. Singh, “U-SCRUM: An Agile Methodology for Promoting Usability,” Agile 2008 Conference, pp. 555-560, doi: 10.1109/Agile.2008.33, 2008

[SS07] J. Sutherland, K. Schwaber (2007). The scrum papers. Nuts, bolts and origins of an Agile process.

[SY07] D. Sy, “Adapting usability investigations for agile user-centered design.” *Journal of usability Studies* 2.3 (2007): 112-132. https://dux.typepad.com/files/sy_agile-ucd.pdf