6 Connecting Agile UX with Agile Software Engineering

Syllabus

Connecting Agile UX with Agile Software Engineering

Contents

6.1 Introduction

- Agile software engineering (SE) approaches, which are now well-known and widely utilized, are incremental, iterative, and test-driven methods of delivering valuable functional software to customers on a regular basis. In the past, agile SE methodologies did not consider user experience. Furthermore, because traditional UX processes do not work well in agile SE contexts, the UX team has struggled to adapt their methodologies to match the SE restrictions.
- At the end of the day, the entire system development team requires an overall strategy that incorporates agile UX while maintaining the fundamentals of agile SE.
- Many people believe that the goal of an agile approach is to be quick, but this misses the point entirely. Being agile means being quick on your feet, nimble, and adaptable to changing circumstances. Yes, an agile process is also thought to be quick. Agile approaches are far faster than the traditional waterfall process. But that is really a side effect, not the main point. It is all about how quick it is.
- Agility is defined as the ability to be quick by not wasting time on things you do not need or will never use. It's all about being quick by recognizing what does not work before investing time and effort on the wrong path. It is all about getting things done quickly by concentrating on the product or system rather than the process. But, above all, it's about being adaptable and receptive to change. A project is not a race; it's more of an obstacle course, or even a labyrinth at times. You never know what will try to derail you from your path until you are confronted with it and must react. You never get the input you need to even know there is a problem to respond to with the old slow and careful waterfall process until the project is practically complete.
- Some agile software developers have nearly turned it into a religion, implementing it without regard for the consequences and pushing it beyond its useful limitations. And it's risky for agile UX practitioners to follow that path. The agile method is merely a tool, and it should not take on a cult-like status. When it works, use it, but do not stop questioning what the optimal procedure is for your particular project.
- We start by going over the basics of the agile SE approach before determining what is required for the two agile processes to work together.
- Monitoring power quality in a system is very important for enhancing the performance of the system. Normally, to monitor the Power Quality (PQ), a monitoring system has to be deployed to collect, assess and infer the collected / measured data.

- In most of the PQ monitoring applications, the deployed system collects voltage and current data. The significance of the PQ monitoring in a system comprises of getting more profit, managing energy saving, carrying proper protective maintenance activities and so on. If PQ monitoring is not properly implemented, then it will lead to industry downtime, frequent system disturbances and economic loss. Therefore monitoring power quality enhances the system voltage and current and thereby the performance of the loads and equipments deployed in the system will deliver efficient operation.

6.2 Basics of Agile SE Method

- Much of what is mentioned here regarding agile methods in SE refers to one of the most authoritative sources on agile SE development methodologies as reflected in the eXtreme Programming paradigm (XP).

6.2.1 Goals and Principles of Agile SE

- To come to an agreement on early agile development methodologies, a group of people got together in a workshop in Snowbird and drafted a "agile manifesto," which included the following principles :
  - Our first aim is to satisfy the customer by delivering valuable software on time and on a consistent basis.
  - Changes in requirements are welcome, especially if they occur late in the development process. Agile procedures take advantage of change to help customers gain a competitive advantage.
  - Deliver working software on a regular basis, anywhere from a few weeks to a few months, with a preference for the shorter timeframe.
  - Throughout the project, business people and developers must collaborate on a daily basis.
  - Build projects around people who are passionate about what they are doing. Give them the space and support they require, and trust them to do the task.
  - Face-to-face communication is the most efficient and effective way of transmitting information to and within a development team.
  - Working software is the most important indicator of progress.
  - Agile processes help to encourage long - term development. Sponsors, developers, and consumers should all be able to keep up with the pace indefinitely.

- A constant focus on technical excellence and smart design improves agility.
- Simplicity - the art of maximizing the amount of work not done - is essential.
- Self-organizing teams produce the finest architectures, requirements, and designs.
- The team reflects on how to become more effective at regular intervals, then tunes and adapts its behaviour accordingly.

- Also from that manifesto, practitioners of agile SE methods value :
  - Individuals and interactions are prioritized over processes and tools.
  - Working software exceeds thorough documentation.
  - Collaboration with customers takes priority over contract negotiations.
  - Responding to a changeover in a planned manner.

6.2.2 Goals and Principles of Agile SE

- Such broad concepts may appear to be cliches but they werenot always so evident. The most severe contrast, a contrast with the old SE "waterfall" lifecycle development process, may be the greatest way to grasp the influence of these principles. The waterfall model Fig. 6.2.1 is a heavy process in that it moves slowly and methodically, completing one process activity before moving on to the next, and so is not adaptable to change. By the end, needs and requirements have shifted, and the product is no longer relevant. Resources are being wasted.

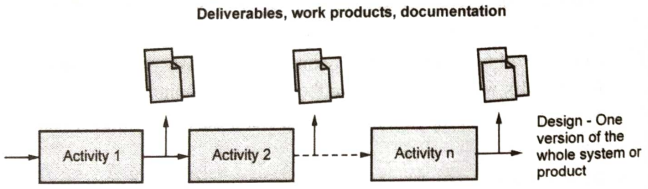

Fig. 6.2.1

6.2.2.1 Operating in Silos

- Each stage of the lifecycle was a complete business in itself, taking place in a more or less isolated environment known as a silo. Each silo had its own management and produced its own deliverables, with a staff of specialists for that stage's work. Each stage usually results in a large amount of documentation.

- You did not obtain a working version of anything until the very end of the lifespan, when you were given a single version of the entire system or product. The difficulty was that it took so long to get there that the design and even the basic concept were frequently out of date by the time you arrived.
- Certainly, any requirements defined and recorded diligently throughout the requirements phase would no longer be comprehensive or correct. And one long iteration would have been insufficient to perfect the design.

6.2.3 Characteristics of Agile SE Methods

- Early coding is a key component of Agile SE development approaches. In comparison to traditional SE, Agile SE features a much shorter, nearly non-existent requirements engineering phase and significantly less documentation. Agile SE code implementation follows the XP model of tiny increments and iterations.
- After each short iteration, or development cycle, the consumer receives a small release. In most situations, these minor releases are meant to be workable versions of the entire system that run on their own and provide some meaningful capacity for the customer, despite their limited functionality.
- Agile SE development methodologies "describe the problem simply in terms of small, distinct pieces, then implement these pieces in consecutive iterations". Each component is checked until it functions properly before being merged into the rest of the system. Then, in what Constantine refers to as "regression testing," the entire system is put to the test until the new feature works with all of the previously produced components. As a result, the next iteration always begins with a working solution.
- Furthermore, agile software development methodologies place a strong focus on communication, particularly ongoing engagement with customers. Formal communication is strongly favoured over informal communication. Close communication is stressed to the extent where they have an onsite client who provides ongoing input as part of the team.

6.2.3.1 Avoiding Big Design Upfront

- The avoidance of big design upfront is a key principle of agile SE methodologies. This means that upfront ethnographic and field research, as well as intensive requirements engineering, are often avoided. The goal is to have code produced as quickly as possible and then react to client feedback to solve problems.

- Pair programming is a technique used by SE practitioners to ensure that they are writing code correctly. Two programmers collaborate on code while sharing a computer and a screen; that is, a colleague is always looking over each programmer's shoulder.
- Pair programming is not new in agile approaches. Outside of agile SE approaches, it was a tried and true strategy with an established track record. Code is also checked against a database of test cases on a frequent and ongoing basis.

6.3 Lifecycle Aspects of Agile SE

- This process would be represented by a set of overlapping microdevelopment activities rather than a waterfall or simply an iteration of phases in a lifecycle diagram. Fig. 6.3.1 depicts XP as an example agile method. In agile approaches, developers do just enough, a microlevel of each task, to provide one single feature.
- This demonstrates that using the waterfall method to construct an e-commerce website would need a list of all requirements that must be met before beginning a top-down design. The same website would be constructed as a series of smaller modules, such as a shopping cart or checkout module, in an agile approach.

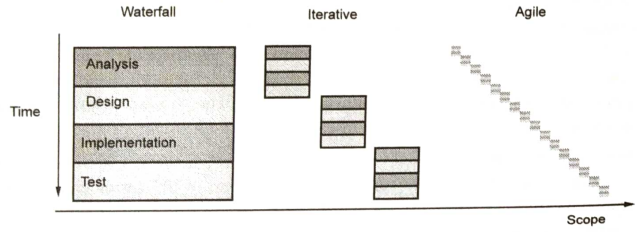

Fig. 6.3.1 Scope of development activities across methodologies

6.3.1 Planning in Agile SE Methods

- We are using XP as a rough guide in our description of how an agile SE method works. Each iteration has two phases, as indicated in Fig. 6.3.2 : planning and a sprint to implement and test the code for one release.

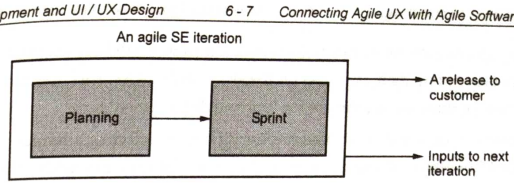

Fig. 6.3.2 Abstraction of an agile SE release iteration

6.3.1.1 Customer Stories

- In late funnel development efforts, customer stories are critical for integrating the UX and SE.
- Fig. 6.3.2 shows how the planning phase of each SE iteration results in a set of customer-written tales that are ranked by business value and implementation cost. A client storey serves a similar purpose to a use case, scenario, or requirement. A customer story is a description of a customer-requested feature written on a narrative card. It is a story about how the system is supposed to address a problem, with each slice of functionality being cohesive and valuable to the client in some way.

6.3.1.2 Story - Based Planning

- Fig. 6.3.3 shows the details of how customer stories are used in SE planning after expanding the "planning" box in Fig. 6.3.2.
- Developers begin the planning phase by meeting with onsite client representatives, as depicted in Fig. 6.3.3. They require customer service professionals to consider the most useful features that can boost business or organization value. The customer tells stories about why these features are necessary. This is the primary method by which developers learn about "requirements," which they learn indirectly from customer service representatives.

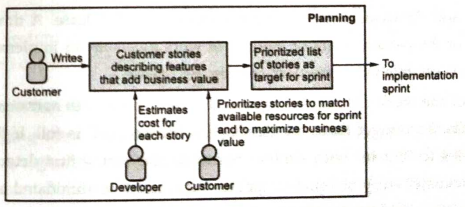

Fig. 6.3.3 Customer stories - basis of agile SE planning

- Developers evaluate the stories and estimate the time it will take to create a solution for each one, noting their estimates on the story cards. In XP, each narrative is given a one, two, or three week estimate for "optimal development time".
- The customer sorts and prioritizes the story cards by selecting a limited number of them that fit within a defined budget and indicate features they wish to include in a "release". Prioritization may result in lists of stories or requirements tagged as "do first," "desired - do if time permits," and "deferred - consider next time". The storylines are broken down into development tasks, which are individually put on a task card. Each of these development tasks is assigned a point value that represents an estimate of the amount of development time required to complete it. The total of all tasks in a story card is then calculated to determine the appropriate development time.
- The planning box produces a set of customer-written stories that are ranked by cost to implement and sent to the upcoming implementation sprint.

6.3.1.3 Managing Customer Stories and Development Tasks

- Atlassian's JIRA software is the most widely used project management system in industry practice for managing stories and associated activities. This system allows product managers and SE roles to plan releases, manage user stories and associated tasks for current and future sprints, user story acceptance criteria, and the current status of each story. JIRA also has a strong defect tracking feature, which allows you to plan which defects will be repaired in a given release, as well as dependencies and other roadblocks to resolving those bugs.

6.3.1.4 Controlling Scope

- Client stories are the local currency in the "planning game," which involves the customer and developers negotiating the scope of each release. A time and effort "budget" of the person-hours or amount of work available for implementing all of the stories is established at the start, usually per release.
- The total of the work estimates is kept as the client prioritizes narrative cards, and when it hits the budget limit, the developers' "dance card" is full. If the customer later decides to "cut in" with another narrative, they must first determine which existing customer story of equal or greater value must be eliminated to make way for the new one. As a result, no one can simply add more functionality.

- This method allows customers influence over which stories are implemented while also providing developers with a tool to combat scope or feature creep. Developer estimations of work can be off the mark, most often underestimating the amount of effort required but it does allow them to draw a line in the sand. Given a specific technological platform and application area, developers become very proficient at this estimation, known as team velocity. The more skilled developers become at estimating, the more smoothly the project will run.

6.3.2 Sprints in Agile SE Methods

- Fig. 6.3.4 depicts an expansion of "sprint" box of Fig. 6.3.2. The activities mentioned in the following areas make up each agile SE sprint.

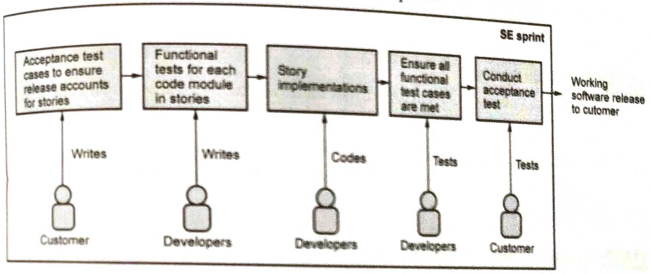

Fig. 6.3.4 An agile SE sprint

6.3.2.1 Acceptance Test Creation

- The functional acceptance tests are written by the customer. There is no formal process for this, so it can be a little hazy, but it does put the client in charge of final code acceptance. Customers get adept at this as they gain experience. The acceptance criteria that the acceptance tests are based on are usually summarized as a list of feature capabilities that the customer must be able to accomplish in the associated customer stories.

6.3.2.2 Unit Code Test Creation

- By assigning client stories to be coded in the following sprint, the team splits the work. A programmer chooses a customer story card and teams up with another programmer. Before beginning any coding, the pair collaborates to develop unit tests to ensure that the functionality of the code to be written is present.

6.3.2.3 Implementation Coding

- The programming teams collaborate to create code for modules that support the functionality of specific client stories. The partners design on the fly as they work. Almost nothing is spoken about design in the agile SE literature. The higher-level architecture is unimportant to the programmers; the system architecture is said to change with each new piece of functionality added to the overall system. This code is integrated into the most recent version by the programming team.

6.3.2.4 Code Testing

- The development team then runs the unit code tests created for the newly implemented modules to ensure that the code implementation of this feature is correct.

6.3.2.5 Regression Testing

- In addition to testing the code for current features, the team repeats all code tests on all previously coded modules until all tests pass, ensuring that the new module does not break any previously built modules. This enables developers to make code changes in response to changing requirements while assuring that all previous code functions properly.

6.3.2.6 Acceptance Testing and Deployment

- This potentially shippable product capability is sent to the customer for approval. The team sends this modest iterative release to the customer once it has been approved.

6.3.3 Sprints in Agile SE Methods

- Fig. 6.3.4 depicts an expansion of "sprint" box of Fig. 6.3.2. The activities mentioned in the following areas make up each agile SE sprint.

6.4 Challenges of Agile SE Methods from the UX Perspective

- Agile approaches make SE practitioners feel productive and in control because the process is driven by them and their clients, not some overarching design. These approaches are low-cost, quick, and lightweight, with quick results. The characteristic of pair programming also appears to result in high-reliability code with fewer defects. Agile SE methods, on the other hand, are programming methods developed by programmers for programmers, and they present certain UX issues.

- There is no upfront study to discern general notions of the system and accompanying work practice because major design is avoided up front. The team's single customer representative isn't necessary to be an actual user and he or she cannot be expected to represent all points of view, needs and requirements, usage concerns, or context. There may not be any real user data available at the outset, and code will be "based solely on assumptions about user demands". There is no methodology for determining user jobs or recognizing user interaction or information demands.
- Aside from these predesign disadvantages, the agile SE technique does not incorporate design, hence there is no room for design ideation in the process.

6.5 What is Needed on the UX Side ?

- Nonetheless, because the UX process lifecycle is inherently iterative, it is an excellent candidate to match with agile software approaches in several areas. However, there is a significant difference. Traditional UX lifecycles begin with a thorough understanding of users and their demands, followed by substantial design activities, both of which are absent in agile SE. In the early stages of agile use research, dedicating an iteration of usage research with real users to discover user demands is recommended. They claim that in one to two weeks, they were able to complete short usage study and early design with five to eight people.
- To work in the UX domain, an agile technique must include some early analysis committed to understanding user work activities and context, as well as gleaning general system concepts inside its ecosystem. Furthermore, some early design activities must be included in the process to provide structure and consistency to how interaction and information are integrated into the design. Also, if top-down design is required for an innovative product design, it should occur prior to any SE engagement. Those requirements are met by early-funnel UX initiatives.
- At the same time, to be compatible with the agile SE side of development, UX methods must :
  - Be lightweight.
  - Put an emphasis on teamwork and require colocation.
  - Include customer and user representatives who are effective.
  - Change the focus of UX lifecycle activities from top-down holistic design to bottom-up feature design to make them compatible with SE sprint-based incremental releases.
  - Include ways to control scope.

6.6 Problems to Anticipate

- A group of UX practitioners met in a special-interest group workshop to share their experiences attempting to implement a user-centered design approach into the agile SE process. These practitioners faced a variety of challenges in their individual surroundings, including :
  - Sprints are far too short; there is not enough time for consumer interaction, design, or evaluation.
  - There were insufficient opportunities for user feedback, and the feedback they did receive was ignored.
  - Customer service representative is ineffective, uncommitted, and lacks colocation.
  - Because the focus is on details, there is no clear vision of greater conceptual design.
  - In a bottom-up strategy, there is a risk of fragmented results.

6.6.1 UX and SE Collaboration

- We offer a general picture of how UX and SE should collaborate in design and implementation in Fig. 6.6.1.
- The UX team is expected to give a UX design as specifications to the SE team to implement and integrate with the relevant functionality, whether in the traditional or agile lifecycle. The problem of merging UX and SE within the project will be well on its way to being solved if everyone on both sides of the project agrees to this straightforward model from the start.

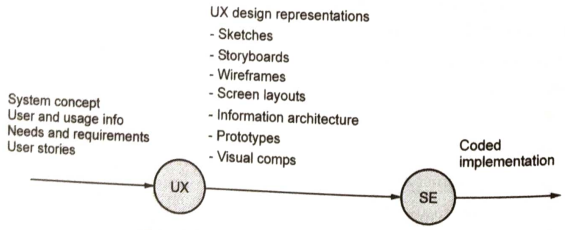

Fig. 6.6.1 How UX and SE work together

- In reality, when the UX team is added to the project, the SE team is frequently well into its process, and the UX team is bound to play catch-up for the rest of the project, and, most importantly, the project will never profit from what the UX team could have done. In the worst-case situation, the UX team is summoned to assess and provide comments on the SE team's efforts thus far.
- While the SE team does require additional inputs beyond those offered by the UX team, this must be done in an open and transparent manner. Both teams receive some of the same types of inputs, but they use them in different ways to create unique concepts. The most important need is that both teams share all inputs acquired from the customer or users. The SE team does not collect requirements or design recommendations without first consulting the UX team.

6.6.2 Need for a Full Overview

- Because the agile approach to SE is iterative and bottom-up, it does not allow for a complete picture of the system to be seen up front. In fact, some agile teams regard the lack of an overview as a "badge" attesting to their proper application of the agile technique, as it allows them to focus on their feature - by - feature evolving picture of the system. The whole point of the agile method is that they can't afford, and don't really need, to develop a system overview up front. This is primarily due to the fact that the user does not see the software directly.
- However, at least on the UX side, developing a system piece by piece might be risky. The user interface represents the user's perspective on the system. This vision, which is entirely dependent on UX design, must appear to be formed around a well-integrated conceptual model, a single unified style or subject, rather than a mess of diverse ideas and viewpoints. Otherwise, the user experience will be unclear and disconnected. It is nearly hard to create a consistent user interface solely using piecewise and bottom-up approaches, one feature at a time. As a result, the UX team requires an overview as quickly as possible.

6.7 Synthesized Approach to Integrating Agile UX and Agile SE

- The majority of the early literature on incorporating UX into the agile SE development process is based on either adjusting UX design methods to keep up with existing agile SE methods or attempting to do only certain parts of UX design processes in the face of an essentially inflexible agile SE method.
- While it is possible for XP and other shortened UX design strategies to live and work together, the two elements are not truly merged in these add-on approaches.

On the UX side, this generates a coping scenario in which UX practitioners strive to live with the limits while also trying to practice their own processes within an overall development environment that is exclusively driven by the agile SE technique.

- We've tried to come up with a way to make our agile UX process work with an agile SE environment without sacrificing important UX requirements.

6.7.1 Integrating UX into Planning

- Fig. 6.7.1 depicts a plan for including the UX role into the Fig. 6.3.2 planning box.
- Small upfront analysis and design by the UX designer, customer, and users are included in the left-most box of Fig. 6.7.1, resulting in a thorough understanding of the work domain, work practice, and conceptual model. The early agile UX funnel contains this upfront UX
- Fig. 6.7.1 Integrating UX role into agile UX funnel planning.

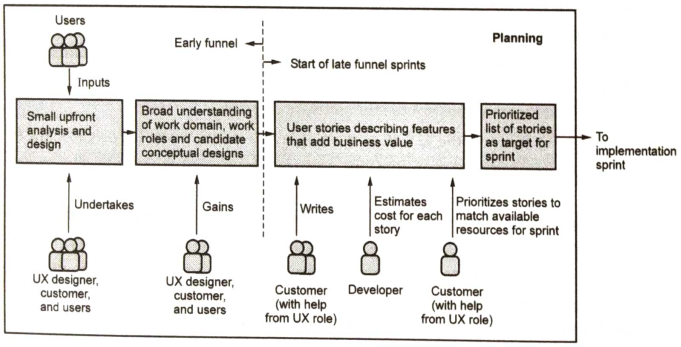

Fig. 6.7.1 Integrating UX role into agile UX funnel planning

6.7.1.1 Small Upfront Analysis and Design

- We discussed how SE practitioners avoid major design upfront in section 6.2.3.1, preferring to build code as quickly as feasible and then address problems by reacting to client feedback later. We cannot afford to operate in this manner in UX. All of our designs, including the small-scope task level designs in the late funnel,

rely on a good general conceptual design in the early funnel to maintain their integrity and consistency. However, the essential groundwork will not allow the entire process to be transformed into a waterfall model. As a result, it must be light on its feet, which is where the preliminary analysis and design come into play.

- The small upfront analysis and design in left-most box of Fig. 6.7.1, is our technique of including an early reduced form of usage research and design that includes the client and users. This is precisely what occurs in the early stages of the agile UX funnel. And most of the "design" in the tiny upfront analysis and design in the early funnel is conceptual design to construct the high-level consistency overview.
- In addition, the UX person aids the customer with additional tasks such as drafting and prioritizing stories while planning. Because the substance of these stories comes from users in the upfront analysis, they are now referred to as user stories rather than customer stories. Although this alters the basic agile model, it gives the UX team a stronger foothold in integrating UX into the broader process.
- The following are the objectives of the small upfront analysis and design in the early agile UX funnel :
  - Recognize the users' work and the context in which it is done.
  - Determine the most important job roles, work activities, and user tasks.
  - Model workflow and activities in the existing enterprise and system
  - Create a high-level conceptual design as a starting point.
  - Identify inputs for selected user stories that reflect the demands of users in their work environment.

Data capture

- Usage research might be as short or as long as desired or possible. At least one or two employees in each important job role should be interviewed and observed at work. There is no transcript or recording of the interviews. To prevent verbosity in the notes, the UX practitioners write notes by hand directly on miniature index cards, which is a Constantine trademark.

Aim toward effective user stories

- We are searching for user stories to help us with our small-scale interaction design and prototype projects. Designers might still begin by telling stories

about their job activities, positions, and duties. Users, on the other hand, are less engaged in tasks and more interested in features. We concentrate on features, which can encompass a large number of connected user work activities and tasks in the context of a job. As a result, for each feature, we normally create a collection of associated user stories.

6.7.1.2 UX Role Helps Customer Write User Stories

- In Fig. 6.7.1, the UX person assists the customer in writing user stories in the third box from the left. As previously said, the user experience person is usually in charge of writing user stories to guarantee completeness, proper scope, consistency, and cohesiveness.
- User story creation will be easier, faster, and more indicative of genuine user needs since UX team members, customers, and users participate in the early agile UX funnel small upfront analysis and design. The user experience job encourages customers to create stories based on workflows discovered during the agile use research phase of small upfront analysis and design.

6.7.1.3 Truth about User Stories

- You can't rely on user-generated content to be accurate, complete, or comprehensive. As a result, in real projects, UX analysts and designers take user feedback gathered from usage research and compose small-scope user capability requirements as user stories, as if the users had expressed them, resulting in a unified format for the entire collection. This is one of the most common ways in which process differences can be seen in practice.

6.7.1.4 UX Role Helps Customer Prioritize User Stories

- The UX person can keep an eye on the overarching vision of user experience and an unified conceptual design by assisting the customer representative in prioritizing the user stories, guiding the result toward an effective set of tales for an iteration.

6.7.2 Integrating UX into Sprints

- Fig. 6.7.2 depicts UX equivalent activities occurring in the late funnel during and agile SE sprint of Fig. 6.3.4.

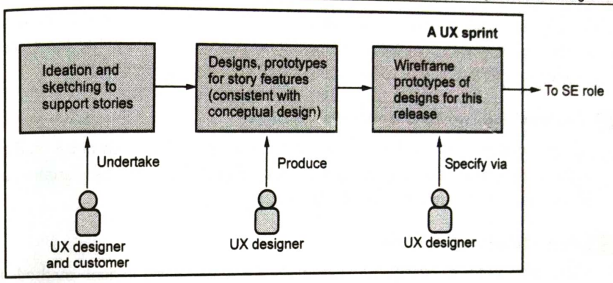

Fig. 6.7.2 UX counterpart of agile SE sprint

- While the SE people are doing a sprint, the UX person and the customer are sprinting in their own way, as depicted in Fig. 6.7.2. They start by choosing a narrative or a feature that has a group of connected user tales.
- The UX designers begin ideation and sketching of an interaction design to support the functionality of the user story with the conceptual design in mind. Static wireframe prototypes are used as rough sketches of interaction designs, progressing to "interactive" click-through wireframe prototypes that show navigation inside interaction sequences.
- Design reviews, design walk - throughs, plus iterative design fixes :
  - The tightly-coupled design creation-prototyping-evaluation cycle will generally begin with frequent numerous static wireframe sketches at this modest scale. Within the UX team, these sketches are repeatedly examined for ideation in task-level design production. Short focused returns to usage research to fill gaps and answer queries about the product might also be part of this early design brainstorming and sketching.
  - When the UX team is convinced that they have their best first design, they show it to all stakeholders, including clients and users, in the form of a click-through wireframe prototype during a walkthrough review. Multiple rounds of feedback, refinement, and additional review are common outcomes.
  - Then, with SE developers, a similar walk-through examination is conducted to obtain comments on feasibility, consistency, platform concerns and so on. This may result in several rounds of repairing and refining before being handed off to developers for implementation. UX designers and software engineers should both be on the same page about the functionality by this point.

- After implementation, the UX designer should compare the results to the UX designs to ensure that the implementation is accurate. The entire team should then submit the completed feature to the customer for approval.

6.7.3 Synchronizing Two Agile Workflows

- We now discuss how the UX and SE teams collaborate and synchronize workflow in their respective areas of the agile process, after describing agile SE planning, agile SE sprints, and UX integration in the late funnel.

6.7.3.1 Dove - Tailed Work Activities

- Miller presented a "staggered" parallel track agile development method that includes a "criss-cross" interplay between UX and SE operations spanning several agile development cycles. "Work ahead, follow behind" is how the general strategy is described. On agile teams, UX people become masters of development time travel, flitting back and forth between past, present, and future development work with ease. We show a strategy in Fig. 6.7.3 for how UX and SE people might coordinate their work via a dovetail interchange of activities across progressive iterations, based on Miller's approach and integrating our early funnel concept.

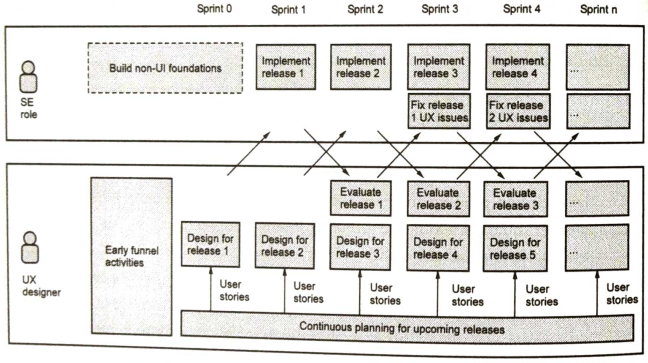

Fig. 6.7.3 Alternating UX and SE workflow in agile process

- SE people started with Sprint 1 in the original agile SE approach, taking a set of stories and constructing a release. When the only thing going on was implementation, that worked. We need to make a few modifications now that we have included UX design and assessment to the mix, starting with the early funnel activities before moving on to Sprint 0 on the UX side.

- Sprint 0 is for starting the synchronization with the SE cadence after early funnel work on usage research and conceptual design on the UX side. The length of time spent on these early funnel activities is determined by the size and complexity of the product or system, as well as the UX team's experience with the problem area. The UX team should have UX designs for the initial feature ready to give to the SE people for Sprint 1 by the end of Sprint 0. During this time, the SE team concentrates on other tasks, such as developing the software infrastructure and services that will support the entire system.
- When the UX team completes their designs for Release 1, they turn them over to the SE team for execution in Sprint 1, which covers both functional and interface design components for that cycle.
- Not all UX design difficulties are created equal; in certain cases, there is not enough time in a sprint to fully address UX design. The design will then have to grow throughout the course of multiple sprints' design and evaluation activities. Also, with interaction design, we occasionally want to try out two alternatives because we are not sure, thus this will have to be spread out over several sprints.
- People on each segment of the team are usually working on various things at the same time due to the delayed or overlapped activities. For example, immediately after passing off the designs for Release 1 to the SE team, the UX team begins work on designs for Release 2 and continues until Sprint 1 is completed. In each given sprint, say Sprint n, the UX team is conducting user research and planning for Sprint n + 2, designing for Sprint n + 1, and evaluating the design for Sprint n - 1.
- Following the "lifecycle" of a single release, Release n, we observe that the UX role designs for Release n in Sprint n - 1, which will be implemented by the SE employees in Sprint n. In Sprint n + 1, UX assesses Release n. In Sprint n + 2, SE patches it and rereleases it at the end of that sprint.

6.7.3.2 Value of Early Delivery

- You have the ability to give design visions very early in the process, resulting in incredibly early customer participation.
- Feedback for the first feature can go well beyond just that one feature's use. This is the team's first chance to receive genuine input. There are a lot of other things that can come out of it that are not related to that feature. You will, for example, receive feedback on how the procedure is progressing. You will receive feedback on your design's overall style. You will learn about other concerns and queries that clients have that you would not know about until much later in a nonagile approach.

6.7.3.3 Continuous Delivery

- Customers and users are delivered in bits, not in a continuous stream. The customer sees a UX prototype of the forthcoming release at the conclusion of each sprint, and the full functional implementation of that prototype in the next sprint, Sprint n + 1. They see the UX evaluation findings for the same prototype in Sprint n + 2, and the final redesign in Sprint n + 3. At each of these periods in time, the consumer can provide input on the interaction design.

6.7.3.4 Importance of Regression Testing

- Regression testing is a stage in the agile SE process that connects the most recently tested - and - passed feature to the ones that came before it. The agile method is nothing more than an incremental waterfall method without regression testing. You cannot iterate major concepts since you are stuck with judgments made in Sprints 0 and 1. Unfortunately, the time constraints of sprints in real life might lead to teams skipping regression testing.
- Furthermore, on the SE side, this type of fast testing is simple and, for the most part, automated. It is a whole different scenario on the UX side. On the UX side, regression testing would mean conducting an evaluation after each sprint and using all prior measuring instruments, as well as those for the current sprint, before releasing it to the customer. There is usually never enough time to achieve this in practice.

6.7.3.5 Planning across Iterations

- Fig. 6.7.3 depicts planning as a single box at the bottom of the page that extends across all sprint cycles. That is, planning does not take place in discrete small boxes over time at a precise point in the flow. Planning is more of a "umbrella" activity that is spread out over time and is cumulative in that it creates a "knowledge base" based on agile user research. Each cycle's planning does not require a restart of the planning process.
- Instead, the same knowledge base is used to consult, update, and massage the original small upfront analysis and design results, as well as anything contributed to complement those results. This type of UX planning adds some top-down benefits to an otherwise solely bottom-up process because an overview and conceptual design are emerging during the process.

6.7.3.6 Communication during Synchronization

- This type of intertwined development process runs the danger of collapsing if something goes wrong. This increases the need for continual communication so that everyone is aware of what others are doing, what progress they are making, and what obstacles they are encountering.
- Agile methods are more prone to failure than their more powerful alternatives. Because each aspect of a closely coordinated overall operation is dependent on the others, if something goes wrong in one location, there may be limited time to react to unexpected events. This risk can be reduced by maintaining constant contact.

6.8 Review Questions

1. *Explain basics of agile SE methods.* **(Refer section 6.2)**
2. *Explain lifecycle aspects of agile SE.* **(Refer section 6.3)**
3. *Explain characteristics of agile SE methods.* **(Refer section 6.2.3)**
4. *Explain challenges of agile SE methods from the UX perspective.* **(Refer section 6.4)**
5. *Explain sprints in agile SE method.* **(Refer section 6.3.2)**
6. *What are different problems to anticipate.* **(Refer section 6.6)**
7. *Explain synthesized approach to integrating agile UX and agile SE.* **(Refer section 6.7)**
8. *Write a note on : Integrating UX into sprints.* **(Refer section 6.7.2)**
9. *Explain synchronizing two agile workflows.* **(Refer section 6.7.3)**