Getting Started with Agile Project Management Methods for Elearning

Megan Torrance
TorranceLearning
Training2013
Session 108 | February 18, 2013 | 8am

Megan Torrance has 20 years of experience in the learning industry, covering everything from instructional design, elearning development, instructor-led training, technical writing, LMS deployment & system administration, to change management and project management. Megan is the Chief Energy Officer (CEO) of TorranceLearning, an elearning design and development firm near Ann Arbor, MI. She graduated from Cornell University with a BS in Communication and an MBA.

Get in touch with Megan:
mtorrance@torrancelearning.com | m: 734-368-5086
www.torrancelearning.com
twitter.com/mmtorrance | linkedin.com/in/megantorrance
Selected Notes & References

How do you keep your project from falling short in delivering client (internal or external) needs on time, and within budget? How do you adjust when client needs are often changing? The software development industry is embracing Agile development methods to address these issues and there is much that the elearning development world can learn from them. Agile provides a framework for authorizing adapting to change as it happens, and working with the client to deliver the content most-needed by learners.

Participants in this session will learn about the Agile Extreme Programming (XP) development methods we’ve adapted and how our project teams actually use them on the job. We’ll talk about the tools & supplies you need, how to choose a project to pilot, and how we train our team. Learn how Agile techniques improve centralizing information, keeping a project on track, tracking billable hours more easily, and defining clear and measurable job tasks. We’ll share experiences from our first pilot project to show how we communicated with the client about the method and how things worked out.

Agile is one of the “next big things” to hit the elearning world. There’s a lot of talk out there about what it is, and how it’s useful (or not) … but not so much about how to get started. This presentation will focus on the actual how-to. Learn about a tool that can streamline your process, figure out if Agile is right for you, and get answers to your questions.

S.Q.E.R.T

Agile – like any other project management method – provides a framework for balance. Scope Quality Effort Resources Time – you bend any one of them and the others get pushed out of whack proportionately. The path of least tension, then, is to balance them all.

Agile lets you adjust your course as you go ... which makes a heck of a lot of sense.
In February of 2001, a group of seventeen software development professionals gathered in Utah to discuss project development methods. Each brought unique perspectives and experiences, and collaborated in an effort to find a more efficient way to develop software. By the end of the weekend, the group—who began to refer to themselves as “The Agile Alliance”—managed to agree on four main values. These values became the Manifesto for Agile Software Development, which was published and used to define the approach now known as Agile Software Development.

“The Agile Alliance” also realized that, while the Manifesto provided some specific ideas, there was a deeper theme that drove many members of the group. They were brought together by a set of compatible values: values based on trust and respect for each other; values that promoted organizational models based on people and collaboration; values that built the type of organizational communities in which people would want to work. They concluded that Agile Methodologists focused on delivering good products to customers by operating in an environment that made people and culture a priority.

--
More:
http://agilemanifesto.org/history.html
http://en.wikipedia.org/wiki/Agile_software_development#Agile_Manifesto
http://www.agilealliance.org/the-alliance/the-agile-manifesto/
Agile Principles

“The Agile Alliance” also established 12 additional principles to support the four core values mentioned in the *Agile Manifesto*. Each is used to further describe what it means to be Agile.
Getting Started with Agile Project Management Methods for Elearning

Megan Torrance | Training2013 | Session 108 | February 18, 2013 | 8am

Agile processes promote sustainable development. The Sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximizing the amount of work not done—is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Agile Variations

Scrum

Scrum is an iterative subset of the agile methodology. Projects are broken up into sprints, which are the basic units of development in Scrum. Sprints are decided upon in a planning meeting between stakeholders and development team members, and can last between one week and one month. After each sprint is completed, the meeting members reconvene to assess the progress of a project and plan its next steps. This allows a project’s direction to be adjusted or reoriented based on completed work, rather than on guesses or predictions.

The scrum methodology also relies on a set of specific roles: the Project Owner, the Scrum Master, and the Team Members. The Product Owner has the most responsibility. They represent the customer’s interests and are in charge of communicating the vision of the product to the development team. The Scrum Master is the go-between for the Product Owner and the rest of the team. They ensure that things are going according to plan and that there aren’t any obstacles preventing the team from completing its sprint. The Team Members are responsible for completing work. They decide how to get the work done and then they execute their plans. A typical team consists of approximately seven cross-functional members. A software project team, for example, would be made up of software engineers, architects, programmers, analysts, QA experts, testers, and UI designers.

More:
http://scrummethodology.com/

Kanban

Kanban is a team management method that encourages small continuous, incremental and evolutionary workflow changes that stick over time. Its core concepts are: make work visible, limit work in progress, and help work to flow. Work is made visible through the use of visual-task boards, which are also called Kanban boards. These are used to show a team how a project is progressing: it reveals activities, limits, and blocks. It also allows them to see the cause and effect of changes to the process, as well as whether or not flow is happening.

A key element of Kanban is that it places a limit on how much work in progress (WIP) can exist in any stage at any one time. This means team members cannot pull a new item from the work queue until they finish an in-progress item and a space on the board opens up. The focus, then, becomes on finishing and multi-tasking or receiving more work than the developers can handle.
becomes minimal. In addition, limits prevent overproduction, reveal bottlenecks before they get out of hand, and maximize throughput.

The WIP limits tie into the third core concept, which is help work to flow. Kanban utilizes the lean concept of flow to continuously and predictably deliver value. Kanban practices allow teams to look at the entire production process, rather than a specific development phase; they spot workflow issues as soon as they occur, allowing for bottlenecks to be confronted and worked out, and they reduce waste, smoothing out costly delays in the process. WIP limits also improve flow in that they bring focus and completion at a steady rate, allowing teams to swarm on limited items and thus increasing the amount of teamwork.

More:
http://www.kanban101.com/
http://richhewlett.com/2012/02/13/an-introduction-to-kanban/
http://niksilver.com/2011/02/16/what-is-a-kanban-limit/

Extreme Programming (XP)

Extreme Programming, otherwise known as XP, focuses on customer satisfaction and teamwork. It values of simplicity, communication, feedback, and courage, and the goal is to produce outstanding products at lower cost, with fewer defects, high productivity, and a high return on investment. XP is also based on 12 key practices: The Planning Process, Frequent Small Releases, System Metaphor, Simple Design, Test Driven Development, Refactoring, Pair Programming, Collective Code Ownership, Continuous Integration, Sustainable Pace, On-site Customer, and Coding Standard. However, the basic life cycle of an XP project is broken down into four stages: Planning, Coding, Design, and Testing.

The Planning Process addresses two key questions in software development: predicting what will be accomplished by the due date, and determining what to do next. The two key steps in this process include Release Planning and Iteration Planning. Release planning is where the Customer presents the desired features to the programmers, and the programmers make a high-level time estimate. This allows the Customer is to lay a plan for the project through feature and cost prioritization. Iteration planning is the direction the team is given direction every couple of weeks. Iterations are established according to what the customer wants done by a certain point. Then, programmers break those needs down into tasks, and estimate their cost (with a higher amount of certainty than the estimate created during Release Planning). Both steps in the planning process keep progress entirely visible, allowing XP projects to deliver more of what is needed, with less pressure and stress.
In an XP project, coding is done in pairs. The programmers sit side-by-side at the same machine, which ensures that all production code is reviewed by at least one other programmer. This results in better design, better testing, and better code. Additionally, pairing serves to communicate knowledge throughout the team. As pairs switch, team members benefit from exposure to everyone’s specialized knowledge. Programmers learn, their skills improve, and they become more valuable to the team and to the company.

XP practitioners also place emphasis on simple design. Programmers only design what’s needed for the current functionality of the system. However, this also means that design occurs throughout the project, as it gets updated with each iteration.

XP requires feedback, and good feedback requires good testing. As such, many XP teams practice “test-driven development,” or working in very short cycles of adding a test, then making it work. Testing allows programmers to receive immediate feedback, and provide invaluable support as the product design is improved.

More:
http://www.extremeprogramming.org/

Lean

The core idea of Lean is to maximize customer value while minimizing waste. The ultimate goal is to provide perfect value to the customer through a perfect value creation process that has zero waste. To accomplish this, lean thinking changes the focus of management from optimizing separate technologies, assets, and vertical departments to optimizing the flow of products and services through entire value streams that flow horizontally across technologies, assets, and departments to customers. Eliminating waste along entire value streams, instead of at isolated points, creates processes that need less human effort, less space, less capital, and less time to make products and services at far less costs and with much fewer defects, compared with traditional business systems. Companies are able to respond to changing customer desires with high variety, high quality, low cost, and with very fast throughput times. Information management also becomes much simpler and more accurate.

Lean is more strategically focused than other Agile methodology. The goals are to develop software in one-third the time, with one-third the budget, and with one-third the defect rate. Lean is not a management or development methodology per se, but it offers principles that are applicable in any environment to improve development. It’s based on several principles:
Eliminate waste, Amplify learning, Decide as late as possible, Deliver as fast as possible, Empower the team, Build integrity in, and See the whole.

- **Eliminate waste**: Waste is defined as anything that does not create value—for example, an activity that could be bypassed or the result could be achieved without it is waste. This can include unnecessary functionality, delay in the development process, slow internal communication, bureaucracy, insufficient testing, and unclear requirements. Lean uses a value stream mapping technique to distinguish and recognize waste.

- **Amplify learning**: Enables programmers and/or developers to attain the knowledge they need to develop a system that delivers business value.

- **Decide as late as possible**: Delaying decisions as long as possible until they can be made based on facts and not on uncertain assumptions and predictions.

- **Deliver as fast as possible**: The sooner the end product is delivered without considerable defect, the sooner feedback can be received, and incorporated into the next iteration.

- **Empower the team**: The team is the most important element in successfully delivering a product. In order to give people motivation and responsibility, they need to be authorized to make it happen.

- **Build integrity in**: There are two kinds of integrity—perceived and conceptual. Perceived integrity is the customer’s experience with the product. Conceptual integrity is how well the architecture and system components flow together to bring about the perceived integrity.

- **See the whole**: Lean thinking has to be understood well by all members of a project, before it can be implemented. It’s only when all of the principles are combined that there’s a basis for success in product development.

---

More:

- [http://www.lean.org/whatslean/](http://www.lean.org/whatslean/)
- [http://en.wikipedia.org/wiki/Lean_software_development](http://en.wikipedia.org/wiki/Lean_software_development)
Agile Techniques

Story Cards

Story cards, or user stories, help the team define and authorize scope within each iteration. They’re high-level and written from the perspective of the customer in order to capture what the customer needs and why, rather than the technical details of how. Good user stories should be six things: Independent, Negotiable, Valuable, Estimable, Small, and Testable (INVEST).

A user story should be independent in that it should have no dependency on or overlap with any other story card. Theoretically, this means that story cards should be able to be completed in any order as well.

A user story should be negotiable in that it can be rewritten or changed up until its production. A story card should capture the essence of the customer’s need without actually going into any technical details. Those should be discussed after the story card has been written.

Every user story should be of value to the customer and/or end user.

A user story needs to be estimable so that developers can provide a time estimate. The estimate allows for prioritization and continued project planning.

A user story needs to be small or sized appropriately so that it’s easy to plan, prioritize, and task. It also leads to more accurate estimates and better predictability.

A user story should be testable in that there’s a way to determine that the customer’s requirements can be or have been met. If a card isn’t testable, it should not be developed as there is no way to tell when the task is done.

--

Sources:
Planning Levels / The Big Planning Board

An Agile project plan can have several levels of detail, depending on the scope of the project. At the high level is a holistic view of product requirements as well as a time frame for when the requirements will be completed. In traditional Agile, there are typically two levels of high-level planning: Product Roadmaps and Release Plans.

The product roadmap is a high-level view of the product requirements in the form of a big visible chart. It lists the key features and characteristics a product will need to deliver in order to achieve the customer’s vision. As such, it typically spans a number of releases of the product. It is also a time-based view of the anticipated delivery lifecycle of the product. The product owner and project manager are often tasked with checking and updating the board as changes are made and items are added, moved, and removed from the roadmap.

The release plan shows the features that are to be delivered in the current release of the product. Release planning starts with the product owner explaining the goals for the release. The team discusses what is needed to deliver against these goals and expresses any other factors that need to be taken into account when delivering the release. The plan contains the currently agreed prioritized list of features at story level. It’s based around the team’s anticipated rate of progress (velocity) and conveys a shared understanding about what needs to be included in the current release.

At TorranceLearning we use code names for each of our projects to protect the anonymity of our clients and, well, let’s face it, it’s fun! This way, anyone can walk through our front door and see our project boards and they won’t have a clue whether project Cher is for one of the Fortune 100 company clients or the coffee shop franchise that started just down the street. Sometimes our clients join the fun and participate with choosing their project’s codename, other times we get the creative juices flowing ourselves.

More:
http://www.dummies.com/how-to/content/the-agile-roadmap-to-value.html
http://www.infoq.com/articles/many-levels-planning-agile-project
Weekly Planning Boards

Weekly planning boards are used within each iteration. After a customer and development team identify the stories to be worked on, the team breaks the work down into specific tasks for each team member. Task allocation is ideally done on a “pull” basis whereby team members identify the work they are able to do and select their own tasks. Tasks should be very small – from a few hours to a day or so, and are discrete measurable activities. The project manager confirms that all the tasks are allocated and makes sure the work committed is realistically deliverable, and then the team members sequence and estimate them. Estimation is now at the level of hours of work needed to undertake a single task. These tasks are then written up on task cards and the task cards tracked on the weekly planning board.

At TorranceLearning, our weekly boards have “swim-lanes” for each person who is assigned to a project. A headshot at the top indicates who is responsible for the cards that are laid out in the vertical column below. Each project’s weekly board has horizontal rows for each day of the week (Monday, Tuesday...etc.) as well as a row at the very bottom, labeled pull-ahead. Any cards placed in the pull-ahead row are cards that can be brought into play if someone completes all of their assigned task cards for that week.

--

More: http://www.infoq.com/articles/many-levels-planning-agile-project
Estimating

In Agile, estimates are made to determine a project’s development schedule as well as costs. There are two levels of estimation: estimation done at the user story level, and estimation done at the task level. Estimations done at story level are usually in increments of days, and estimations at task level are broken up into hours. In classic agile, everyone on the team makes an estimate, then the team calculates the average and reports it back to the client. However, when it comes time to finally work on the task, the estimate that wins out is the one that was made by the person who’s completing the task.

At TorranceLearning, we estimate cards in two different ways. Whenever possible the person who will be doing the work estimates the amount of time needed to complete the work—bottom-up estimating. However, there are times when the person who will be doing the task needs someone else to provide an estimate for them—top-down estimating. This happens most often in one of two scenarios. First, when the person isn’t experienced in the task; it’s something they haven’t done before so they’re unsure of how long it would take. Second, when a task is nebulous; we need to place boundaries on it for the sake of project management and budgeting (e.g. research best practices for implementing corporate mentoring program—this could take two weeks or two hours depending on the depth of research needed).

A few of our estimating tips:

- Break estimates down to the smallest time increments
- The more estimates you do, the better you’ll get at them
- No one is yelled at for a bad estimate
- We can discuss it and disagree but if it takes you 4 hours, it takes 4 hours. This is one of the hardest things to communicate with clients because they assume that an estimate is either our set price or our maximum price. An estimate is an estimate.

--

More: