TechExcel

A three-part DevSuite whitepaper Series

AGILE Made Easy

Learn how to implement agile techniques in your team and build better software.

BEST PRACTICES

What is Agile Development?

Whether you're developing web applications, mobile applications, enterprise multi-platform applications or products and embedded systems, agile seems to be everywhere.

But what is agile development? According to the Agile Manifesto *, "it's a methodology based on iterative and incremental development, in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams."

It helps reduces development cost and risk by delivering higher quality software and products more rapidly. Quality and customer satisfaction is increased by effectively meeting customer needs and expectations through constant customer interaction.

Below we will cover some of the best practices we have learned throughout the years while implementing and utilizing Agile.

User Stories

By definition, user stories are short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template:

As a <type of user>, I want <some goal> so that <some reason>

The user story is typically written into some sort of index card, whether it's paper or electronic and arranged in a way to facilitate discussion. The purpose of the simple user story is to shift the focus from writing about the feature to discussing them. Open and constant communication is the key to making Agile work.

Sometimes a user story may be too big to fit into the sprint or there is not enough detail to it. In this case, the best practice is to split a user story into multiple smaller user stories or add some condition of satisfaction to it.

Consider the following example as an agile user story:

As a director of marketing, I want to be able to report on the effectiveness of our lead generation sources so that I can identify which sources to keep.

We can further break this down into additional user stories:

- Make sure the system can track all sources.
- Support the ability to run a trend report on all sources.
- Make sure the sources can easily be added by the end user.

We want to make sure user stories are short enough to fit into a sprint so that we deliver working software frequently as what's stated in the Agile Manifesto.

User stories are also assigned an arbitrary story point. Story points measure the value and complexity of the story.

For example,

- How important is this feature to the customer?
- How much revenue can it bring in for the company?

It is a high-level estimate used for prioritization and filling up the iteration and should not be confused with number of hours it takes to implement the story; that's where breaking a story down to tasks come into play.

Product and Release Backlogs

The product backlog is a prioritized features list, containing user stories that define all functionality desired in the product. It is not necessary to start a project with the lengthy, upfront effort of documenting all requirements. The product backlog is almost always more than enough for a sprint and can grow and change as more is learned about the product and its customers.

A typical backlog comprises of features and bugs. Because there is really no difference between a bug and a new feature – each describes something different that a user wants, and each requires some work to be done – bugs are also put into the product backlog.

A release backlog can be very useful in defining a roadmap of the product. It is a subset of the product backlog that is planned to be delivered in the coming release, typically, a 3-6 month period. During release planning (not to be confused with sprint planning), we break down the product backlog and move stories into the release backlog. This then, represents a list of what we think will be committed in the upcoming release. Of course it is subject to change as priorities shift, but it's a good indication of what's to come. The release backlog is especially useful for fixed-scope contracts.

Effective Planning

There are two types of planning: release planning and sprint planning. As a best practice, we recommend a release planning meeting every quarter. This helps identify what's coming in the next release and make changes to the roadmap as product development progresses. The attendees for this meeting are generally the product owner, scrum master and stakeholders.

During release planning, the release backlog is populated from the product backlog. High-level story point estimates are used to determine which stories should be moved into which release.

The goal of sprint planning is to enable the product owner the ability to describe the highest priority features to the team. The team asks enough questions so that they can turn a high-level user story from the release backlog into more detailed tasks for the sprint. Again, the story point estimates are used to prioritize these features in the backlog first. For those features we want to commit into a sprint, we break them down into tasks. This includes such tasks as implementation, documentation, unit testing, and etc. Any effort needed to complete a story should be accounted for as tasks.

For each task, hours are associated with the tasks. So while we use story points as a measure to populate the sprint, the task hours can provide more accuracy. We use both metrics to determine what and how much can go into our two week sprint. Although story points and task hours are not the same, it's not unusual to see story points correlate more and more with hours over time.

During sprint planning, there may be situations where a user story is high in value and high in complexity, as well. There may not be sufficient information to properly estimate the tasks that need to be done. In this case, we use a concept called spiking. Its sole purpose is to provide the answer or solution to the unknowns. Like any other task, the spike is given an estimate and included in the sprint. It does not affect other deliverables because it's properly prioritized and planned into the sprint.

It's important to plan correctly so that all assigned stories can be finished within a sprint. In the case where there are remaining items, don't fret. Unfinished stories and tasks are placed back in the backlog and the product owner reprioritizes and reassesses ROI and effort. Unfinished items typically end up back at the top, but not always. As sprint planning and your Agile process matures, you will see better estimates resulting in less and less unfinished stories and tasks.

Agile Roles

Agile promotes and relies on empowering developers, open communication and continuous improvement.

Agile methods empower the team by letting them work interactively.

It is not necessary for a project manager to have authority and responsibility over all decisions and plans. In fact, a single person acting as boss inhibits the team's full potential. A team that is calm and confident about their work is very decisive. Any team member who knows enough to foresee a problem also knows enough to lead the solution.

The role of developers is different than just writing code as directed. Their responsibilities should include deciding how to get organized and what to do next. Trust that the team can accomplish the tasks, after all, the end result of a sprint IS working software.

As a customer on an Agile project, you are actively involved in most discussions to make sure your vision is followed. The most challenging part will be setting priorities and making sure what will get done and what will not be finished.

As a manager/scrum master, watch how the team interacts and be ready to resolve broken relationships and obstacles. You don't get to decide what an obstacle is and what must be accepted as non-negotiable. The team will tell you every day at the daily stand-up meeting and you must act to the best of your abilities.

Open Communication

Daily standup meetings are a great vehicle to open up communications, particularly when enhanced by an Agile story board. Obstacles are identified and progress is tracked in real-time. There is no better way to communicate than person-to-person, face-to-face. All team members are required to attend the meetings, including the product owner and scrum master. Anyone else is welcome to attend, but they are only there to listen.

Our daily stand up meetings are usually 10 minutes long because we just ask each team member three simple questions:

- What did you do yesterday?
- What will you do today?
- Is there is anything you need?

The scrum master will try to resolve issues as quickly as possible, but if he can't, he will make sure to find someone who can. Keep in mind that the meeting is not used for problem-solving or issue resolution. Issues are taken offline and dealt with by the relevant subgroup, usually immediately after the meeting

Metrics

Since we use task hours when breaking a user story down to tasks, this time automatically becomes the time remaining for a task during implementation. We ask our team to always put in the proper time remaining and time spent for each of their tasks. From this data, we generate burn down reports to help us understand how a sprint is progressing.

The burndown chart compares the remaining hours the team needs to complete against the ideal burndown to ensure at the end of the sprint the time remaining is 0. It is important to note that if there are any open tasks left at the end of a sprint, the story is always moved back into the backlog for reprioritization. Nothing is ever added while a sprint is in progress.

Whether you're starting Agile or have already been practicing it, these best practices will help you deliver what your customers need, on time and frequently. Furthermore, it's important to learn from your releases to improve both your software and the process in place.

There is no "right" way to do Agile, but if your Agile process can achieve the above outcome, then your team is doing the right thing.

* The Agile Manifesto

In February 2001, 17 software developers (see below) met at the Snowbird, Utah resort, to discuss lightweight development methods. They published the Manifesto for Agile Software Development[1] to define the approach now known as agile
software development. Some of the manifesto's authors formed the Agile Alliance, a non-profit organization that promotes software development according to the manifesto's values and principles.

Corporate Headquarters
3675 Mt. Diablo Blvd., Suite 200 Lafayette, CA 94549
Tel: (925) 871-3900
Fax: (925) 871-3991
www.techexcel.com

TechExcel EMEA
Crown House, 72 Hammersmith Road London W14 8TH, UK
Tel: +44 (0)20 7470 5650
Fax: +44 (0)20 7470 5651
Email: emeainfo@techexcel.com