TECH-ON-TAP E-JOURNAL

ISSUE-6

MARCH – 2017

DEPARTMENT OF MCA

1. Scrum Introduction to scrum terms

2. Life Cycle of Scrum Methodology

3. Important Facts about Agile Testing

4. Good Reasons to do Agile Development

5. When to Use Agile, and When Not to

Agile Scrum Methodology

Scrum is an agile process most commonly used for product development, especially software development. Scrum is a project management framework that is applicable to any project with aggressive deadlines, complex requirements and a degree of uniqueness. This E-journal article explore all the recent trends in agile.

DEPARTMENT OF MCA
II Year MCA Students

(This article is extracted work of the assignment prepared by II MCA students)

Scrum Overview - Introduction to Scrum Terms

An introduction to Scrum would not be complete without knowing the Scrum terms you'll be using. This section in the Scrum overview will discuss common concepts in Scrum.

Scrum team: A typical scrum team has between five and nine people, but Scrum projects can easily scale into the hundreds. However, Scrum can easily be used by one-person teams and often is. This team does not include any of the traditional software engineering roles such as programmer, designer, tester or architect. Everyone on the project works together to complete the set of work they have collectively committed to complete within a sprint. Scrum teams develop a deep form of camaraderie and a feeling that "we're all in this together."

Product owner: The product owner is the project's key stakeholder and represents users, customers and others in the process. The product owner is often someone from product management or marketing, a key stakeholder or a key user.

Scrum Master: The Scrum Master is responsible for making sure the team is as productive as possible. The Scrum Master does this by helping the team use the Scrum process, by removing impediments to progress, by protecting the team from outside, and so on.

Product backlog: The product backlog is a prioritized features list containing every desired feature or change to the product. Note: The term "backlog" can get confusing because it's used for two different things. To clarify, the product backlog is a list of desired features for the product. The sprint backlog is a list of tasks to be completed in a sprint.

Sprint planning meeting: At the start of each sprint, a sprint planning meeting is held, during which the product owner presents the top items on the product backlog to the team. The Scrum team selects the work they can complete during the coming sprint. That work is then moved from the product backlog to a sprint backlog, which is the list of tasks needed to complete the product backlog items the team has committed to complete in the sprint.

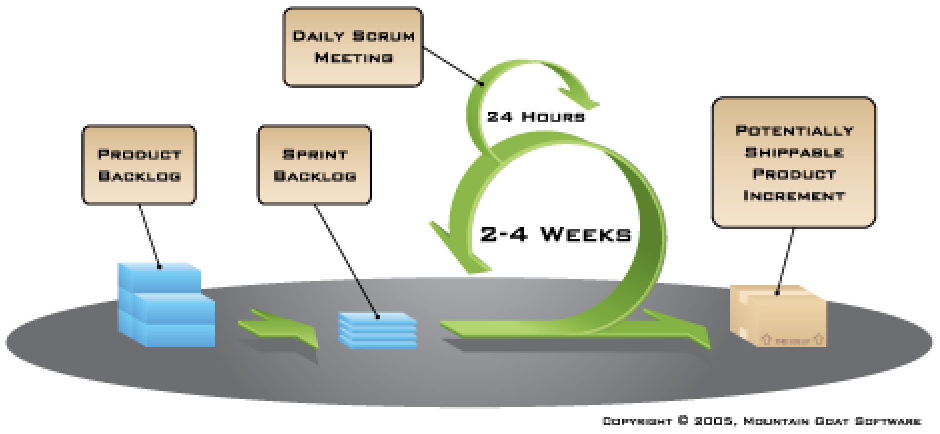
Daily Scrum: Each day during the sprint, a brief meeting called the daily scrum is conducted. This meeting helps set the context for each day's work and helps the team stay on track. All team members are required to attend the daily scrum.

Sprint review meeting: At the end of each sprint, the team demonstrates the completed functionality at a sprint review meeting, during which, the team shows what they accomplished during the sprint. Typically, this takes the form of a demonstration of the new features, but in an informal way; for example, PowerPoint slides are not allowed. The meeting must not become a task in itself nor a distraction from the process.

Sprint retrospective: Also at the end of each sprint, the team conducts a sprint retrospective, which is a meeting during which the team (including its Scrum Master and product owner) reflect on how well Scrum is working for them and what changes they may wish to make for it to work even better.

Each of the Scrum terms has its own page within this section, so be sure to check out all the pages in the navigation to the right to better understand the Scrum process.

Graphically, Scrum looks something like this:


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

This graphic is an introduction to the essential elements of using Scrum for agile software development. On the left, we see the product backlog, which has been prioritized by the product owner and contains everything desired in the product that's known at the time. The two to four week sprints are shown by the larger green circle.

At the start of each sprint, the team selects some amount of work from the product backlog and commits to completing that work during the sprint. Part of figuring out how much they can commit to is creating the sprint backlog, which is the list of tasks (and an estimate of how long each will take) needed to deliver the selected set of product backlog items to be completed in the sprint.

At the end of each sprint, the team produces a potentially shippable product increment — i.e. working, high-quality software. Each day during the sprint, team members meet to discuss their progress and any impediments to completing the work for that sprint. This is known as the daily scrum, and is shown as the smaller green circle above.

Life cycle of Scrum Methodology

Scrum framework allows you to implement Agile development methodology. Unlike the waterfall software development life cycle, the distinctive feature of Scrum is the iterative process of developing.

Development divides into several phases. Each of them results into a ready-to-use product. At the end of each step (called sprint in Scrum terminology) a ready product is delivered to a customer. Customer's feedback helps reveal possible problems or change the initial plan, if needed. If you want your project to strictly follow the main principles of Agile manifesto, you can use Scrum and be sure that you're on the right path.

Phases of Scrum Model

Step 1. Product Backlog Creation

Product backlog is a list that consists of features that should be implemented during the development process. It's ordered by priority and its every item is called a User story. Every user story gets a unique ID. This list below shows how these stories can look like.

ID	User Story
a-001	As a manager, I want to have the possibility to add, delete and edit tasks to manage the employees' workload
a-002	As a manager, I want to have the ability to add new tasks and change the duration and starting date of the current ones using drag-and-drop
a-003	As a manager, I want to assign two types of tasks to employees: -Part-time -Full-time task

The description of every user story should include the following required fields:

- **Importance** of a user story. It's acceptable to use any number you want
- **Initial estimate** describes the overall capacity of work. It's measured in story points
- **How to demo**. Describes the way of how the working product will be demonstrated

Besides these required fields, the optional ones can be added in case of need:

- **Track** is using to select all user stories of a certain type to change their priority. Can be used it to increase the priority of user stories that relate to the Control panel, for example.
- **Components** make up a list of components that will be changed during the work. For example, an application's modules, such as authentication or search.
- **Requestor** is a customer who's interested in implementing some particular functionality.
- **Bug tracking ID** contains a list of detected bugs that relate to a proper user story.

After the product backlog creation is finished you can move to the next step – sprint planning.

Step 2. Sprint Planning and Sprint Backlog Creation

Firstly, you should determine what your sprint's duration will be. A short sprint allows you to release the working version of a product more frequently. As a result, customer's feedback will be received more often and all the possible bugs and errors will be revealed in time.

As an alternative, you can prefer a longer sprint duration. It will allow developers to work more thoroughly. The optimal sprint duration is defined as an average of these two options. As a rule, a sprint lasts about **2 weeks**. What's more important at this phase is the cooperation between stakeholders and team members. The product owner determines the importance of a proper user story, while the scrum team defines the appropriate labor costs.

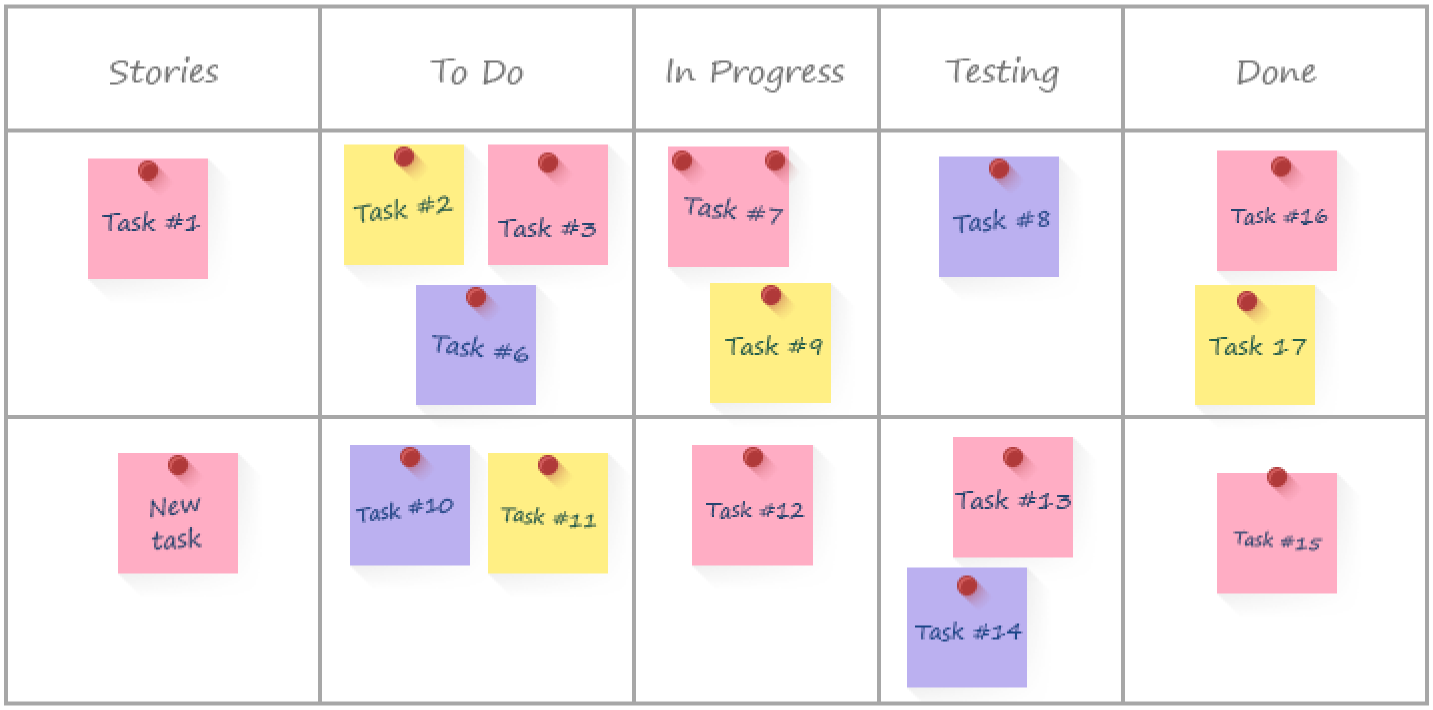
After that, the scrum team can select the most important user stories from the product backlog. Then team members should decide how they will solve this or that task. The **Sprint backlog** should be created next. It consists of user stories that will be completed during the current sprint. The amount of these stories depends on their duration in story points assigned to each story during evaluation stage. The team should be capable to finish all these stories on time.

Step 3. Working on the Sprint. Scrum Meetings

After actual user stories for the current phase are chosen, the website development process begins.

To track the current working process, a task board is commonly used. There are usually big cards with the names of particular user stories and a bundle of little sticky notes with description of single tasks which are needed for implementation of this or that story.

These cards are arranged according to their importance. When work on a task has been started, the corresponding sticker is moved from the "To do" field to the "In progress" one. When work is completed, sticker can be moved to the "Testing" field and after the task is successfully tested, the sticker goes to the "Done" field. An example of how the scrum task board can look like is shown below:



Stories	To Do	In Progress	Testing	Done
Task #1	Task #2, Task #3, Task #6	Task #7, Task #9	Task #8	Task #16, Task 17
New task	Task #10, Task #11	Task #12	Task #13, Task #14	Task #15

Step 4. Testing and Product Demonstration

Since the ideal result of every sprint is a working product, the full life-cycle testing process is very important. There are different ways to minimize costs of the testing period. For example, you can decrease the overall amount of user stories. As a result, the number of possible bugs will be minimized. The other way is to include QA engineers into the scrum team.

The result of every sprint is product demonstration. The Scrum team creates a review and demonstrates the results of their work. On this basis the stakeholders take a decision about further project changes.

Step 5. Retrospective and Next Sprint Planning

Retrospective's main aim is to discuss the results and determine the ways how to improve development process on the next step. The team should conclude what went well during the working process and what can be done better during the future iteration. When the ways of improvement are defined, the team can concentrate on the next sprint planning.

Important Facts about Agile Testing

The capacity to generate and react to change in order to succeed in uncertain and volatile conditions is what makes AGILE environment to stand at par.

Challenges faced at every phase of agile:

1. **Change is the only constant in AGILE:** The biggest challenge faced by every agile tester in their career. We cannot stop accepting change requests.
2. **Evaluating time for testing**: This varies for every project. During the sprint planning meet we need to fix the testing time, which is always tight.
3. **Lack of focused testing**: Actual testing work gets a setback when the focus is shifted more on other work like documentation, generating test data, environment setup issues.
4. **Accepting frequent builds**: During sprint execution, builds are accepted too often, which has the high possibility of being a broken build as the code is changed and complied frequently.

Major impact of these challenges is directly reflected on the Quality of the project. Functionality of application is hampered due to last moment changes, which ultimately results to project failure.

Focusing on basic Strategies can surely help us overcome some of these issues.

1. **Need to learn**-Continuous learning process helps us to sharpen our skills.
2. **Automate things** -Automation helps to have a tight grip on the changes. Try to automate as many of things possible. Use of different tools to makes the process easier and faster.
3. **Environment setup**- Keep test environment ready before sprint starts. Run dummy application to check whether the test environment setup is ready and up to date.
4. **Collaboration** – Build good relation with the project team. Good and healthy professional relation results in smoother and transparent collaboration.
5. **Retrospection meeting is important** -In this meeting give proper inputs on overall sprint execution process. Raise prior alarms whenever required.
6. **Communication over Documentation** -Document only those things which are required.
7. **Test Driven Development (TDD) approach** – Adopt TDD approach, this will eliminate basic issues in unit testing phase.
8. **Maintains Test Repository** -Like code repository, testing team can also create Test Repository. All the generic test cases, important links and test data can be stored in this repository. So we can reuse those whenever required

A willingness to communicate and collaborate is important to the success of agile. Earlier QA team has often been able to succeed by working as individual team by presenting them as gatekeepers of the product. Now with time change, we really need to work as a PROJECT TEAM and deliver best QUALITY PRODUCT.

Good Reasons to do Agile Development

Here are 10 good reasons to apply agile development principles and practices proposed by Kelly Waters.

1. Revenue

The iterative nature of agile development means features are delivered incrementally, enabling some benefits to be realised early as the product continues to develop.

2. Speed-to-market

Research suggests about 80% of all market leaders were first to market. As well as the higher revenue from incremental delivery, agile development philosophy also supports the notion of early and regular releases, and 'perpetual beta'.

3. Quality

A key principle of agile development is that testing is integrated throughout the lifecycle, enabling regular inspection of the working product as it develops. This allows the product owner to make adjustments if necessary and gives the product team early sight of any quality issues.

4. Visibility

Agile development principles encourage active 'user' involvement throughout the product's development and a very cooperative collaborative approach. This provides excellent visibility for key stakeholders, both of the project's progress and of the product itself, which in turn helps to ensure that expectations are effectively managed.

5. Risk Management

Small incremental releases made visible to the product owner and product team through its development help to identify any issues early and make it easier to respond to change. The clear visibility in agile development helps to ensure that any necessary decisions can be taken at the earliest possible opportunity, while there's still time to make a material difference to the outcome.

6. Flexibility / Agility

In traditional development projects, we write a big spec up-front and then tell business owners how expensive it is to change anything, particularly as the project goes on. In fear of scope creep and a never-ending project, we resist changes and put people through a change control committee to keep them to the essential minimum. Agile development principles are different. In agile development, change is accepted. In fact, it's expected. Because the one thing that's certain in life is change. Instead the timescale is fixed and requirements emerge and evolve as the product is developed. Of course for this to work, it's imperative to have an actively involved stakeholder who understands this concept and makes the necessary trade-off decisions, trading existing scope for new.

7. Cost Control

The above approach of fixed timescales and evolving requirements enables a fixed budget. The scope of the product and its features are variable, rather than the cost.

8. Business Engagement/Customer Satisfaction

The active involvement of a user representative and/or product owner, the high visibility of the product and progress, and the flexibility to change when change is needed, create much better business engagement and customer satisfaction. This is an important benefit that can create much more positive and enduring working relationships.

9. Right Product

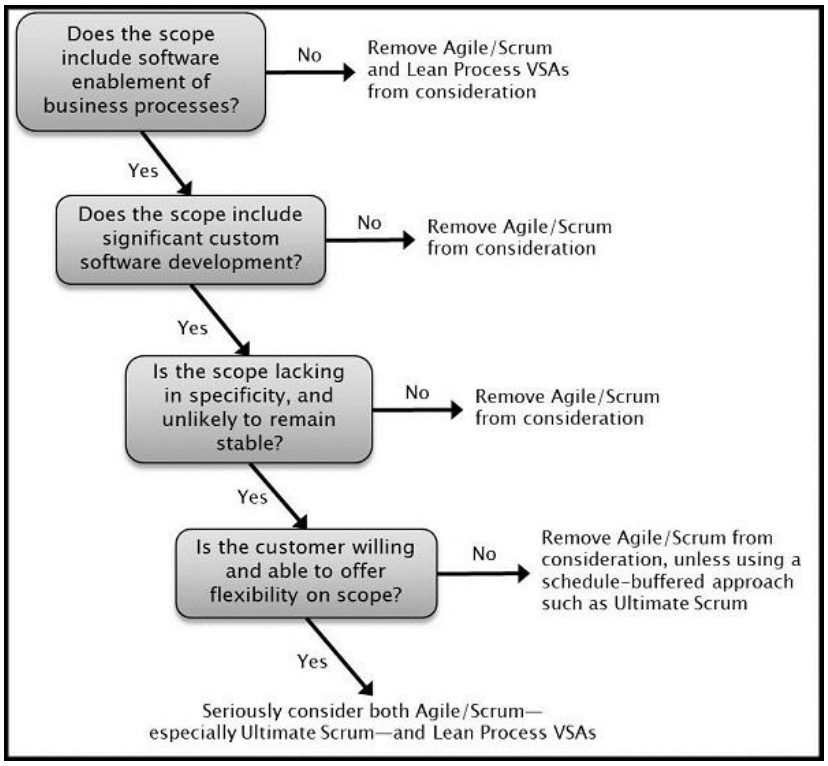
Above all other points, the ability for agile development requirements to emerge and evolve, and the ability to embrace change (with the appropriate trade-offs), the team build the right product. It’s all too common in more traditional projects to deliver a “successful” project in IT terms and find that the product is not what was expected, needed or hoped for. In agile development, the emphasis is absolutely on building the right product.

10. More Enjoyable!

The active involvement, cooperation and collaboration make agile development teams a much more enjoyable place for most people. Instead of big specs, we discuss requirements in workshops. Instead of lengthy status reports, we collaborate around a task-board discussing progress. Instead of long project plans and change management committees, we discuss what’s right for the product and project and the team is empowered to make decisions. In my experience this makes it a much more rewarding approach for everyone. In turn this helps to create highly motivated, high performance teams that are highly cooperative.

When to Use Agile, and When Not to

The decision tree in the figure below lays out logic for when Agile should be seriously considered, and when it should be removed from consideration.



Does the scope include software enablement of business processes?
No
Remove Agile/Scrum and Lean Process VSAs from consideration
Yes
Does the scope include significant custom software development?
No
Remove Agile/Scrum from consideration
Yes
Is the scope lacking in specificity, and unlikely to remain stable?
No
Remove Agile/Scrum from consideration
Yes
Is the customer willing and able to offer flexibility on scope?
No
Remove Agile/Scrum from consideration, unless using a schedule-buffered approach such as Ultimate Scrum
Yes
Seriously consider both Agile/Scrum—especially Ultimate Scrum—and Lean Process VSAs

1) Does the scope include software-enablement of business processes?

Both Agile and Lean Process VSAs offer significant benefits to software projects. Lean Process VSAs can help shrink the process footprint of both custom-developed and COTS software, significantly tightening scope to drive dramatic speed improvements. Neither Agile nor Process VSAs as we've proposed using them in our book will be very helpful on projects that are not software-enabling a business process.

2) Does the scope include significant custom software development?

Agile is far more beneficial for custom software development than it is for COTS implementations, while Lean Process VSAs can be very effective for both.

3) Is the scope lacking in specificity, and unlikely to remain stable?

Agile is far more applicable for scenarios in which scope requires significant refinement, and is likely to change frequently through the course of the project. Consider, for example, a construction project, for which the blueprints must be highly specified and must remain pretty stable. In software it's usually pretty straightforward to remove a chunk of code during the course of execution; in contrast, it's not so simple for a construction project to remove, say, an entire floor that's already been built. While software projects might typically lack this level of specificity, this is not always the case.

4) Is the customer willing and able to offer flexibility on scope?

Agile defaults to scope buffers to help manage project risk—specifically, the part of the product backlog comprised of “nice-to-have” features, as opposed to the “must-have’s.” If things go wrong and you’re only able to deliver on the must-have’s, then the project is still successful, even if only minimally so. If things go well and you’re able to deliver on the entire product backlog, then you will have exceeded customer expectations; but this only works if the customer is willing and able to offer flexibility on scope. If they don’t stipulate any “nice-to-have” features, or are unwilling to pay for them, then the scope flexibility that Agile prefers simply won’t be there.

If all four of these questions can be answered affirmatively, then Agile will likely be your most effective choice of project-level delivery methodology (at least for the software-development tasks).

---The End---