Table of Contents

What Does It Mean To Be Done? 3
What is the Goal of the Definition of Done (DoD)? 3
How Does the Definition of Done Help Us Address Technical Debt? 4
Who is Responsible for the Definition of Done? 5
Are There Different Kinds of Definition of Done? 5
What is an Example of a Definition of Done? 6
Want to Know More? 7

What Does It Mean To Be Done?

Or what is the “Definition of Done”?

One of the key ideas in Scrum and Agile is to focus delivering valuable work (or more basically, running, tested features.) There are two key tools in understanding when something is complete:

- Acceptance Criteria (or Conditions of Satisfaction): From an end-user perspective, what does it mean for this slice of a feature to be considered complete. Here we talk about the workflow that is being supported. In other words, the Acceptance Criteria describe the “extrinsic” quality standard of the work, what external people directly see; it answers the question “How does the work help me (the customer) get the job done”
- Definition of Done (DoD): From an Scrum or Agile Team perspective, have we done everything we need to in order to say this is a good piece of work. Here we talk about testing, coding standards, documentation and so on that tells us that all the work is complete. In other words, the DoD describe the “intrinisic” quality standard of the work, what external people don't usually see, except afterward when they see defects, for example.

What is the Goal of the Definition of Done (DoD)?

The primary goal of the DoD is for every team to be able to say “we are potentially shippable at the end of every Iteration (Sprint).” The idea is that we can then let the business make decisions on when to release based on the value provided as opposed to the current place we are in the release plan.

While this is important from a business perspective, the real benefit of the “potentially shippable” standard is that at any given time we know the status of the software and have low risk going forward because we have done all the work required in order to release our software. At many organizations I have worked at most of the (legacy systems and) software is not in this state and so part of this discussion is that we need to be working to get closer and closer to this state.

And finally, by leveraging the DoD we can establish a quality standard for the Team which helps ensure that we are not creating new Technical Debt as a result of the work we are doing, and also allows us to incrementally address the Technical Debt we already have in the code base.

In summary, the “Definition of Done” (DoD) helps us:

- Understand what it means to be done from an internal perspective.
- Understand progress we are making by being clear that we are showing “done” work (with no remaining work required)
- Reduce the build-up of Technical Debt.

Because of this:

> “A stronger 'definition of done' will always increase velocity and improve quality” - Jeff Sutherland at Agile 2008

How Does the Definition of Done Help Us Address Technical Debt?

The DoD helps us deal with Technical Debt in two ways:

1. We ensure the Team does not cut corners thus intentionally increasing their technical debt.
2. We ensure the Team can do the small incremental work to regularly improve the quality of existing work.

Many people think that because you are doing all this work you are slowing delivery down. Reality is that re-work takes a lot more time than completing work correctly in the first place, but the work is often hidden and so it really just seems this way. Bottom line is that the Lean discussion has taught us that if you focus on quality you will increase how much you deliver over time. This is a classic example of:

> “Go slower to go faster” - Unknown

By leveraging the DoD, Teams make sure they do not cut corners in on their work. The DoD establishes the quality standard for the Team. As professionals we know cutting corners slows down all future work and is not good practice. Often we will pretend to ourselves that we will come to back to it later. But the reality is that:

> “Later = Never” - LeBlanc's Law

By having a public Definition of Done we can help ensure that we are able to maintain our quality standard. The plan for the Iteration (Sprint) takes into consideration all the things that are part of the DoD and so we are able to complete quality work. It's up to us to ensure we don't add to the mess; we need to maintain our quality standards:

> “A mess is not Technical Debt; a mess is just a mess” - Uncle Bob Martin

We also know that left alone without care, existing code rots. Its kind of like entropy applied to code; if you don't apply effort to the code it will decay. The DoD can help here to. As part of the DoD we can think about an incremental approach to improving code. For example, perhaps our DoD includes the Boy Scout rule “Leave the code in better shape than when you found it.” This allows you to do random acts of kindness to the code; if the naming of variables doesn't make the code easy to follow, fix them so it does; if it doesn't have an automated unit test, put one in; if it has a Cyclomatic Complexity above 21, refactor so it is less; and so on. Over time this kind incremental improvement will result in a code base that is easier to work on.

There are a lot of ways to focus this incremental improvement. For example focus on improvement before you start working on a new capability:

> “Make the change easy; then make the easy change.” - Kent Beck

Or establish a working agreement whereby as you work an area of code and see something that needs fixing, you create a reference and estimate the changes needed. Then, the next time someone on the Team has to go into that code base, you ensure that the estimate that you provide to do the new work also includes the work required to fix the existing problems.

No matter what, do not underestimate the impact of incremental improvements - see How Do Small Changes Lead to Big Improvements? for more information.

Who is Responsible for the Definition of Done?

Agile Teams are responsible for maintaining their quality standard. They do this by:

- Working to their DoD
- Working to make their DoD tighter and tighter as they become more effective at delivering software.

Having said that, the Team's DoD will not be a purely Team creation. Definitions of Done will be influenced by the business. The input will come in to support differing levels of need:

1. Corporate: Corporate standards that need to be met
2. Portfolio: Portfolio standards that need to be met
3. Program or Release: Criteria that must be met to consider a Release ready for production
4. User Story: Criteria that must be met to know that all the work is complete
5. Agile Team: Things the team learns it needs, especially at a technical level

Teams have a known, public DoD. Teams are explicit both in being visible about their DoD (for example, available on the Team's page). They are also transparent about the practice they use to ensure their Definition of Done is addressed. It is up to the team to determine their approach.

Example approaches we have seen include:

- Teams create tasks associated with their DoD so that when the sub-tasks are done, the story is done.
- Teams maintain a spreadsheet and check-off the definition of done at the end of the Sprint.
- Teams create a task that is the Definition of Done checklist that is updated as work proceeds on the story

Are There Different Kinds of Definition of

Done?

As said, the aim of this DoD is to ensure we are getting closer to “potentially shippable” and that we are addressing technical debt. The DoD typically is created to cover most of the work that the Team typically does. So if you are a company doing new feature or maintenance work, then your DoD would be created to match most of that work. Other DoD's might be required for other kinds of work:

- If your team does work that is not necessarily new feature or maintenance oriented. For example a documentation or QA team would have different considerations. (Note: this is not an ideal set up but still does happen - see What Should We Consider When Forming a New Team for more discussion here.)
- If your team identifies different types of work. For example, a research user story (spike) would have different DoD than development work.

Most of the examples that you will see are DoDs related to software delivery. If you are doing other types of work - hardware, marketing, finance - then your DoD will be substantially different. The key thing for the DoD is that all the stakeholders understand what it means when the team says it is done so communication is a required part of the on-going work associated with the DoD.

What is an Example of a Definition of Done?

Here is an example DoD for a team, for new product and on-going maintenance development work:

- Code is complete:
  - Build and package changes are communicated to build master (e.g., when introducing a new file)
  - Code meets product coding standards
  - Code has been run through static and/or dynamic analysis tools (e.g., Coverity)
  - Code is checked into source control (with developer and peer reviewer names recorded)
  - Internationalization considerations have been met
  - Internal and external documentation is updated with applicable support information included such as help files and public APIs
  - Code is integrated with the main trunk (if not already done)
  - Code (or other deliverables) has been peer reviewed (e.g., pair programming counts as peer review)
- Code meets quality standards
  - Test plans are in place for: Unit, Function, GUI
  - Test plans have been run and passed and (hopefully) automated
  - Test plans are in place for the '-ilities' including performance, stress, load, security, etc
- No new defects
- End-user documentation is complete
  - Internal and external documentation is updated with applicable support information included such as help files and public APIs

- “Acceptance Criteria” (or “Conditions of Satisfaction”) or have been met
  - Some organizations also add in “Product Owner has accepted the story” as part of this thinking
- Item statuses in related tracking systems (Jira, Siebel, etc.) have been updated
- Code area is left in better shape than when we started the work on this user story

Want to Know More?

- How Do We Initially Setup Our Definition of Done
- How Do Small Changes Lead to Big Improvements?

Consultant, Tools, DefinitionOfDone, DoD, FirstSprint, FAQ

From:
https://www.hanssamios.com/dokuwiki/ - **Hans Samios' Personal Lean-Agile Knowledge Base**

Permanent link:
https://www.hanssamios.com/dokuwiki/what_does_it_mean_to_be_done

Last update: **2021/04/28 12:45**