An Agile project has ceremonies (sprint planning, release planning, sprint retrospective, etc.) and metrics (sprint & release burndown charts) designed to ensure that the project is in a healthy state. Unfortunately, many Agile projects fail even after following all the ceremonies religiously. Why? One of the primary reasons is that they are not able to deliver value to the customer at the end of each sprint. They deliver a product at the end of each sprint, but not a product that is potentially shippable. Here lies the root cause of failure. The software these projects deliver at the end of each sprint is half tested, half documented, half refactored and only half ready for release.

An explicit and concrete definition of done may seem small but it can be the most critical checkpoint of an agile project. Without a consistent meaning of done, velocity cannot be estimated. Conversely, a common definition of done ensures that the increment produced at the end of sprint is of high quality, with minimal defects. The definition of done is the soul of the entire Scrum process.

Before we explore a definition for done, it is important to define another Scrum term: potentially shippable product.

What is a potentially shippable product?

A potentially shippable product is one that has been designed, developed and tested and is therefore ready for distribution to anyone in the company for review or even to any external stakeholder. Adhering to a list of “done” criteria ensures that the sprint product is truly shippable.

The agile community differs in their opinion of what is shippable and what is potentially shippable. Many agile practitioners believe that potentially shippable is just one sprint away from the release. Others believe that potentially shippable means just that, ready to go to the customer immediately following the sprint. What does this mean for our definition?

For the sake of building a reference, done in this paper means that the features developed in each sprint should be 100 percent complete, whatever that means for your particular circumstances.

Am I finished with a feature?

You are not finished with a feature until it meets the benchmark defined by done. A very simple definition of done can be encapsulated as follows:

- Code Complete
- Test Complete
- Approved by Product Owner

This simplistic benchmark can ensure a favorable set of practices and the discipline to generate an increment of a product, but does not suffice as a reference.

An improved definition would contain the following details:

- Code Complete
- Unit tests written and executed
- Integration tested
- Performance tested
- Documented (just enough)
Better. But, it still does not ensure that the product owner is getting a potentially shippable product at the end of the sprint.

For example, is just mentioning “code complete” enough? Would doing so ensure that source code has been checked in properly in the code library using the best source control management practices and that the code adheres to the established coding guidelines? Would it ensure that the source code has been merged with the release branch and properly tagged?

Projects need better guidance in terms of what a done means, otherwise the end product of the sprint cannot truly be called shippable.

Building a definition of done

It is difficult to pin down a definition of done that suits all circumstances. The following definition is an attempt towards creating one common definition. This is not end of the road but a humble beginning. This, again, is intended to be high-level guidance that projects can tweak to suit their environment. In this attempt, different scenarios are considered to ensure necessary checkpoints are met.

Please be advised that this is intended just a reference and not as a done definition for a particular project. When it comes to agile methodologies, the KISS principle (Keep it Simple, Stupid) should be followed. Even seasoned practitioners would be scared of having such an exhaustive checklist. Therefore, use this just as a reference in building a done definition for your project.

A general definition of done can be spelled out in a “done thinking grid” (see Figure 1). The grid components are defined and explained below.

COLUMNS ONE

User Story Clarity
User stories selected for the sprint are complete with respect to product theme, understood by the team, and have been validated by the detailed acceptance criteria.
**Tasks Identified**
Tasks for selected user stories have been identified and estimated by the team.

These first two topics are typically covered in the sprint planning meeting but are mentioned as done criteria for the sake of verification.

**Build and package changes**
Build and package changes have been communicated to the build master. These changes have been implemented, tested and documented to ensure that they cover the features of the sprint.

**Product owner approval**
Each finished user story has been passed through UAT (User Acceptance Testing) and signed off as meeting requirements.

**Updating Product Backlog**
All features not done during the sprint are added back to the product backlog. All incidents/defects not handled during the sprint are added to the product backlog.

**Column Two**

**Environment ready**

- Development environment is ready with all third-party tools configured.
- Staging environment is ready.
- Continuous integration framework is in place. The build engine is configured to schedule hourly, nightly, and release builds.
- Desired build automation is in place. Why "desired"? Because there is no end to build automation :)
- Test data for the selected features has been created

**Design complete**
Design analysis is complete as per the user story or theme. UML diagrams are either created or updated for the feature under development.

You might need to prototype various components to ensure that they work together. Wireframes and prototype have been created and approved by the respective stakeholders.

**Unit test cases written**
Unit test cases have been written for the features to be developed.

**Documentation Ready**
Documentation (Just enough or whatever the team agrees to) to support the sprint demo is ready.

**Pre-Release builds**
Pre release builds (hourly/nightly) have been happening and nightly build reports have been published on regular basis. The following could/should be part of pre-release builds:

- Compile and execute unit test cases (mandatory)
- Creation of cross reference of source code
- Execution of automated code reviews for verification of coding rules
- Code coverage reports are generated
- Detection of duplicate source code
- Dependency analysis and generation of design quality matrix (static analysis, cyclomatic complexity)
- Auto deployment in staging environment
It comes down to build automation; there is no end to what can be achieved from automated hourly, nightly builds. The team along with the product owner needs to decide on how much build automation is required.

**Column Three**

*Code Complete*
Source code changes are done for all the features in the “to do” list.” Source code has been commented appropriately.

*Unit testing is done*
Unit test cases have been executed and are working successfully

*Code Refactoring*
Source code has been refactored to make it comprehensive, maintainable and, amenable to change.

A common mistake is to not keep refactoring in the definition of done. If not taken seriously, refactoring normally spills out to next sprint or, worse, is completely ignored.

*Code checkin*
Source code is checked in the code library with appropriate comments added.

If project is using tools which help in maintaining traceability between user stories and the respective source code, proper checkin guidelines are followed.

*Code merging and tagging*
Finalized source code has been merged with the main branch and tagged appropriately (merging and tagging guidelines are to be used)

**Column Four**

*Automated Code reviews*
Automated code review has been completed using the supported tools/technologies. Violations have been shared with the team and the team has resolved all discrepancies to adhere to the coding standard. (Automated code reviews should be hooked up with CI builds.)

*Peer reviews*
Peer reviews are done. If pair programming is used, a separate peer review session might not be required.

*Code coverage is achieved*
Code coverage records for each package are available and whatever the team has decided as the minimum benchmark is achieved.

*Project metrics are ready*
Burndown chart has been updated regularly and is up to date.

**Release Build**

- Build and packaging
  A Build (successful) is done using continuous integration framework. Change log report has been generated from Code Library and Release notes have been created. Deliverables have been moved to release area.
- Build deployment in staging environment
  Build deliverables are deployed in staging environment. If it is easy, this step should be automated.

**Column Five**

*Functional testing done*
• Automated testing
  All types of automated test cases have been executed and a test report has been generated. All
  incidents/defects are reported.
• Manual testing
  Quality assurance team has reviewed the reports generated from automation testing and conducted
  necessary manual test cases to ensure that tests are passing. All incidents/defects are reported.
• Build issues
  If any integration or build issues are found, the necessary steps are repeated and respective “Done” points
  are adhered to.

Regression testing done
Regression testing is done to ensure that defects have not been introduced in the unchanged area of the
software.

Performance testing done
A common mistake is to not keep performance testing in the definition of done. This is an important aspect.
Most performance issues are design issues and are hard to fix at a later stage.

Acceptance testing done
Each finished user story has been passed through UAT (User Acceptance Testing) and signed off as meeting
requirements (see also Product Owner Approval).

Closure
All finished user stories/tasks are marked complete/resolved. Remaining hours for task set to zero before
closing the task.

• Other necessary/optional activities
  Anything which is very specific to the project environment
  These might include:
    - Security audit sign off
    - Deployable on multiple platforms such as Windows, Linux, and Mac OS X

Summary
Preparing a single definition of done that suits every situation is impossible. Each team should collaborate and
come up with the definition that suits its unique environment.

Organizations that have just started with agile may find it difficult to reach a mature level immediately;
therefore, they should take the steps, sprint-by-sprint, to improve their done definition. Feedback from
retrospective meetings should also help to improve the definition of done.

Whether you use this grid or develop one of your own, remember that solid definition of done is essential to
producing potentially shippable products at the end of each iteration.