DEFINITION OF DONE

Outcome

The definition of done is a brief but precise description of the state that the work needs to be in to be considered 100% complete.

Function

There is an old joke in software development that software products spend most of their time 90% done. The joke is founded on truth: 'nearly done' affects project predictability, and is a major cause of software project overrun and failure. Until work is 'totally done', experience tells us that we can't be certain how much more needs doing and how long it will actually take to do it.

In iterative/incremental development we have the opportunity to prevent significant late slippage of release dates by ensuring that we progress in a controlled series of steps towards a releasable product. This means preparing a part of the product that is ready for release each and every iteration throughout the development cycle.

Benefit

Without a precise, rigorous, shared and agreed Definition of Done, we often fall into the trap of getting things only '90% done', rather than successfully released and used.

Implementation

1. As a team we agree, write up and publish a set of bullet points that form the rules for what is 'usable, releasable, shippable, acceptable'. These include what needs to be done to achieve our definition of done and how we can prove 'done' has been achieved. These might include criteria such as:

Who

   - Apart from working code, what other supporting user materials or documentation is necessary for the requirement to be acceptable and usable?
   - What types of test need to be passed?
   - In what environments?
   - What levels of test coverage are required?
   - Who needs to agree that the defined set of tests for a requirement is the right set?
   - What is an acceptable level of escaping defects (how many, of what severity)?
   - Who decides what severity the defects are?

Scaling Factors

   - What design standards or code standards need to be adhered to?
   - How is code quality and adherence to standards checked (e.g. peer review, code quality metrics tools)?
   - To whom does the product need to have been demonstrated?

Difficulty

   - Who decides what feedback needs to be responded to and what does not?
   - Who ultimately accepts the requirement as 'done'.

2. As a team we ensure that the definition of done is fully achieved for each and every requirement that we implement before we declare that it has been done.

Potential pitfalls

- Inadequate Definition of Done. When teams settle for a weak definition of done, they often lay up problems that could derail the project just prior to release. Final testing, integration, support documentation ... these can feel like minor matters in the excitement of demonstrating working code, but if not finished then the time required to finalise them is unpredictable. The result tends to be schedule slippage or poor quality. A key sign if a final time-box is scheduled for 'release hardening' – in a project where each iteration is truly done, this should not be necessary.
- An overly stretching Definition of Done. If the team's definition is actually beyond their remit or control, it becomes unachievable. The team then consistently fail to achieve it every iteration, which can be demotivating.

If you want to learn more, consider reading:
The Art of Agile Development by James Shore and Shane Warden