Definition of Ready & Definition of Done

Joseph Barjis

/PhD, SAFe® SPC 4.6
Contributor to SAFe® Framework/

PART 1: Concepts and Foundation

Definition

The concepts of Definition of Ready (DoR) and Definition of Done (DoD) are terms used to reinforce *Transparency*, assure *Built-In Quality*, and set the right expectations for the work items to be planned, developed, and completed during an Agile product development.

Why?

Written DoR and DoD are tools of communication between parties, e.g., Product Owner and Development Team to set clear expectations – *Transparency*.

A formally agreed upon standard and common understanding will reduce reworks and defects by taking care of quality during building the product not through product inspection – *Built-In Quality*.

DoR and DoD set a quality standard for all involved participants, therefore, it is crucial that teams themselves create their DoR and DoD, own them, and adhere to them. In creating these definitions, two things are crucial: consulting Agile (SAFe) principles and ensuring team's full agreement.

How?

DoR and DoD should be viewed as entrance and exit criteria, with DoR serving as a pre-condition checking if an item is ready to be taken into current step and DoD testing if an item is done to be moved to the next step.

DoR and DoD can be applied at any level (e.g., team, program, large solution, portfolio, enterprise, etc.) and to different items (e.g., User Stories, Features, Capabilities, Epics, Iterations, Releases, Program Increments, and even a whole Agile pilot program). The most common and basic application of the DoR and DoD concepts are seen at the team level as they are applied to User Stories.

These definitions must be visible in the teams working environment and used when items are moved from one step to another, especially when items are taken for execution and when they are moved to done!

The items these definitions are applied to must have binary outcome: Done or Not-Done. Avoid the temptation of an "almost done", "kind of done", or "99% done".

DoR and DoD Evolution

DoR and DoD are living agreements that evolve as the users learn better ways of doing their work: they acquire excellence in their work, their practice matures, their architectural runway or development environment improve, they acquire better tools, the context of their work changes, new regulations or compliance requirements arise, etc.

DoR and DoD are reflection of the current reality and should not impose a standard that is unrealistic. Over time, DoR and DoD improve by pushing the standards and quality higher and higher. In a sense, DoR and DoD can be used as an indicator of Agile teams' maturity and their Agile journey. Teams move from a high-level DoR and DoD to a more detailed level, from more flexible to more rigorous ones, and eventually achieving the capability of "Potentially Shippable Product" each Iteration.

This approach is a reflection of the Agile Manifesto principle *"Continuous attention to technical excellence and good design enhances agility"*.

DoR and DoD vs Acceptance Criteria

DoR and DoD are more generic and often applicable to a wide set of items, while Acceptance Criteria (AC) are specific for one item.

For example, at the team level, DoR and DoD apply to all team backlog items such as User Stories, while at the program level, they apply to all program backlog items such as Features, and so on.

DoD and AC are orthogonal concepts, and both are required to consider an item Done. DoD is a kind of superset of AC. Together, DoR and DoD ensure quality of an item and the AC ensure its functionality. Through AC, work items get more specific.

AC validate that the right item is built. DoD verifies that the item is built right, therefore Teams can include relevant NFRs (non-functional requirements) into their DoD as constraints on local design and implementation decisions.

DoR and DoD facilitate cultural changes and foster new mindset while AC target technical excellence.

DoR and DoD are the authority of the team, while AC are the authority of the customer or their representatives (Product Owner/Manager).

Autonomy vs Conformity

An enterprise might have certain requirements at the program or solution levels that must be incorporated into the DoR and DoD. For example, use of certain tools, a certain way of reporting, testing, funding, etc.

While teams have autonomy of creating their own DoR and DoD, some requirements may come as part of the enterprise policy or standard that require conformation of the teams. These latter requirements will be part of the, let say, fixed criteria for the DoR and DoD.

Beware of DoR Anti-Pattern

DoR can easily cross the line between Agile and Waterfall when the team is unable to pull any work or sufficient work for an Iteration until something else is completed. This is when a DoR turns into an anti-pattern. It is just fine to let in some user stories if they are not 100% complaint with the rules.

Remember, the iterative and incremental principle is also true about the rules contained in the DoR.

As per Mike Cohn: when a DoR "includes a rule that something must be done before the next thing can start, it moves the team dangerously close to stage-gate process… And worse, it can be a large and perilous step backwards toward a waterfall approach".

Some rules in the DoR must be met such as Acceptance Criteria and Estimation for User Stories, while some of the rules could be only desired.

As teams mature in their practice, their DoR may become redundant. It is only at the beginning of the Agile journey that teams need to follow some agreed upon guidelines in a written form.

PART 2: Guidelines for Creating DoR and DoD

These guidelines are developed in a generic fashion and can be adapted to any level (Team, Program, Large Solution, Portfolio) or purpose and activity such as Release, Iteration, Program Increment, or a whole Agile Pilot program.

Facilitate a workshop with the entire Agile team. Ensure that this is a whole team exercise. Depending on the level and purpose of the DoR and DoD, participants may include the product owner, product management, solution management, release management, architects, the development teams, system team, legal office, customers, vendors, and more. Arrange for online participants, if required.

Depending on the level and the size of the team, a 2 to 4 hours session is recommended. A follow up session (or sessions) can be organized if needed.

Ensure that the room has a whiteboard, supplies such as post-its, markers and sharpies, and enough space for collaboration and interaction. Accommodate for online collaboration, if required.

Creating DoR for User Stories/ Features	Creating DoD for User Stories/ Features

▪ Identify all the activities needed to get a User Story ready for Iteration Planning/ a Feature ready for PI Planning or Agile teams to write User Stories.
  ○ Focus on value-adding activities
  ○ These activities may widely vary depending on the product (software, hardware, mainframe, design or conceptual artifacts, etc.)
▪ Select only those activities that reoccur for almost every User Story/ Feature to become ready
▪ Record all these reoccurring activities into a comprehensive list of potential criteria for DoR
▪ Prioritize the list of the activities in a top down fashion
▪ Shortlist only those criteria that can be realistically met before a User Story is pulled for Iteration Planning/ Feature is pulled for PI Planning or Agile teams to write User Stories
  ○ The team decides how complete and perfect they want their DoR at this point
  ○ Pay attention to how Features estimation must be done prior to the associated User Stories Estimation and after the User Stories are estimated
▪ Get team's agreement and commitment for the shortlisted criteria, which are now their DoR for User Stories/ Features
	▪ Identify all the activities needed to complete a User Story/ a Feature.
	  ○ Focus on activities directed on the product quality (Built-in Quality).
	  ○ These activities may widely vary depending on the product (software, hardware, mainframe, design or conceptual artifacts, etc.)
	  ○ Approval or sign-off activities with external parties and stakeholders that are outside the team's control can be omitted.
	▪ Select only those activities that reoccur for almost every User Story/ Feature to be complete.
	▪ Record all these reoccurring activities into a comprehensive list of potential criteria for DoD
	▪ Prioritize the list of the activities in a top down fashion
	▪ Shortlist only those criteria that can be realistically met before a User Story/ Feature is complete
	  ○ The team decides how complete and perfect they want their DoD at this point
	  ○ Pay attention to what to do if a Feature can be considered complete, but it still has associated User Stories open
	▪ Get team's agreement and commitment for the shortlisted criteria, which are now their DoD for User Stories/ Features

Once DoR and DoD are created, they must be published, kept visible, ideally in the team's working area and Agile tools they use. Print it as a colorful poster that attracts attention to emphasize its importance.

Repeat the whole exercise described in the above guidelines on a regular cadence such as end of Iteration or PI to further expand and refine the DoR and DoD. Use the remaining criteria of the comprehensive list initially created and any new criteria ought to be added.

PART 3: DoR and DoD Examples

The DoR and DoD examples below contain general criteria for different levels. As stated, DoR and DoD are reality informed agreement by all participants, therefore, the table should be considered only as an example.

	Definition of Ready	Definition of Done
Product Development		
Team (User Stories)	**A User Story is Ready if, for example:** • Has Acceptance Criteria that can be tested objectively • Estimated by the entire development team • Socialized with the entire team • Has the right size that can be completed within a Sprint, preferably 2-3 days or not bigger than [certain] story points • Complies with the INVEST Model (Independent, Negotiable, Valuable, Estimable, Small, Testable) and has no external dependencies • Uploaded/ created in the team's Agile tool/ environment • Written in the user voice format: **WHO** <someone>, **WHAT** <do something>, **WHY** <some result or benefit>. E.g.: **AS A** <someone>, **I WANT TO** <do something>, **SO TAHT** <some result or benefit>	**A User Story is Done if, for example:** • Satisfies its Acceptance Criteria • Tested (e.g., for software/hardware) • Validated (e.g., for design documents, models) • Demonstrated to the stakeholders • There are no must-fix defects left • Documented • Accepted by the Product Owner
Program (Features)	**A Feature is Ready if, for example:** • Has Acceptance Criteria that can be tested objectively • Has the right size that can be completed within a PI • Has User Stories defined • Uploaded in the team's Agile tool/environment • Estimated (Story Points, T-Shirt size, …) • Written in the Features and Benefits (FAB) Matrix: **Feature** – A short phrase giving a name and context; **Benefit Hypothesis** – The proposed measurable benefit to the end user or business	**A Feature is Done if, for example:** • Satisfies its Acceptance Criteria • Tested and integrated • Demonstrated to the stakeholders • Documented and training provided • There are no must-fix defects left • Its required User Stories are completed, and any remaining User Stories are taken care, for example, decoupled from the Feature and moved to backlog, or deleted, or… • Accepted by the Product Management
Large Solution (Capabilities)	**A Capability is Ready if, for example:** • Has the right size that can be completed within a PI, although by multiple ARTs • Its associated Enablers for the required technical work are identified • Socialized with the Product Management of the relevant ARTs • Described using a phrase and benefits hypothesis, similar to Features: **Capability** – A short phrase giving a name and context; **Benefit Hypothesis** – The proposed measurable benefit to the end user or business	**A Capability is Done if, for example:** • Its associated Features are Done • It is deployed in the staging environment • Its associated NFRs are met • End to end integration, testing, validation and verification are done • There are no must-fix defects left • Demonstrated to the stakeholders • Documented and training provided • Accepted by the Solution Management
Product Release		
	A Release is Ready if, for example: • All Features/ Capabilities are done • End to end integration testing is done • Regression testing is done • Release documentation is complete • There are no must-fix defects left • User guide is created • User training is created • Approved by Product/Solution Management • Approved by the Release Management	**A Release is Done if, for example:** • End to end integration testing and V&V are done • Regression testing is done • Performance testing is done • NFRs are met • There are no must-fix defects left • The set standards are met • Training is delivered • Approved by the Product/Solution Management and Release Management

Conclusion

DoR and DoD are a check-list to verify that all value-added and quality driven activities are completed. This seemingly simple approach has a tremendous impact on the product quality, delivery, predictability, and accuracy with the work estimation.

While DoR and DoD are generic for all items in the category, not all of their activities might be applicable to each item (User Story, Feature, etc.). Thus, authority and discretion of the team are required to handle exceptions. Often, these definitions are a comprehensive list to ensure that all important activities are counted for.

In developing DoR and DoD, if one precaution must be undertaken, that is do not set the quality bar too high to make it unrealistic. Do not set it to the height of the best achieving team to only see the new teams fail or struggle to reach to it.

Anything that is not realistic to be achieved at the current level can be moved to the next higher level. For example, if integration testing is not possible within the Iteration, move it to the System Increment, if not, move it to Release, etc. This is what is considered attaining Agile maturity over time. Of course, the aim is to constantly shift quality left.

Finally, as the SAFe Framework as a whole, the DoR and DoD are scalable too, that is, the criteria are cumulative at each level.

Each higher-level definition assumes all the criteria of the lower levels. For example, the DoD at the program level assumes all the team level criteria plus the new criteria from the program level, and so on. Similarly, the Release criteria assume all predecessor levels plus the ones specific to the Release.

Reference

- Built-in-Quality by the Scaled Agile Framework Inc. https://www.scaledagileframework.com/built-in-quality/
- The Dangers of a Definition of Ready by Mike Cohn. https://www.mountaingoatsoftware.com/blog/the-dangers-of-a-definition-of-ready