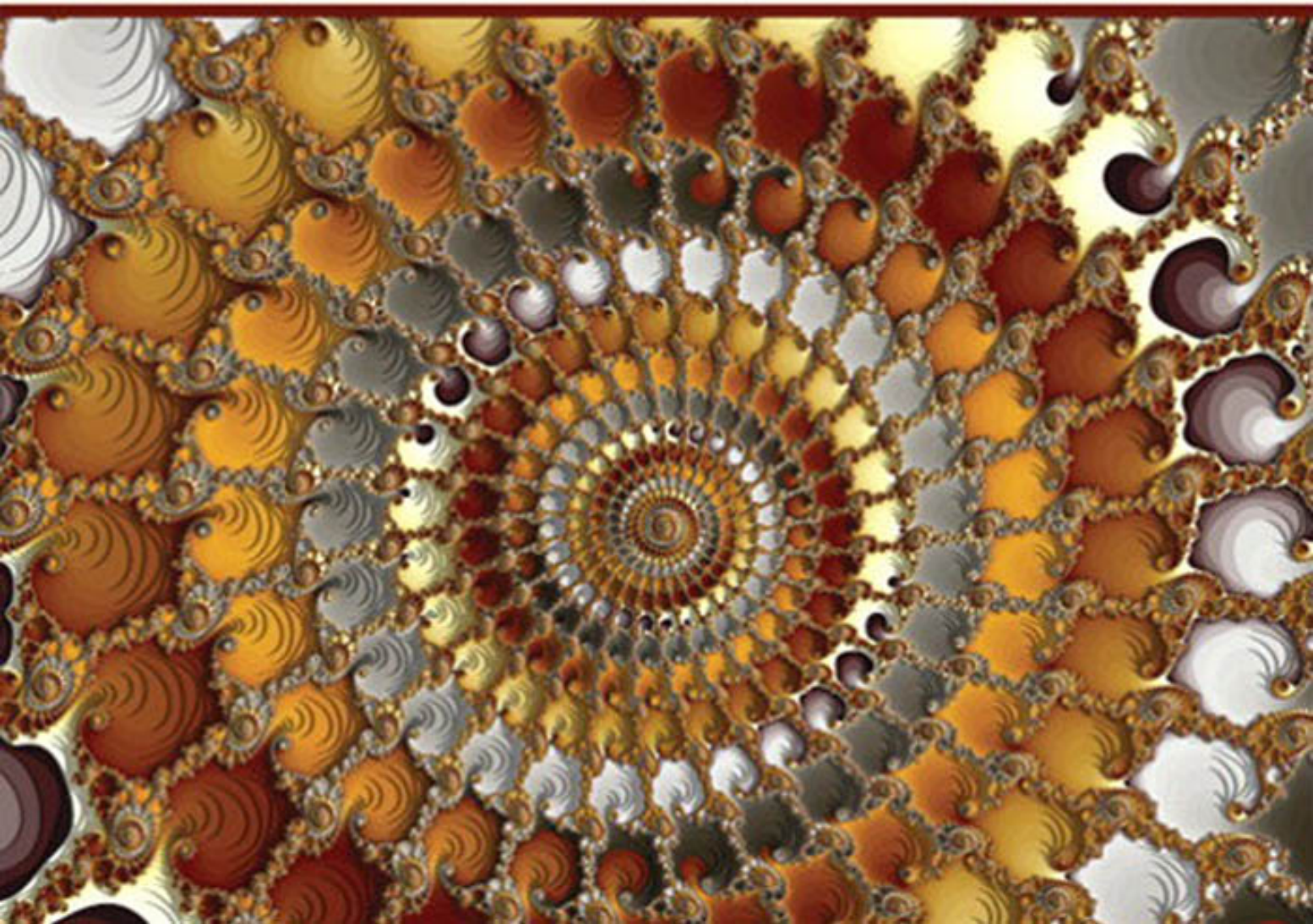
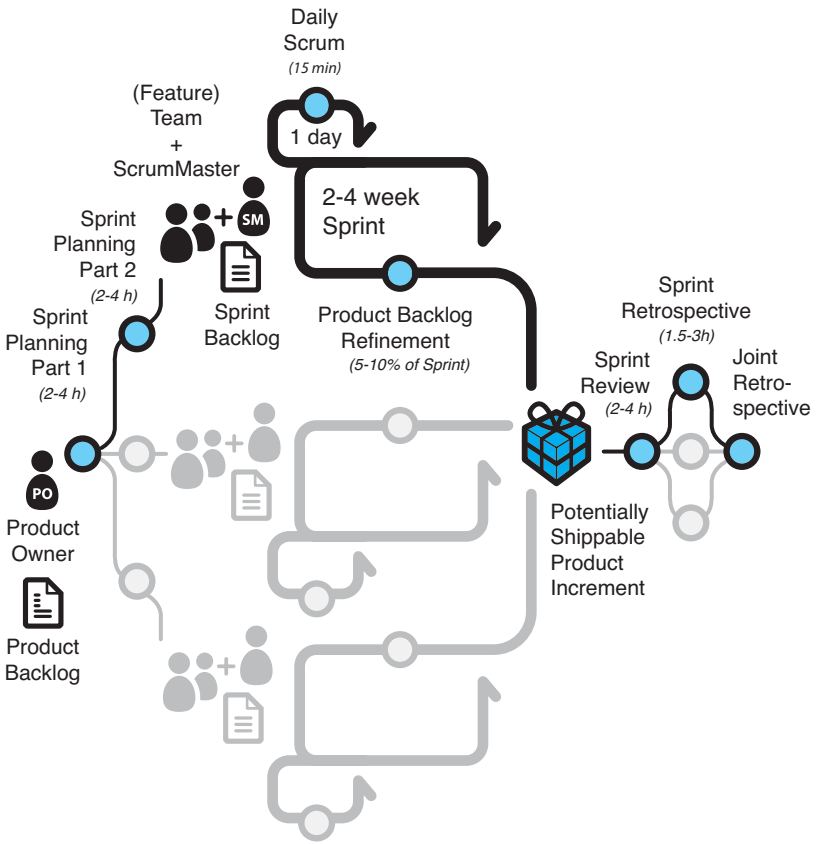
Practices for Scaling Lean & Agile Development

Large, Multisite, and Offshore Product Development with Large-Scale Scrum

Craig Larman
Bas Vodde



One Example Framework for Large-Scale Scrum



Do not confuse the DoD with *criteria of satisfaction* or *acceptance criteria*. The first are valid for all items in the Product Backlog, whereas the latter are item-specific criteria that evolve in acceptance tests, probably during a requirement workshop with A-TDD.

> Work of iteration = Product Backlog items + Definition of Done.

Here is an example (imperfect) DoD from one of our clients building a large product:

Category	Definition of Done	Details and Exceptions
Implementation	Item implemented in SW build.	
	No open faults caused by item.	
	Code peer reviewed & improved	
	Coding style followed.	For changes on top of 3rd party component such as Linux, we follow their existing coding style.
	Code cyclomatic complexity at 'good' level.	Complexity for new code below 15 for all routines, no increase in complexity for code changes based on inherited code.
	Static analysis done; no new warnings.	
Testing	Unit tests done for new and old changed code.	
	Unit tests under version control, peer reviewed & improved, and run automatically by CruiseControl.	
	Integration & functional testing done for items on the latest working platform build.	Integration and functional testing shall be done with the build which was available two working days before the Sprint Review.

Category	Definition of Done	Details and Exceptions
	Integration & functional test cases peer reviewed & improved; under version control.	Use the I&V Checklist in review.
	Integration & functional testing in target environment.	
	… more testing elements …	
Documentation	Requirements are documented, peer reviewed & improved, and under version control.	Requirements are recorded (1) in English in wiki or (2) as automated test cases in Robot Framework.
	Technical design documentation done, peer reviewed & improved, and under version control	In wiki; see "Design Documentation."
	API documentation updated, peer reviewed & improved, and under version control.	Automatically generated from source code and comments.
	Customer Doc Inputs done.	See separate input list for each guide-type customer document.

Product-level baseline

For meaningful tracking of overall progress, there needs to be a common product *baseline* DoD that all teams conform to. Our clients usually record this common definition in a wiki page.

In some cases, there is also an intermediate *requirement area* baseline that extends the product baseline.

When to first define it? In a special workshop that must include involvement by the hands-on teams. If there is contention about the original definition, the Product Owner has the final say.

When to evolve it? During Joint Retrospectives.

See "Try…Joint Sprint Retrospectives" on p. 403.

Why bother? Without a product-level definition, there is

- reduced visibility into the state of the overall product—it is not possible to track meaningful system-level progress
- less focus on the overall system—there is less discussion and attention to the state of whole product
- increased variability—different features can and will have different degrees of Undone Work
- reduced ability to deliver a potentially shippable product increment each iteration—some features can and will have more Undone Work than others

Avoid…Definition of Done defined by quality group

We coached a group in India that included a quality manager who wrote checklists and wanted teams to follow those. His focus was on centralized defined processes and top-down conformance. When this group started to adopt Scrum, he injected his checklists and centralized conformance into Scrum by defining the DoD. Avoid that—the DoD is an agreement between the *Product Owner and teams*. It is not an agreement with a quality group.

Avoid…Undone Work

When the DoD does not *yet* include all the work needed to ship the product to the customer—common in big groups first adopting

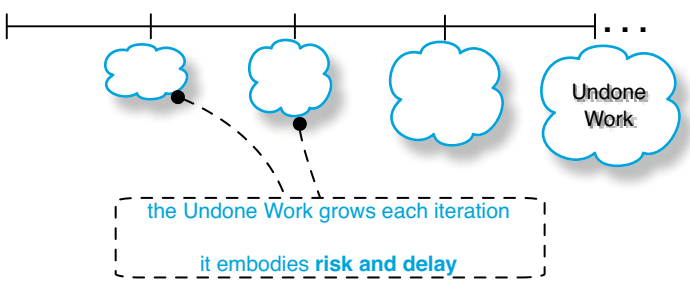
Avoid…Needing a Release Sprint

Scrum—then work is left before the product can ship. This is the **Undone Work**,[10] and it increases each iteration.

Potentially Shippable	
Done	Undone
• implementation	• performance tests
• unit tests	• reliability tests
• functional tests	• deployment
• architecture doc update	• marketing material updated
• customer doc	• support people trained

For instance, if the group cannot (yet) do performance testing each iteration (perhaps it is not automated and is outsourced to Vietnam), then this is Undone Work and the amount of performance testing increases over time, building up (implicitly) as a big batch of work in a queue to be done before release (Figure 5.8).

Figure 5.8 Undone Work grows over time



See "Try…"Undone Work" and system-level NFRs as PBIs" on p. 225.

Risk and delay—Since the Undone Work has to be done before shipping, it represents delay for the Product Owner—and therefore also responsiveness. Undone Work also represents risks—performance testing being an excellent example. When it is delayed until near the end of the release and the group discovers performance problems late in the game…bleh!

10. Not to be confused with unfinished work from the iteration. The Undone Work exists by intention or plan.

Also: See “Avoid…Try…Separate “Undone Work” from the Product Backlog” on p. 226.

Avoid…Needing to ‘harden’

Hardening—When we first visit a client new to lean or agile development, a word we frequently hear is…*hardening*. Such as, “We are planning to do three iterations, and then do a hardening iteration.” ‘Hardening’—an aspect of Undone Work—is so common that some do not see it for what it is: a *failure*, not a solution. In lean thinking, *defects* and then all that follows (test and fix at the end of a cycle) is considered waste. The lean perfection challenge is to *build quality in*—through *changing the system to* address the root cause problems—so that hardening and similar superficial *quick fix* reactions are no longer needed.

But, sometimes there still is Undone Work. Then, how to do it?

Try…Include Scrum teams in a Release Sprint

In small-scale Scrum, the standard solution to do the Undone Work is to hold a **Release Sprint** in which the regular Scrum team does all this work—they do not work on new features—and then releases the product (Figure 5.9). Good idea[11]—especially because it *educates* the team through painful experience about this Undone Work. Since they are “eating their own dog food,” the team is increasingly likely to think of ways to reduce it in the future—to improve the DoD and reduce the Undone Work.

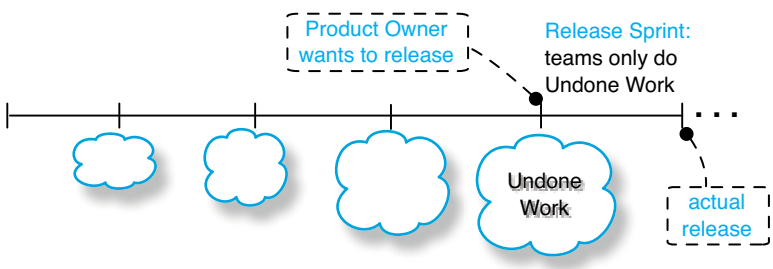


Figure 5.9 Release Sprint

11. The need for a Release Sprint is not good. Rather, if needed, including the teams is good.

see Organization in the companion

As a rule, big groups moving from sequential development have a dedicated "Undone Unit"—the group that handles all the Undone Work such as documentation for field engineers, acquisition of part numbers, system testing, fulfillment of legal or government regulations, and so forth. When there already exists a big *institutionalized* Undone Unit that has done this end-of-release work before, it is mighty tempting to fall back on old habits and simply, once again, hand over the Undone Work to them. And we have seen 'Undone' management groups with a stake in keeping the Undone Unit alive—they press the Product Owner (or someone) to give them all the Undone Work.

Resist that temptation.

Rather, (1) hold *one* Release Sprint with some or all of the regular Scrum feature teams. For exactly the same reasons as above—learning through eating their own dog food. Also, (2) mix Scrum team members with the Undone-Unit experts, to reduce handoff problems and improve two-way learning (Figure 5.10). Examples:

- One of our clients is a bank. They have a production operations group that, before adopting Scrum, received a candidate release and tested it in a sandbox. After adopting Scrum, regular Scrum team members physically join with (pair-work with) product operation people during a Release Sprint to do this testing in a sandbox.
  - This is a temporary phase. The longer-term change is for the production operations group to diminish, and some members join regular development Scrum teams. Then, regular Scrum teams will have development knowledge to help with operations.
- One of our clients builds ship-control systems. They have a ship-installation group that installs a control system (wires, hardware, software, …) in a ship. They are exploring including regular software-development team members on the ship during installation, to help. (This change is complicated by safety and insurance issues).