# If I am going to scale agile to most of my organization, what do we need to do at the portfolio level?

**June 2016**

## Scope of Report

In this report, we suggest some considerations for executives seeking to grow the number of agile teams in their organization. At some point, changes are needed at the top. In particular, the portfolio management team needs to reorganize the proposed software development work to allow it to be pulled by the programs and teams from a portfolio backlog prioritized by economic value.

## Definition

In "Agile Software Requirements," Dean Leffingwell (a founder of the Scaled Agile Framework® or SAFe® quotes Mikko Parkolla's description of portfolio management:

> *Portfolio management is a top-level authority that makes long-term investment decisions on strategic areas that affect the business performance of the company.*
>
> *The main responsibility of portfolio management is to set the investment levels between business areas, product lines, different products, and strategic portfolio investment themes; these are a collection of related strategic initiatives.*

Leffingwell goes on to list three sets of activities that should be under the control or influence of the organizations' software portfolio management team:

- Investment funding: Determining the allocation of the company's scarce R&D resources to various products and services.
- Change management: Fact patterns change over time, and the business must react with new plans, budgets and expectations.
- Governance and oversight: Assuring that the programs remain on track and that they follow the appropriate corporate rules, guidelines, and relevant standards.

As organizations start to establish Agile outside of a few isolated teams, introducing or refocusing a portfolio management team will be a big part of the change. To stand any chance of success, using business value as the driver for the improvement of software development, the leaders of the organization must first instil a culture of lean software development.

## Lean Software Development

The first step in establishing software value strategies is to learn about and commit to a philosophy of lean software development. We do not have space to dive too deeply into the theory and history of lean manufacturing and the evolution of lean software engineering in this report. Interested readers should refer first to "*Implementing lean software development*" by Mary and Tom Poppendieck. Instead, we focus on some important ideas from

lean and lean software engineering. In particular what we consider to be important principles that an organization should adopt before it attempts to drive its Agile software development across the whole business.

**Removal of waste**

A key principle of lean is that activities that do not add value should be removed from the process. "Activities that do not add value" is the definition of "waste" in lean. Of course, in the spirit of "kaizen" or continuous improvement, the phrase "do not add value," is interpreted relatively as in "Activity A adds less value than Activity B – how can we make an improvement?"

**Pull not Push**

Historically, in software development the business side of the organization has almost always created more ideas for work than the software development group can handle within the desired time or budget. This has led to competition between business heads to get their work prioritized by the software development teams ahead of their colleagues. Also, it led to a mentality that it was important to "push" as much work as possible into the software development group such that every available drop of capacity was utilized, even if this meant overloading and burning out development teams. This still happens today.

However, flow theory based on ideas from lean manufacturing and telecoms routing in Reinertsen's "The Principles of Product Development Flow," suggests that the strategy of bringing resources to projects and optimizing their utilization is a poorer strategy for delivering economic value than applying lean principles to the flow of work through small teams of expert resources. The lean answer is to establish appropriately-sized teams for the desired work steps and the available budget and then let the teams "pull" work, from a backlog of tasks that is ready to be worked on, as and when they are ready to do the work. This approach combined with Work In Process (WIP) limits at each step avoids queues of "work in process" throughout the team. Queues represent waste because there is no value being added while work items are sitting in queues. These principles are fundamental elements of "kanban" systems.

Importantly, "pull not push" requires a change in understanding at the strategic level. Executives must be persuaded, and accept, that in organizing software development under lean principles, to maximize flow they need to organize their desired deliverables into "portfolio backlogs" in which the desired deliverables are prioritized.

We must acknowledge that this is a big cultural change because most executives' experience (and ego?) tells them that the software development department is there to do what executives want, when they want it. This pushing of requirements and delivery dates onto software development results in projects that are short on scope, over budget and late. Many executives experience these results with disappointment and develop lack of trust in software development. With "pull not push", the executives get a promise of greater value flow, greater agility and some acknowledged uncertainty (grounded in reality). We believe that the change is worthwhile and the benefits are real but we acknowledge that the cultural change is difficult and important.

**Economic focus:**

This might seem obvious to a senior management team and they are likely to push back on the suggestion that they do not prioritize everything with an economic focus. Further, they might jump quickly from the term "economic focus" to the concept of business cases. Many of us have experience of business cases as "box checking" in the bureaucratic culture of some organizations. Such business cases do not add value and should be eradicated as waste from a lean perspective. The economic focus we want to engender in the executive team is that of maximizing the flow of value through software development. This means assigning relative business value to projects or epics at the strategic value. This means being aware that, as Reinertsen tells us, "If you only measure one thing then measure cost of delay." Cost of delay is a useful default metric that can be easily evaluated as a relative metric which includes business value, timeliness constraints and risk reduction or opportunity

enhancement. Further, when cost of delay is incorporated into the "Weighted Shortest Job First" metric then we have a more refined metric for prioritizing by business value that can still be easily and quickly assessed on a relative basis by a knowledgeable group of business and IT stakeholders.

## A Portfolio Management Kanban System

Figure 1: Portfolio Management Kanban Board

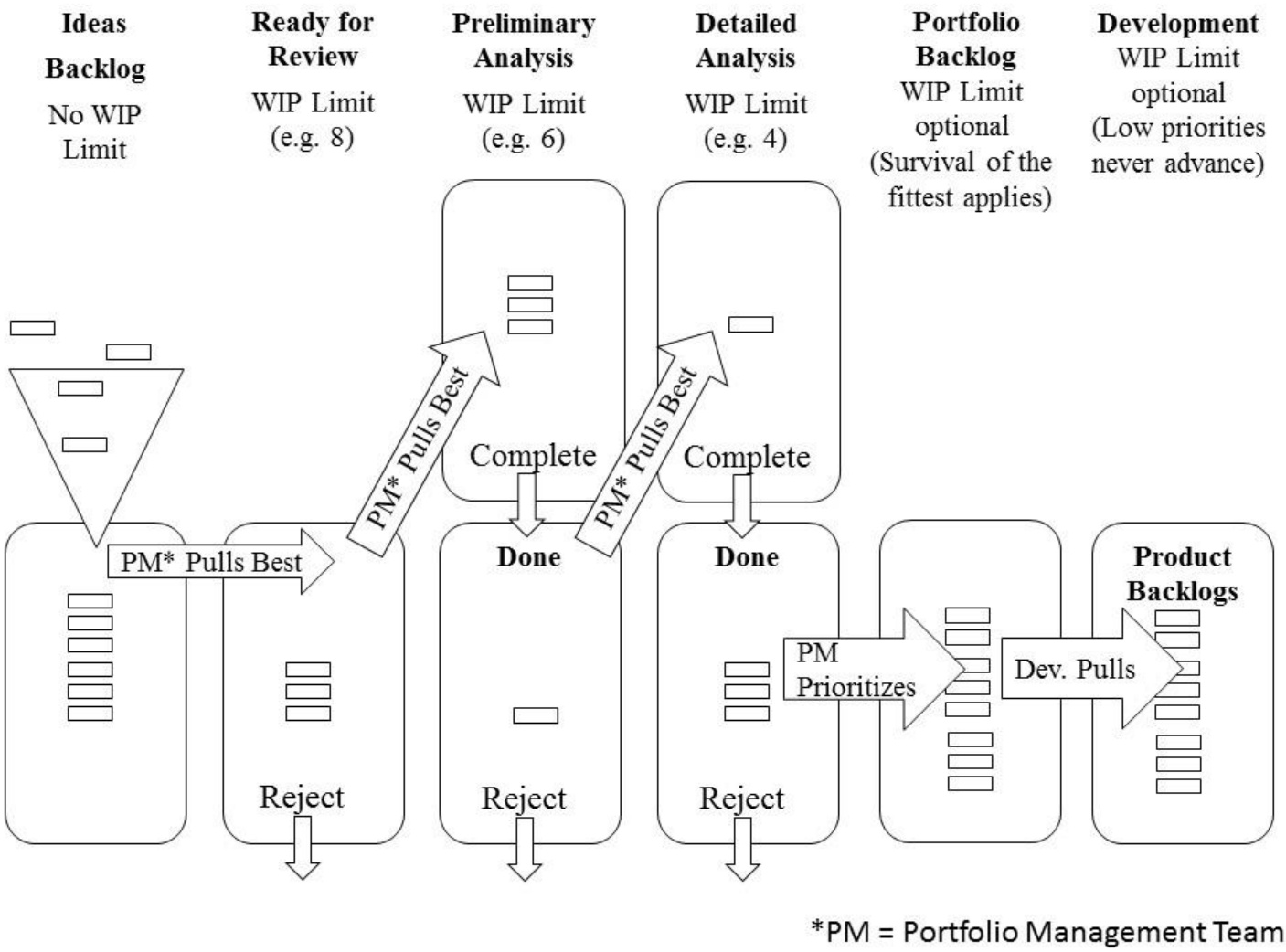


Figure 1 illustrates the sort of Portfolio kanban system that can be used at the portfolio level to prioritize new software development investment. It is important to emphasize here that the goal is prioritization of new software development projects (or epics) to maximize the flow of business value. This implies that we have a way to measure business value at this level and we cover a simple, default technique, "Weighted Shortest Job First", below. It also implies that some overarching strategic directions have been set for the organization against which competing investments with similar value (to different business units) can be prioritized.

The portfolio management system in Figure 1 is just one example and is not intended to be prescriptive. The process steps are as follows:

**Ideas Backlog:** Any and all ideas from the business go into this backlog. At this point, the ideas are high-level with limited descriptions and a simple value statement because there is no point in wasting effort (good lean practice!) in preparing detailed descriptions or value statements if these ideas are never going to make it to the next step. There is no WIP limit because there is benefit in considering all ideas and let the best float to the top. In practice, weak ideas will "age out" of the ideas backlog as better ideas continue to be prioritized above them.

**Ready for Review:** The key activity at this process step is for the portfolio management team to pull "the best" ideas from the ideas backlog for further analysis. Some sort of simple algorithm is required for establishing "the best." Probably, it should include inspection of the ideas for certain minimum information such as a business value statement and then a ranking based on that business value statement and the current strategic priorities of the organization. Some interaction between the portfolio management team and the business will be required to develop and document a refined understanding of the idea. There is a WIP limit on this step because the organization has limited funds to invest in software development, the portfolio management team has limited capacity and the WIP limit acts as a filter to accept on the best ideas. Ideas that are processed in this step may be rejected, returned to the ideas backlog or passed on for Preliminary Analysis.

**Preliminary Analysis:** In preliminary analysis, it is necessary to make a first pass at measuring relative business value. Note that this is a prioritization process so relative business value is more important that actual, specific business value. Further, it is unlikely that future business value flows can be predicted with any real accuracy so there is little point wasting effort to try. For relative business value at this point, we recommend using Cost of Delay and Weighted Shortest Job First (WSJF) – the relative values of these can be estimated by a workshop involving business, portfolio management and software development representatives. More details are provided below. The project with the highest relative WSJF should be done first. At this point, ideas become projects (or epics). Projects that are processed in this step may be rejected, returned to the ideas backlog or passed on for Detailed Analysis.

**Detailed Analysis:** At this stage, there is some certainty that the projects under consideration have real business value. Hence, it is appropriate for the portfolio management team to work with the business to develop a lightweight business case – a few pages of structured information – and work with the software development group to build some high level cost and duration estimates. Information from these documents should be used to refine the WSJF estimate. Projects that are processed in this step may be rejected, returned to the ideas backlog or moved into the portfolio backlog according to their relative WSJF priority versus the items already in the portfolio backlog.

To some extent, there is tension between strategy at the executive level being all about deciding on the big things that need to get done and Reinertsen telling us to reduce batch size which implies pushing small things through the portfolio backlog pipeline. Big ideas can certainly enter the ideas backlog but their chances of progressing quickly and smoothly through the portfolio management kanban system are limited by their size. Hence, savvy business heads will submit smaller ideas to the ideas backlog to increase their chances of getting investment quickly.

## Cost of Delay (CoD) and Weighted Shortest Job First (WSJF)

Before we consider Cost of Delay and Weighted Shortest Job First, we must describe a technique for estimating that uses the shared expertize of a group of informed people to achieve a relative estimate. The technique is based upon the Delphi Method developed by the Rand Corporation in the 1950's which was simplified as "Planning Poker" by Agile guru, Mike Cohn. The relative estimates for a set of items are considered by a group of informed people. First, the group agrees on which item has the smallest estimate. This is arbitrarily assigned a value of 1. Next, each individual considers another item from the set and privately evaluates its estimate relative to the smallest unitary item from a specified, limited set of numbers. Typically, a modified Fibonacci sequence is used

such as the Cohn Scale popularized by Mike Cohn for use in Story Points. For portfolio management purposes, we recommend a simple set: 1, 2, 3, 5, 8, 13, 20. Having privately decided on an estimate, the estimating group each share their estimates. The resulting reconciliation of individuals' estimates based on sharing of individual knowledge and justifications is the true value of the process because all the individual assumptions upon which the individual estimates were made are made explicit and accepted or rejected by the group. The process continues with consideration of the next item to be estimated. It should be noted that at this point we have not specified what attribute of the items the group is estimating nor what units they are using. The technique can be applied to any attribute and there are no units because the estimates are relative.

All projects, epics, stories, tasks have a Cost of Delay. The Cost of Delay is the hourly, daily or monthly cost associated with NOT starting the project. For our prioritization purposes, we are interested in the cost of delay if we start Project A while setting aside Project B i.e. the Cost of Delay of Project B. Consider two software projects, A and B, to be carried out by the same team over 60 days and similar in every way except that project A requires an external specialist in Oracle databases who can be hired at $100 per hour and project B requires an external specialist in Mumps databases who can be hired at $150 per hour. Both specialists are available and must be hired now. The team can only do one project at a time. Which should they do first? Well, they should do the project with the highest cost of delay first. If project A is deferred, it has a cost of delay of 60 days * 8 hours * $100 = $48,000. If project B is deferred, it has a cost of delay of 60 days * 8 hours * $150 = $72,000. Project B has the highest cost of delay so it should be started first.

For cost of delay in this example, we were able to use explicit financial values. More often, this sort of explicit financial information is not available or is, at best, fuzzy. So how would this have looked from a Planning Poker perspective? Probably, we would have been clear that project A had a smaller cost of delay than project B because Oracle databases are much more common than Mumps and therefore the supply of Oracle experts would probably be greater and their costs lower. We would assign project A an estimated Cost of Delay of '1'. The discussion around project B would revolve around how much more expensive Mumps experts are than Oracle experts. Probably, we would have assigned an estimated Cost of Delay of '2' (although in similar circumstances we have seen groups reassign project A an estimate of '2' so they can assign project B an estimate of '3' because they don't believe B is twice the size of A!). The key lesson here is we are seeking to prioritize; so the absolute dollars are not as important as their relative size in the set under consideration for prioritization.

In the above example, we were careful to use the same duration for both projects i.e. 60 days. What if the duration had been different? Let's say that project A, the Oracle project, had a duration of just 10 days. The cost of delay for project A would stay the same at $48,000 because project B still has a duration of 60 days. However, the cost of delay for project B is now 10 days * 8 hours * $150 = $12,000. In this case, project A has the highest cost of delay and should be started first. Clearly, duration has an impact on cost of delay and must be considered in prioritization decisions. Hence, we use, Weighted Shortest Job First, where:

$$Weighted\ Shortest\ Job\ First = \frac{Cost\ of\ Delay}{Duration}$$

Now, we prioritize highest those projects with the highest WSJF. Typically, we do not use Planning Poker to estimate relative WSJF. Instead, we use Planning Poker to estimate a relative Cost of Delay and a relative Duration for each project and then apply the formula for WSJF. In SAFe, it is recommended to break down Cost of Delay into three constituent parts which are estimated using Planning Poker and then summed to give the total relative Cost of Delay:

Cost of Delay = User or Business Value + Time Criticality + Risk Reduction or Opportunity Enablement Value

Since the same Cohn Scale numbers are used, this might effectively weight Cost of Delay at three times more than Duration in the WSJF formula but, in our experience, one of the three sub-components of Cost of Delay usually dominates in a given business case so the potential distortion is rarely realized in practice.

## SAFe Portfolio layer

In a pull organization, informed decisions about the prioritization of the items on the portfolio backlog are vital and add to the relevance and workload of traditional portfolio management teams. The activities associated with preparing items for the portfolio backlog justify the introduction of a software portfolio management team in those organizations that do not already have one. Hence, in SAFe, there is a Program Portfolio Management team at the Portfolio level.

The portfolio management team needs an algorithm for deciding which ideas might be best from the portfolio backlog which should include a ranking based on the current strategic priorities of the organization. At the Portfolio level, SAFe includes the concept of "Strategic themes" derived from the Enterprise goals to address this need.

In addition to driving prioritization decisions in the portfolio management kanban system, strategic themes should also drive the prioritization of budget allocation at the portfolio level. For example, if a bank decides that it wants or needs to build its business banking capabilities as a priority then that becomes a strategic theme. When the time comes to allocate annual budgets to software development value streams, it could be appropriate to allocate more of the budget to business banking (more software development teams) at the expense of, say, consumer banking (fewer software development teams).

## SAFe Value Stream Layer

The organizational structure in large enterprises most often consists of business units organized around functionality (e.g. HR, Design, IT, Manufacturing, etc.) or business units organized around products and services (e.g. Ford Trucks division, Ford cars division, etc.) or business units organized around customers/geographies (e.g. Consumer Division, B2B Division, North America, Europe, etc.). In many, if not most, large organization some combination of all of these is in place.

SAFe applies lean principles by seeking to orient and organize software development teams around the stream of business value "from concept to cash." Clearly, this "concept to cash" model is most easily understood in the context of the "for profit" end of our lens but restated as "concept to value." It can work equally well for the "not for profits." Also, this means that sometimes value streams are defined by individual business units in large organizations if the business units are organized around products/services or customers. More often, SAFe value streams involve parts of several business units.

## SAFe Program Layer

The next layer below the Value Stream layer in the end-to-end business value flow (and also the layer below Portfolio for smaller businesses or smaller SAFe implementation) is the Program layer which is also sometimes thought of as the Product layer. Again, the flow of business value is via epics being pulled into a Program Backlog where they may be decomposed into smaller units, organized in a kanban system with WSJF reviewed and updated.

## SAFe Team Layer

From the Program Backlog, in Agile software development groups, epics are decomposed into stories at the Team level and pulled into the sprint or iteration backlogs of individual teams. This is a key step because at this point,

teams control the prioritization of work flow. How the teams do the prioritization is important for the flow of business value and we have suggested some approaches in our work on value visualization (see Sources).

We have reached the point where requirements are turned into working code but we have not yet delivered any business value. In SAFe, integrated, working code is a deliverable of each two week iteration. The deliverables from these two week iterations are aggregated up and demonstrated as working code in Program Increments which may contain 4-6 iterations. Developing and delivering working code in structured time boxes like this is called "Develop on Cadence" in SAFe. Still we have not delivered business value into the hands of the users and business. In SAFe, this last step is decoupled from the delivery of working code. A release management team prioritizes the working code that best suits the immediate needs of the business and releases it into production, usually with the assistance of a DevOps team.

## End-to-end business value

Finally, we have delivered business value and the end-to-end system is complete. But let's return to the question that opened this book: How do we know how much business value we have delivered? Does this represent the maximum that we could have delivered?

In SAFe, we have applied prioritization by WSJF at each level except the team level. We could have applied other business value metrics and prioritized using them but, by using WSJF, at least we can be sure that we have a basis of prioritization by economic value. To ensure that this is the maximum that we could have delivered, we need metrics – the WSJF values are a good start – and monitoring. We also need a culture of lean thinking and continuous improvement.

## Conclusions

In this report, we have introduced some of the main challenges that organizations will face when applying executive-level strategic decision-making to a business value-driven software development group. Primarily, this report has been about a new approach to portfolio management. Lean thinking and scaled Agile techniques, even if the predominate methodology is Waterfall, will help and there are techniques to help with prioritizing work by business value.

## Sources


1. Leffingwell, Dean. Agile Software Requirements:- lean requireements practices for teams, programs, and the enterprise. Pearson Education, Inc. 2011
2. Excerpt from Mikko Parkkola's master's thesis: Product Management and product Owner Role in Large Scale Agile Software Development.
3. Poppendieck, Mary and Tom. Implementing lean software development. Pearson Education Inc. 2007.
4. Reinertsen, Donald. The principles of prodcut development flow: Second generation Lean Product Development. Celeritas Publishing. 2009
5. http://www.rand.org/topics/delphi-method.html
6. http://store.mountaingoatsoftware.com/pages/planning-poker-in-detail
7. http://www.scaledagileframework.com/wsjf/
8. http://www.valuevisualization.com