Agile Patterns

WRITTEN BY **IAN MITCHELL,** CHIEF SCIENTIST AT PROAGILE LTD

CONTENTS

ABOUT AGILE PATTERNS

An agile way of working brings empirical process control to complex and uncertain environments. By releasing value early and often, lessons can be learned through direct experience and risks can be addressed in a timely manner.

It's widely claimed that there are no "best practices" for achieving an agile mode of delivery. Nevertheless, it is possible to identify certain recurring patterns that typify a productive implementation. Whether it be working iteratively, releasing increments either on demand or at a regular cadence, limiting work-in-progress, or having a clear understanding of what "done" means, a common vocabulary can be distilled. In short, there are patterns of thought and practice through which an agile enterprise might be forged.

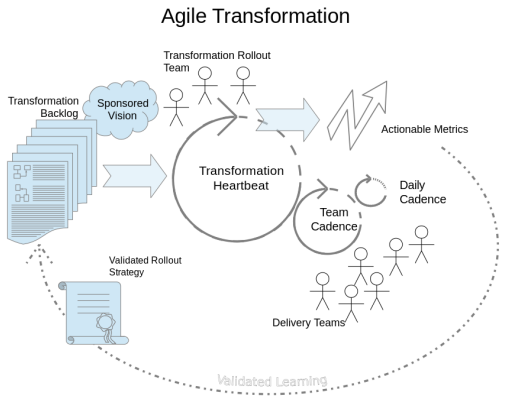
USING AGILE PATTERNS

In many organizations, only executive sponsorship can give sufficient heat to the crucible of change. Agile patterns often differ greatly from established practice, and hence there is usually an organizational gravity that must be overcome. Most companies do not work iteratively, or deliver increments to release-quality standard, or even have the cross-functional teams in place that could master such a process. Without clear sponsorship for deep and pervasive change across the enterprise, an agile way of working can fail to imprint upon organizational culture, and the establishment will revert to convention.

A pattern may be needed for organizational change so that agile patterns can be used. A transformational pattern will reinforce the change process and make success more likely. For example, when agile ways of working are coached, new practices can be ordered and used to shape and assess progress and maturity. The challenge is to ensure that each change introduces the most relevant agile patterns at the most opportune time, and is not implemented for its own sake, but has an empirically demonstrable effect upon outcomes.

A PATTERN FOR USING AGILE PATTERNS

To "SOLVE" the agile transformation problem:



Sponsor change. If organizational gravity is to be overcome, then sponsorship for an agile vision must come from the top. Even getting just one team to produce release-quality work by the end of a time-box can require change at multiple points across the enterprise. There are likely to be many dependencies that will have to be resolved, including upon business, technical authorities, and management itself. Sponsoring a transformation team to coach the use of agile patterns to the right people at the most opportune time is one way forward. The objective must be to engender cross-functional agile delivery teams that can build "done" increments and can learn to inspect and adapt their processes.

Order a transformation backlog. A large enterprise cannot undergo transformation all at once. The process of change will need to be measured and managed. Ordering and prioritizing improvements that leverage useful agile patterns, such as by means of one or more transformation backlogs, can help. Each change should be of deliberately limited scope with clear acceptance criteria, and there should be an understanding of the benefits that are expected to accrue. A change should move the needle on empirical improvement with

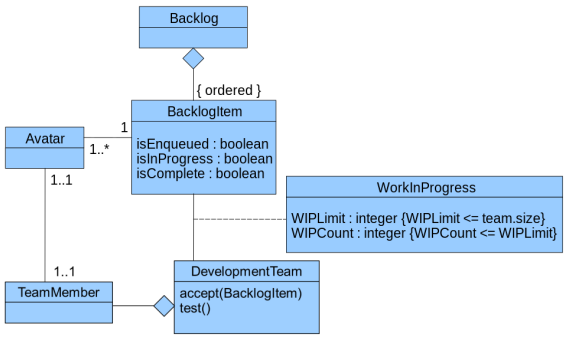
validated outcomes, and not be done for its own sake. The relevant measurements should be actionable metrics. Eventually, agile teams ought to be able to manage their own process of change.

Localize action. The focus of a transformation should always be upon the delivery teams where value is created. Each team should be invested with all of the powers and skills to create release-quality work. A transformation team should actively seek to make itself redundant by fostering that quality of self-organization in teams. The ability to inspect and adapt both product and process locally, and in a timely manner, is instrumental in achieving empirical process control and in reducing waste. Teams may observe their own cadence, including a daily cadence, for this purpose. Sprints and daily Scrums are an example.

Validate improvement. Agile change is not done for its own sake but to improve the delivery of value. For example, work in progress (WIP) may be limited in order to improve focus, quality, and throughput. Limiting WIP to a certain amount may be suitable as an acceptance criterion for such a change, but it must also empirically improve such outcomes. Agile transformation is an exercise in validated learning, where changes are framed and tested in terms of an enhanced innovation capability. These changes should realize the organization's strategic vision.

Enable self-organization. The ultimate aim of an agile transformation team is to make itself redundant.

AVATAR

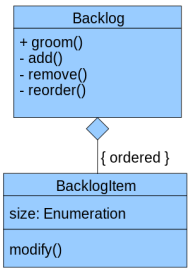


Purpose: Have a signal on an information radar that indicates who is working on what. It can also help to limit work in progress since each developer has only one avatar and can therefore only take off one backlog item at a time before completing it.

Use when: Avatars can be used on both Scrum and Kanban boards to show which team member is progressing each item. Single-piece flow obviates the need for avatars since all developers will, by definition, be working on the same item.

Example: Cartoon characters are a popular choice of avatar. They can be printed out, potentially laminated, and then attached to index cards on a physical Scrum or Kanban board. Fridge magnets can be used on magnetic boards. Electronic boards may not always support avatar images, but most will allow the name of an item's assignee to be registered and displayed instead.

BACKLOG

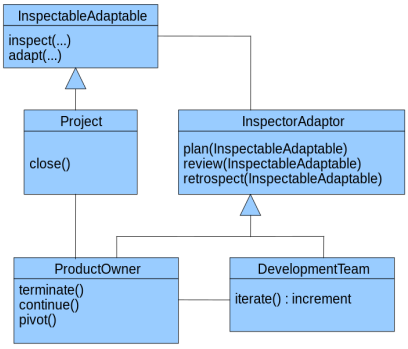


Purpose: Enumerate all work to be done in the form of an ordered list.

Use when: Work must be ordered so that it can be actioned according to urgency, value, or some other criterion. Requirements can originate from multiple sources and stakeholders. As such, priorities are often unclear and the scope of changes may overlap, making estimation difficult. A single repository of ordered requirements is needed. Backlogs and the associated principles of queue management are applicable to all agile methods. A sized and ordered backlog allows estimates to be made regarding the likely time of item completion, assuming that the velocity of delivery is known.

Example: Scrum teams have two backlogs: a product backlog, which contains all product requirements as they are understood at the current time, and a Sprint backlog, which contains the work items needed to complete a selected portion of those requirements. DSDM has a prioritized requirements list, which is analogous to a product backlog. Kanban teams generally only operate using one backlog, although the constituent items may require different qualities of service.

CONTROLLED FAILURE



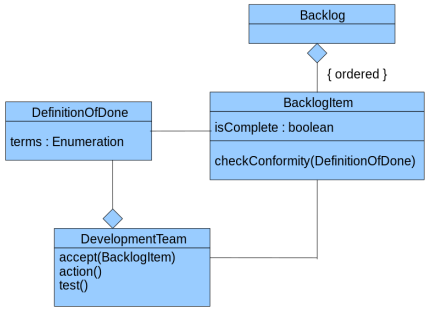
Purpose: Terminate a project once it becomes clear that it is not viable. Accrued value is retained and project resources are freed for other activities.

Use when: Controlled failure should always be an option on an agile project. It may be used when an inspect and adapt opportunity indicates further work will not prove valuable. It is irregular to implement a controlled failure before the current iteration has terminated and an increment is available for evaluation.

Example: It might be decided by a product owner that it would not be profitable to continue with the initiative. A decision of this nature may be arrived at following business advice from product stakeholders or

technical advice from the development team. Reviews, retrospectives, and planning sessions all represent suitable points at which a controlled failure might be initiated. Termination during an iteration, such as when a Sprint goal cannot be met by the team or is no longer relevant to the business, is possible — although it is usually best to wait for an empirically assessable outcome.

DONE

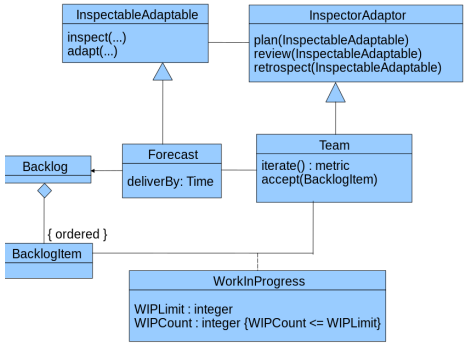


Purpose: Ensure all work is completed to a known standard, so misunderstanding is avoided and rework minimized.

Use when: The done pattern can be used whenever a workflow item is transitioned between states. It is most important to define done at the point where an item is considered to be fit for release. Therefore, there must always be a definition of done that applies to the increment. The pattern can, however, be applied at any station in a workflow.

Example: A consistent understanding of what "done" means is most often encapsulated within a team's "definition of done." In Scrum, this definition may be revised during a Sprint retrospective. "Definition of ready" is another common implementation of this pattern, where conformance indicates that a backlog item is sufficiently well defined to be actioned by a team.

FORECAST

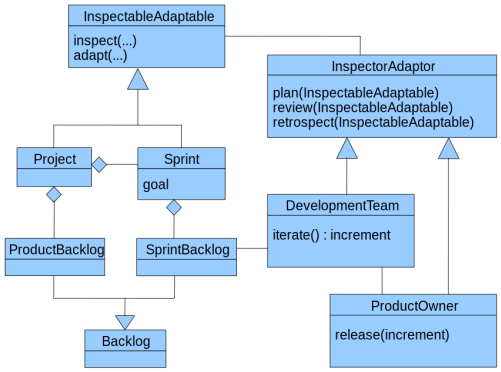


Purpose: Predict completion time based on the estimated size of a backlog and a known velocity.

Use when: Agile teams must be prepared to respond to change and cannot always make a delivery commitment. Yet, even in conditions of extreme uncertainty, it is valuable to have a provisional timeline based on existing data and clear assumptions. Forecasting is used in most agile methods in order to determine the likely deliverables by the end of a given timebox.

Example: Scrum development teams generally limit their forecasts to end-of-Sprint deliverables. The Sprint backlog and Sprint goal effectively represent their forecast and will depend on estimates. Kanban forecasting does not necessarily use estimation at all. For small and repeatable changes, forecasts based on backlog position and throughput are possible.

INCREMENT

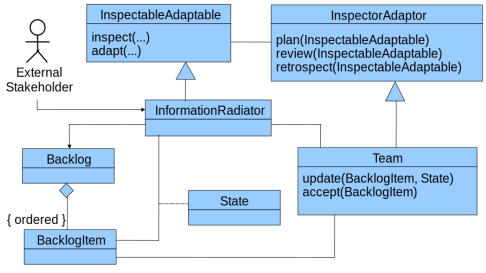


Purpose: Deliver a potentially releasable piece of work early and often.

Use When: Incremental release is appropriate for project work where scope risk and delivery risk are high, and that work can be batched in terms of meaningful intermediate goals.

Example: A development team plans a Sprint backlog with a product owner. The Sprint backlog chosen will be drawn from the product backlog, and in line with an agreed Sprint goal. It will represent those items that the product owner considers most appropriate and timely for a return on investment on the project. The team must also be satisfied that they can in fact deliver an increment of corresponding value by the end of the Sprint (iteration) to the product owner. Once the product owner is in receipt of the increment, he or she may then choose to release it. The value delivered will be reviewed, and the process followed can be retrospectively inspected and adapted to improve value further.

INFORMATION RADIATOR



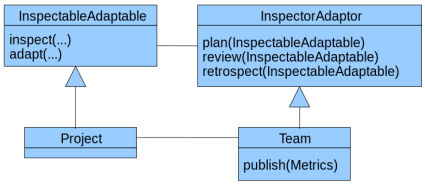
Purpose: Make the status of a team and its work immediately apparent.

Use when: Lack of timely and appropriate information can lead to faulty decision-making and poor collaboration. It is important that a team and their stakeholders have ready access to the status of work in progress, any impediments, any future work that is anticipated, metrics

that allow practices to be inspected and adapted, any forecasts or commitments that have been made, and what responsibilities are being assumed.

Example: As items are accepted and worked on, an information radiator will be updated to show the state of progress. The information radiator can be inspected and adapted by the team in order to make sure that its contents remains current and that it is structured in the most useful way possible. The most common information radiator is a Kanban board or task board. Variants of these can be found in use by most agile teams.

INSPECT AND ADAPT

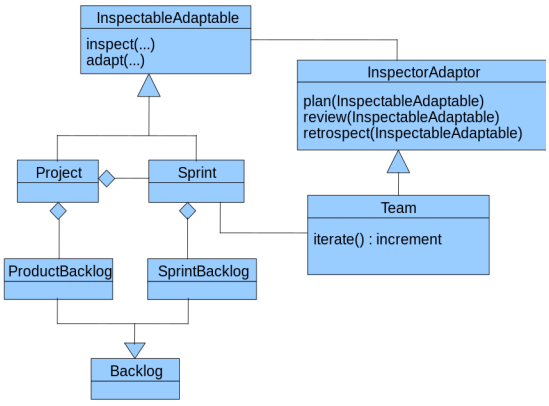


Purpose: Teams delivering value should be able to critique and improve their own working practices.

Use when: A team will inspect and adapt its products and processes at regular points so that the quality of delivery can be improved. Inspection and adaptation must occur as closely as possible to where the work is done. Continual improvement and response to change lie at the heart of agile ways of working. There must be numerous opportunities for this to happen, and they must occur on a predictable and regular basis.

Example: Scrum provides four discrete events at which inspection and adaptation may occur. These are the Sprint review, which looks at what work has been done and remains to be done; the Sprint retrospective, which considers how work is being done; Sprint planning, which determines the Sprint goal; and the daily Scrum or standup, which supports the inspection and adaptation of the team's daily plan for meeting the Sprint goal. Extreme programming does not adopt a comparable discrete approach but rather encourages inspection and adaptation to occur at any point on a development continuum.

ITERATE

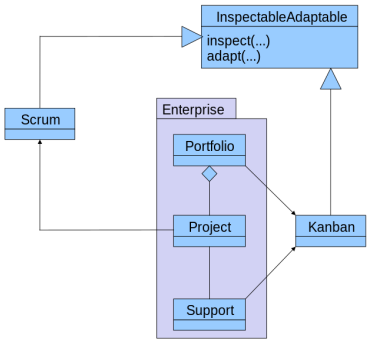


Purpose: Handle small pieces of work at regular intervals.

Use when: Iterative development and delivery are features of any truly agile process. Although it is possible to develop products iteratively and without the incremental release of value, this limits feedback and the ability of a team to inspect and adapt their process in a risk-reductive manner. It also results in the depreciation of product worth. The application of iterative development in agile projects must therefore align with the incremental delivery of value.

Example: The canonical iteration in most agile ways of working is the Sprint. This is a time-box of no longer than one month and always results in a potentially releasable increment. This implementation originated with Scrum, but it has subsequently gained wider currency. Each working day is also an iteration and is commonly demarcated by the inspect-and-adapt opportunity of a daily stand-up. DSDM and XP support iterative feedback at additional levels of granularity.

KANBAN SANDWICH

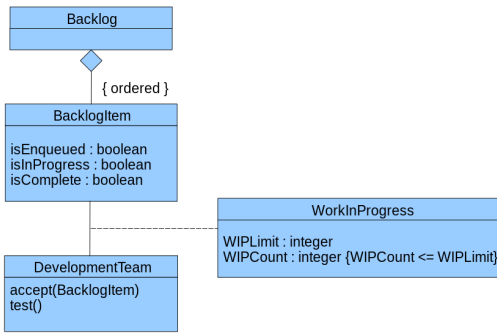


Purpose: Have an appropriate agile way of working at each of three enterprise levels.

Use when: This pattern is applicable across most verticals. It should be noted that technical support teams will need representation at the project level if they are to participate in large scale technology roll-outs.

Example: This pattern can be found in the scaled agile framework. It can also be correlated to the agility at scale approach where such issues have been framed in the context of DevOps and a business-driven project pipeline that supports architectural visioning.

LIMITED WIP

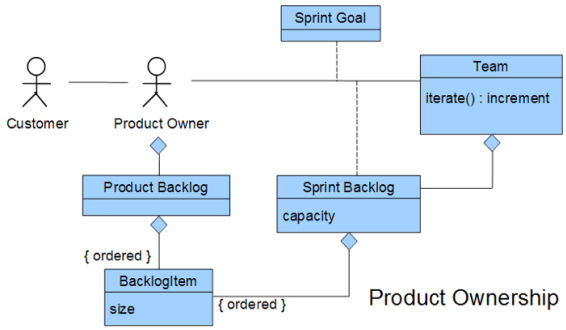


Purpose: Minimize stock-on-hand so as to deliver value more quickly and reduce waste.

Use when: Limited WIP is applicable to all business-as-usual workstreams. It may also be applicable to project work, although the WIP limits can be overridden or changed by an incremental delivery plan.

Example: Limited WIP is a key feature in all Kanban and lean configurations, and it is recognized as a beneficial practice in Scrum ones. It is possible for a Scrum team to plan to action all of a Sprint backlog in one go, and not limit work in progress further. However, this is considered bad practice as it defers acceptance of multiple items to the end of the Sprint, and thereby increases the risk of failing to meet the Sprint goal.

PRODUCT OWNERSHIP

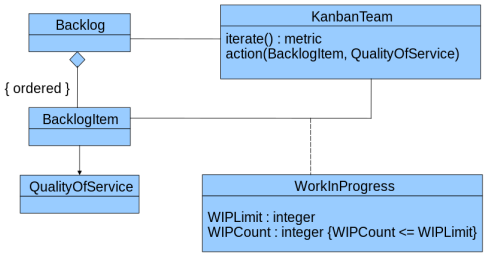


Purpose: Provide a single business liaison who is able to represent customer needs in an accountable manner, able to provide effective pull on a team backlog, and who is empowered to make business decisions.

Use when: Product ownership is key to a collaborative way of working, and as such, it is applicable to all agile methods.

Example: Scrum expressly supports and requires a product owner role. In XP, product ownership is represented by the onsite customer role. In DSDM, the operational duties of a product owner to a development team are fulfilled by a business ambassador. A DSDM business ambassador may proxy for other business roles; these can include a business visionary and one or more business advisors.

QUALITY OF SERVICE

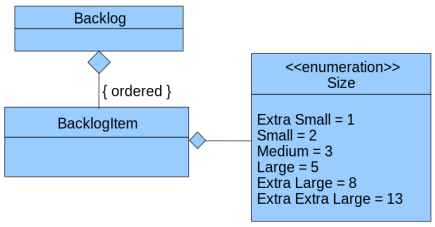


Purpose: Vary the way backlog items are expedited and handled.

Use when: Quality of service is applicable to situations where the handling of work varies by context.

Example: Examples include service-level agreements where the demands of certain clients may be prioritized over others; variations in the skills or technologies required for the implementation of a backlog item; or the prioritization of emergency work over that which is in progress. Quality of service is allowed to vary in most Lean-Kanban implementations, and in hybrid Scrum-Kanban implementations that allow the quality of service to vary (such as by supporting a "fast track" or "expedite" class of service).

RELATIVE SIZING

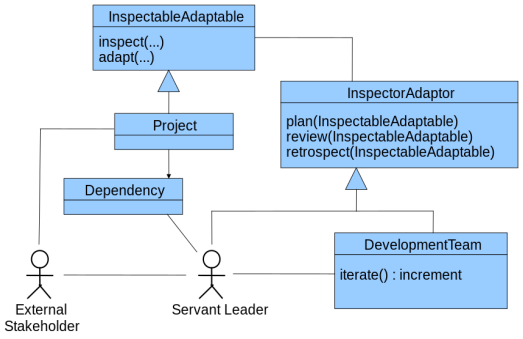


Purpose: Allow a backlog to be ordered when it is difficult for team members to estimate the size of each item in a backlog.

Use when: Relative sizing is useful when teams are unable or unwilling to create time-based estimates. Time-based estimates are unreliable and are apt to become even more so at larger magnitudes.

Example: T-shirt sizing uses extra small, small, medium, large, extra large, and extra extra large as relative sizes. Items may be written as index cards and sorted by the team by placing each one below the most appropriate sized heading. Estimation poker uses playing cards with sizes on a near-Fibonacci sequence (e.g. 1, 2, 3, 5, 8, 13). Cards are played by each team member when jointly estimating an item; the cards are turned over to expose their values and any significant discrepancies can be reviewed by the group and challenged.

SERVANT LEADERSHIP

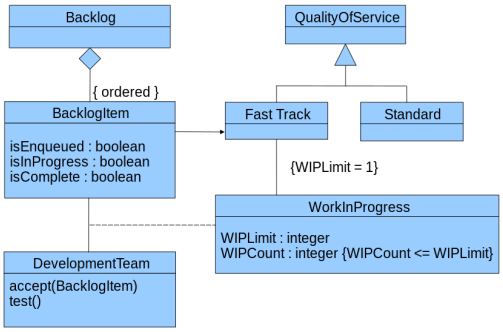


Purpose: Protect a team from distraction and impediment — and facilitate their progression towards their goals. A servant leader helps a team inspect and adapt the iterative process they follow. The leader resolves any dependencies that the team may encounter so that they do not become impediments that would compromise the incremental release of value. The leader also represents and advocates the team's position to external stakeholders so that the team are not approached directly and remain free to self-organize around their goal.

Use when: Servant leadership is applicable to all agile methods.

Example: The term "Scrum master" is commonly used to describe this role, and is no longer restricted to Scrum alone. A coach in extreme programming has a similar remit, as does the team leader in DSDM. It is common for all of these to be referred to as Scrum masters.

SWARM

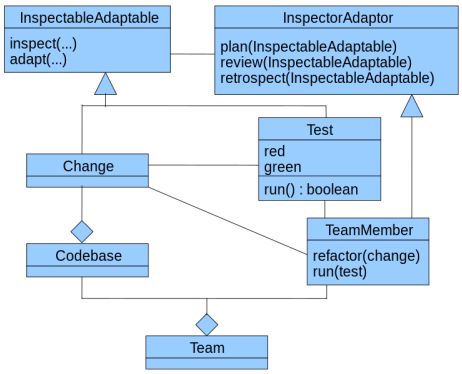


Purpose: Have all available resources work on one thing in order to expedite it as quickly as possible.

Use when: Swarming should be considered in situations where an urgent requirement demands the attention of the whole development team, and where work on other requirements can be suspended. Note that for swarming to be viable, it must be possible for multiple developers to work on one item simultaneously and without causing each other impediment.

Example: Swarming can be implemented for Lean Kanban teams of small size, where developers will not cause each other impediment by working on the same item.

TEST-DRIVEN DEVELOPMENT



Purpose: First, prove that an unmet need has been met without breaking anything else. Then, optimize the solution.

Use when: Test-driven development is applicable to all forms of development. It is also applicable for maintenance and repair.

Example: A team member creates a test for a change that has not yet been made. The test is run. If it passes (green), then this indicates that the change would be unnecessary, as the desired conditions are already satisfied. If it fails (red), then this proves that the conditions are not yet met. The change can then be made and the test can be run again. If the change has been implemented successfully, then the test will pass (green). The code is refactored to improve design, performance, or other qualities that may have been impacted by the change. The test is re-run and if this refactoring has not compromised the desired behavior, then the test will still pass.

FURTHER AGILE PATTERNS

There are many other patterns that can be found across agile practices. These include (but are not limited to):

Pivot: A structured course correction that allows a new hypothesis involving different choices to be tested.

Co-location: The placing of team members in physical proximity to each other so that the best communication and collaborative potential can be leveraged.

Refinement: The clarifying and ordering of items on a backlog so that they are ready to be worked on.

Scrum of Scrums: A collaborative activity between representatives of multiple agile teams in order to focus and replan around a shared purpose.

Championship: The accountability of a particular agile team member toward providing a needed skill or capability.

Scrumban: The gradual replacement of Scrum Sprints with a more Kanban-like system of continuous flow.

Single-piece flow: A special case of limited WIP intended to maximize throughput, such that a team only has one item of work in progress at any given time.

Time-boxing: The discipline of ensuring that an event, such as an inspect-adapt meeting or an iteration, does not exceed a maximum period of time.

Trade: The moving of one or more work items out-of-scope so they can be replaced by more valuable work of similar size.

Management by exception: The delegation of authority to act within specified tolerances, only involving others if those tolerances are exceeded.

Peer: A team member who can support team responsibilities and accountabilities.

Teamwork: Getting people to collaborate on work being actioned so that increments are developed as efficiently as possible.

Release orchestration: Ensuring that delivery teams contributing to the release of a product or service are aligned to do so, and that no opportunity to address market demand is lost.

Epic: The grouping of certain items on a backlog in terms of a high-value feature.

Metrics: Provide actionable, non-subjective data so that informed decisions can be made.

Minimum viable product: Deliver a small portion of value in order to provide an early return on investment, and/or to allow lessons to be learned as quickly as possible.

Proxy product ownership: Authorizing a suitable stand-in should a product owner be indisposed.

Value stream: Understanding the value being added at each stage of a process with a view to eliminating waste.

Sponsorship: The sense of urgency conveyed by management in support of an initiative, and which must be sufficient to overcome organizational gravity.

DEALING WITH ANTIPATTERNS

Agile antipatterns are organizational impediments to agile practice that should be challenged and removed. They are best tackled by means of the patterns that are most likely to eliminate them.

Cherry-picking: Team members are often tempted to select items from a backlog that they expect to be the most gratifying for them or the easiest to do. Swarming on items may help, as it requires team members to collaborate more closely.

Cloned avatar: If development team members are given unclear priorities in a high-pressure environment, they can be tempted to start development on an item without first completing the work they have in progress. By duplicating their avatar, they can then claim to those exerting the pressure that work is underway. Limiting work in progress is likely to be beneficial, as a pull-based system ought to be encouraged.

Death march: It can be politically difficult to cancel a struggling initiative if it has already absorbed significant resources. To do so may imply that the project has been a poor investment and that the time and money committed has been lost. Stakeholders would prefer to continue with a failing project and hope for sudden and improbable change. Controlled failure must be recognized as an option.

Disguised project: Some changes to IT systems can be trivial in nature, such as minor amendments to site content or defect fixes. This type of work is considered to be "business as usual" (BAU) and as such, it is often absorbed by the organization-at-large as an operational expense. Departmental stakeholders who want more substantial changes must usually resource a suitable project from their capital budgets. They therefore have an incentive to disguise such work, either by misrepresenting it as a normal operational small change or by breaking it up into a series of small changes that they hope will slip through a BAU work-stream unnoticed. The Kanban sandwich and value stream patterns might provide remedy.

Distributed team: Physical constraints, such as desk space and the wider geography of an organization, may inhibit the co-location of team members. The need for agile team members to work in proximity to each other can present logistical issues that managers are unwilling to overcome. As such, a logical team boundary will be made to span physical boundaries. Managers can thus avoid an immediate resourcing issue while offloading the management of any risk incurred to the teams themselves. Co-location is generally advantageous and can reinforce the teamwork pattern.

Management by reporting: Highlight issues and concerns while deferring the risk of taking action to others — at least to a later date. The hope is to avoid personal exposure should there be negative consequences from taking action. Management by exception can be a more practical alternative in so far as it establishes clear tolerances.

Micromanagement: Managers can find it hard to delegate operational responsibilities. There are a number of possible motives for a manager wishing to retain control. For example, the manager may not trust others to perform the duties satisfactorily, or he or she may simply enjoy dealing with operational matters. A servant leader will coach more constructive behaviors, and the team should be encouraged to inspect and adapt its own way of working. Again, establishing clear tolerances through management by exception can help.

Sprint zero: Set up an agile project, while contextualizing unplanned initialization overheads in agile terms. No value is released, and empirical process control is not established. Ensuring that the iteration and increment patterns are used together can be wise.

Time theft: Agile teams draw their work from prioritized backlogs, which means that those waiting at the bottom of the queue may expect some degree of delay. Such parties may thus be incentivized to circumvent the backlog management process by approaching team members directly for assistance. Parties seeking assistance in matters that lie beyond the team's remit can have an additional incentive, given that such activities would not be appropriate for inclusion on the team's backlog in the first place. A good servant leader, such as a Scrum master, will protect a team from such interference.

Too busy: Key stakeholders often have responsibilities that span multiple teams (e.g. product owners, senior designers, and architects). If they do not value a team's product or service particularly highly, they may be tempted to abdicate or defer their stakeholder duties in order to give other matters their attention. Sponsorship for the initiative is essential. Controlled failure may be the best option in severe cases.

Unbounded team: The allocation of team members to certain workstreams implies that they will not be available for others. This is a constraint on organizational behavior since it means that managers cannot assign people to multiple duties in a reactive or ad hoc manner. Organizations can be tempted to compromise on such discipline for the sake of expediency or in support of "firefighting." Teams should be able to inspect and adapt their own membership and to frame their own teamwork commitments.

Unbounded timebox: Allow an indeterminate amount of time for the completion of a task. Teams can be tempted to try to extend an available time-box if doing so increases the chances of an intermediate goal being met. The team can then create the illusion of success even though the deliveries that were forecast have not been made. Good time-boxing will avoid this by ensuring that a maximum time limit is observed.

Uncommitment: Team members who do not value the product or Sprint goal, or who are allowed to accept other priorities, may be tempted to admit unplanned work into a Sprint. This can include work from forceful parties which the product owner does not value. Strong product ownership is important if this problem is to be overcome.

Undefined done: Progress work as quickly as possible and without sufficient regard for quality. Technical debt is likely to result. Observing a definition of done that assures each increment as being of release quality is essential.

Unlimited WIP: When faced with competing and vociferous stakeholder demands, team members can feel obliged to action multiple items simultaneously. In an attempt to pacify stakeholders, they can therefore claim that an item is being worked on and is no longer enqueued. Both good product ownership and a team determination to limit WIP appropriately are needed to overcome this.

Unresolved proxy: Multiple stakeholders can have an interest in a product and each may only be able to articulate the requirements in their area. Consequently, a senior product owner may be tempted to delegate ownership to multiple proxies. Good product ownership ensures that one clear authority is accountable for value.

Vanity metrics: Use only the most favorable metrics in order to strike a posture. Team members can be tempted to show their efforts in the best light rather than in a more objective one. Good metrics are based on data which supports inspection and adaptation and the taking of informed, constructive action.

Waterscrumfall: Organizations may often have the goal of adopting an agile way of working, but they can lack the cultural grit to transition away from an established stage-gated culture. They then attempt to affect a compromise in which an iterative development approach is encapsulated within a development stage. In doing so, they hope to leverage the benefits of agile practice while not actually changing the organization's delivery approach or the terms of reference that are comfortable to stakeholders. The transformation pattern helps challenge this thinking, as it puts empirical process control at the heart of an enterprise change attempt.

Written by Ian Mitchell

Ian Mitchell's experience in iterative and incremental delivery began with a PhD in object-oriented rapid prototyping in 1997. He then worked as a developer for several years where he witnessed the failure of many initiatives that were allegedly implementing Scrum. He realized that his true vocation was to help teams and organizations to better understand agile practice. Dr. Mitchell has a particular focus on evidence-based enterprise transformation and is the curator of a related site, agilepatterns.org. A Scrum advocate and accredited Professional Scrum Trainer, he is a veteran member of the scrum.org forums.

DZone, Inc.
150 Preston Executive Dr. Cary, NC 27513
888.678.0399 919.678.0300