AGILE TRANSFORMATION STEP-BY-STEP

Ian Mitchell
agilepatterns.org

THE TRANSFORMATIONAL MECHANISM

"The only way you survive is you continuously transform into something else. It's this idea of continuous transformation that makes you an innovation company."

- *Ginni Rometty*

Introduction: The mastering of innovation at scale

Reduced to its barest essence, enterprise agility is really about mastering innovation at scale. Most organizations are conservative rather than innovative, and their respective establishments can be quite reactionary when faced with change. A strategy must therefore be identified, sanctioned, and put in place to achieve enterprise transformation. The strategy must leverage the innovation potential that lies within the organization so that disruption is managed productively, and not feared and retreated from, as though it were a threat. All of the varied parts of an enterprise, which when brought together generate value, have a role to play in the innovation process. In other words the organization must become a disruptive innovation network, an integrated enterprise which can identify, shape, and harvest new opportunities in a systemic and collaborative fashion. To put it more succinctly, it must become a "startup at scale". If an organization can master this agility, then it will be able to perform as nimbly as any small startup, whilst exploiting the resources and scaled benefits that only a larger and shared venture can bring.

Now, if an agile transformation is to succeed, the associated patterns and practices must be implemented in a managed fashion. We need to get our ducks in a row. There are many things that need doing, many

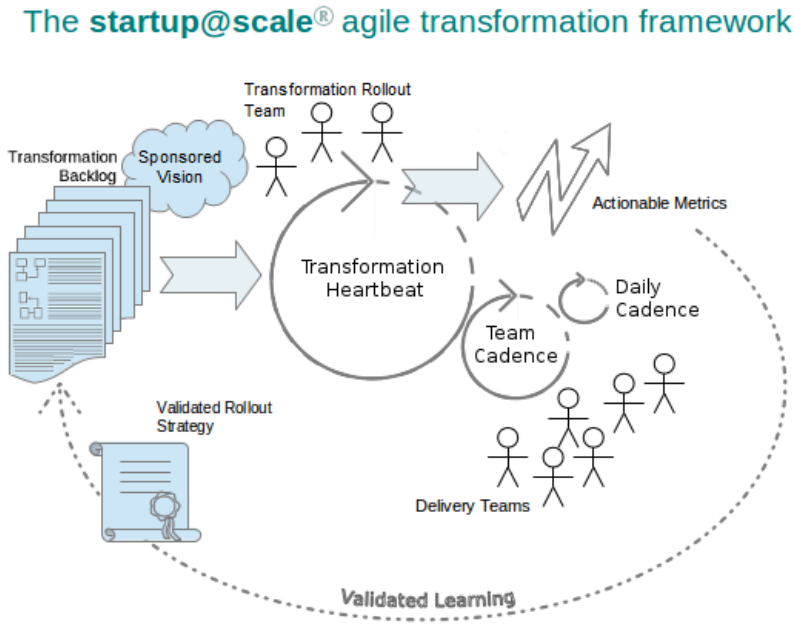
workflows that will need improving and new teams that must be formed, many new skills to be learned and old assumptions challenged, and many forms of waste that will need to be found and eliminated. It's no use setting about this change in a haphazard way, trying to do everything at once in the hope that some of it will stick. This is where a transformation framework comes in. A framework will help us to manage innovation at all organizational levels, from portfolio management to technical implementation and support, and across multiple value streams. We will seek empirical evidence of improvement, not the submission of reports or promissory notes. Innovation accepts no substitute for actual delivery. The framework must therefore help us to focus on the coaching and empowerment of agile delivery teams.

Agile transformation must be measured and managed. At each point of application, the transformational effort ought to be captured as a backlog of changes, each of which is the useful application of one or more agile "patterns", or good practices. These transformation backlogs will need to be ordered and managed by a sponsored and qualified team: change agents who can roll out transformational change until the process is normed, and a self-adapting and self-sustaining innovation network is in place.

In identifying each backlog item, this transformational Rollout Team will select the *right patterns* at the *right time* for application in the *right organizational areas*. As the organization becomes more focused on delivery, the respective teams become more agile, and more able to inspect and adapt their own process. Each team will be able to populate its own backlog of planned improvements, to identify new opportunities and innovative solutions, to highlight impediments, and to provide transparency to the wider organization over quality and progress. The success of each change will be evaluated empirically by means of clear acceptance criteria and iterative inspection. These validated lessons help to reorganize and reprioritize each transformation backlog. A measured

and managed enterprise transformation, like the agile delivery of a product, is sustainable, transparent, and incremental in nature.

The need for ongoing delivery must be recognized and supported. Organizations do not typically come to a halt in order to change; instead, transformation happens while those organizations are in flight. A well structured transformation backlog is instrumental to managing this. Change must be ordered so that risk is controlled, and the ongoing viability of the business is not compromised.

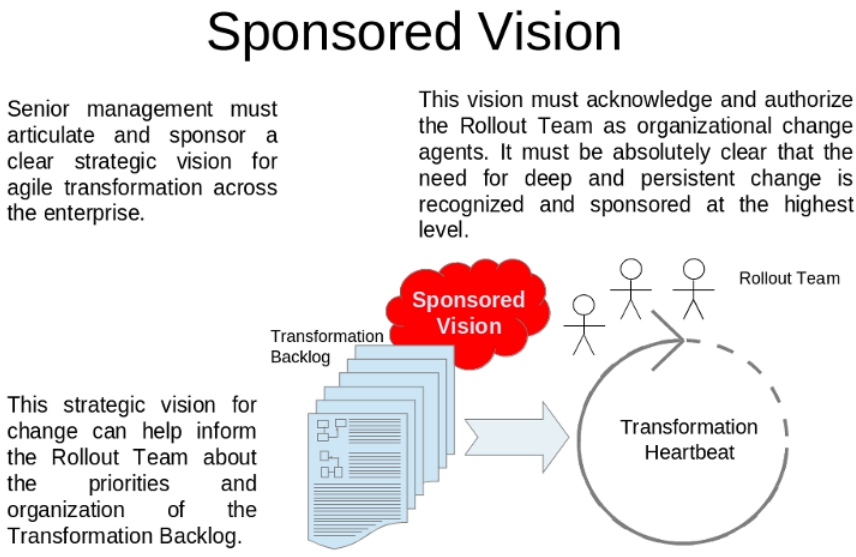


Here is a framework, a structured remedy by means of which enterprise transformation can be enabled. It is captured as a pattern in itself - a pattern for agile transformation. It consists of 12 components: Sponsored Vision; Transformation Rollout Team; Delivery Teams; Transformation Backlog; Transformation Heartbeat; Team Cadence; Change Actioning; Daily Cadence; Verification; Actionable Metrics; Validated Learning, and a Validated Rollout Strategy. Let's now look at transformation in those

terms, and at the implementation of each component as a step which might reasonably be followed.

Step 1: Begin with a Sponsored Vision

Enterprise transformation requires clear executive sponsorship. If top-level support for the initiative is not made clear at every level of the organization, then only localized or tactical improvements are likely to be made.



Make no mistake about it – enterprise agile adoption is a strategic issue. It requires organization-wide, deep and persistent change. The support for this change must be driven by senior management. It has to be sufficiently strong to overcome “organizational gravity”...the tendency of people to revert to traditional ways of working. Agile transformation is hard. It means learning new ways of working and new ways of thinking, and unlearning old cultural norms that masquerade as common sense.

Senior management has three key responsibilities to fulfil in this regards:

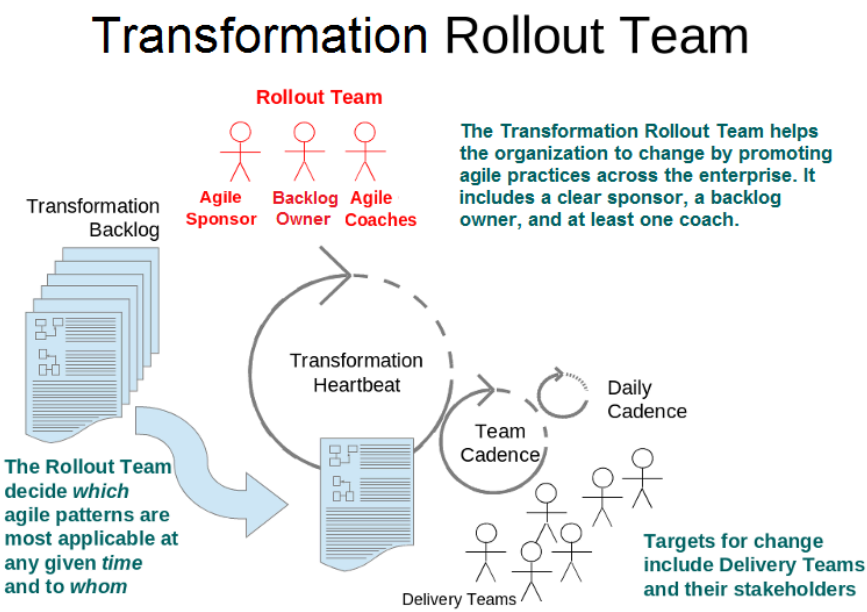
1. Craft a vision for agile transformation and share it with the whole organization. The expectation must be set that agile change requires the participation of each and every person who contributes in some way to the delivery of value. Note that potential trigger words such as “agile” and “innovation” do not necessarily have to be used when describing or referring to the vision. Avoiding specific terms can reduce prejudice against the change initiative and inappropriate early solutioning, and help to keep implementation options open.
2. Create and empower a Rollout Team to implement the vision. This team must include suitably qualified Agile Coaches who can guide and mentor stakeholders through the process of transformation. The team must also include a clear Agile Sponsor who can represent senior management and the stake it has in enterprise agile adoption. The sponsor must be able to articulate the need for wide and deep change to existing organizational practices, and navigate the political landscape on behalf of coaches so change is facilitated. The coaches themselves cannot be expected to deal with organizational politics and reactionary elements. A key responsibility of the Agile Sponsor is therefore to eliminate any doubt concerning the Rollout Team's remit and the authority of its members as organizational change agents.
3. Become change participants. Enterprise agile transformation requires change on the part of senior managers too. There are important lessons to learn about the flow of value in the organization, quality assurance, the management of batch sizes, and servant leadership.

Now, we must be clear on the following important point. Although executive sponsorship is a critical success factor in enterprise transformation, agile change is not something that can be imposed. The duty of sponsors is to encourage the adoption of good practices by responsible and self-organizing teams. Also, enterprise agile

transformation should not be regarded as a separate initiative, but rather as a means of adding value to the organization and its ongoing business.

Step 2: Assemble a Transformation Rollout Team

Agile transformation only makes sense if current ways of working are unsatisfactory and innovation is not well managed. For example, an organization's operating model may no longer be fit for the market, or opportunities may be missed unless such a transition happens. However, the successful adoption of agile practices across the enterprise has two other essential pre-requisites beyond a pain-point. Firstly there must be organizational willpower for change, including clear executive sponsorship for an agile transformation program. Secondly there must be a critical mass of skilled people who are able to implement this strategy, all of whom must be proficient in the core competencies of actually rolling out agile change, and who understand the importance of building largely self-adapting and self-directing teams.



Both of these pre-requisites are met by the Transformation Rollout Team. This is the guiding body for enterprise agile transformation, the one which takes the first and reinforcing steps, and which coaches, trains, mentors and leads the rest of the enterprise in agile adoption. This team decides which changes need to happen and when, the patterns that will be involved, and how those changes should be ordered. It determines where change should be targeted at any given time, and what success criteria will apply. A strategic team with an enterprise-wide remit may identify and empower other teams within the organization as change agents.

A successful change program is determined by the following relationship*:

D x V x F > R

where

D is the Dissatisfaction with the Status Quo;

V is a clear, compelling, believable Vision;

F is the First and reinforcing steps;

R is resistance to the change.

Any Transformation Rollout Team, regardless of the level and context in which it operates, will have three clear roles within it: an Agile Sponsor, a Transformation Backlog Owner, and at least one Agile Coach.

Agile Sponsor

The Agile Sponsor represents the executive authority for agile adoption. It is the Agile Sponsor's responsibility to ensure that enterprise-wide support for the transformation program is forthcoming. The Agile Sponsor makes executive authority clear and challenges any perceived or claimed exceptions. He or she removes any doubt among stakeholders that deep and pervasive change has the full support of the organizational

executive. The Agile Sponsor must have the authority to fulfil these duties, and the social and political skills to ensure that they are evenly applied.

As the representative of executive authority, the Agile Sponsor is ultimately responsible for the success of the transformation program.

Transformation Backlog Owner

Someone must have the final say over the content of the Transformation Backlog and its ordering. The owner of that backlog needs to have a clear understanding of the appetite for change amongst organizational stakeholders, including which of them are most likely, at any given time, to benefit from new agile working patterns and practices. The owner does not need to be an expert in agile transformation, or in the patterns and best practices that are behind each change on the backlog. The Transformation Backlog Owner will be informed and advised by Agile Coaches in such matters, and will work with them to ensure that an appropriate Transformation Backlog is crafted and implemented.

The Transformation Backlog Owner may in fact be the same person as the Agile Sponsor. There is no need to identify separate people to fulfil these two roles. The vesting of these authorities in one person can be advantageous, although it is not always practical.

Agile Coach

An Agile Coach is an expert in the application of agile methods, including the associated patterns and best practices. He or she understands the business of facilitating organizational change, and will help stakeholders at an operational level to adopt agile ways of working. An Agile Coach demonstrates servant leadership. He or she will lead by example if needed, and will make increasingly agile teams less dependent upon their assistance. The Agile Coach may be a Scrum Master for one or more of the teams being coached.

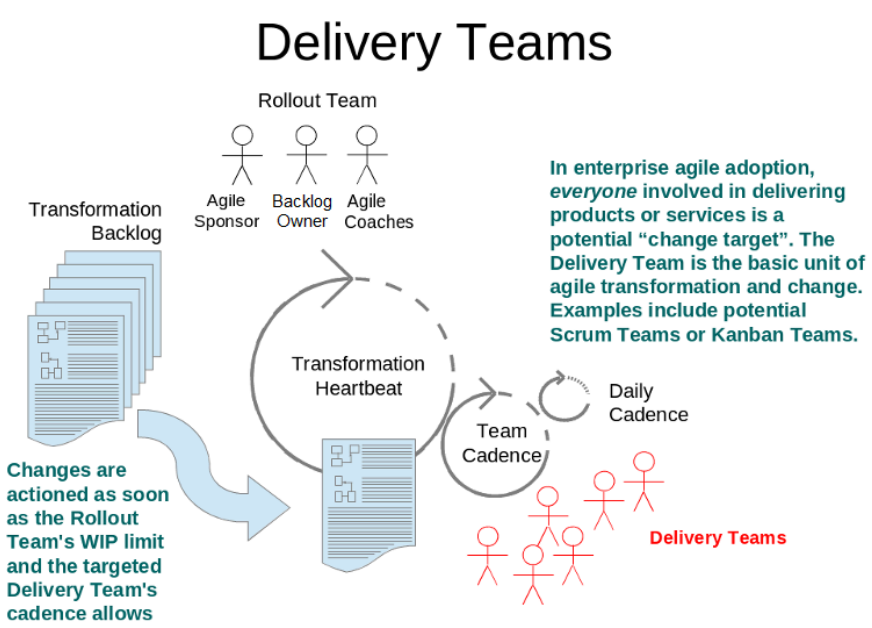
A Transformation Rollout Team can have multiple Agile Coaches in order to facilitate change. When actioning an item from the Transformation Backlog, the Team will self-organize to decide which coach or coaches should apply each change to each target for change.

As a servant leader, an Agile Coach will remove impediments to agile adoption. A coach must be able to rely on the support of the Agile Sponsor, and the collaboration of the Transformation Backlog Owner, for coaching the change expected.

Step 3: Identify your Delivery Teams

Each change drawn from a Transformation Backlog should be applied to a narrowly targeted part of the organization. A good agile pattern or practice can support many change items in the backlog, and each change item can implement several good patterns and practices. Furthermore, the same pattern or practice may be applied, at different times, to different organizational areas. For example, the orchestration of releases might be coached to certain groups prior to others. By limiting the scope of change, regardless of how many good practices are being leveraged, to carefully selected organizational elements we can transform the enterprise in a managed and controlled fashion.

It is important to think of these organizational elements as “Delivery Teams”. Each part of an organization that is undergoing agile transition should make a contribution to the delivery of value in some way. As such, Delivery Teams might exist at operational, project, program, or portfolio levels. In fact any stakeholder who is responsible for planning or facilitating the delivery of a product or service should be regarded as a member of a Delivery Team. These teams can include people we might typically think of as “managers” as well as analysts, developers and engineers. We therefore need to include those with strategic management responsibilities – such as deciding what projects or programs to deliver – as well teams with entirely technical remits.



How do we transform a "change target" into an actual Delivery Team, which can become self-managing? The answer lies in the establishment of cadence and the empirical adoption of changes that appeal to good agile patterns and practices. It is cadence that allows stakeholders to systemically participate in agile improvement. By improving their delivery of value to a regular beat, and by successfully applying the associated patterns of change, stakeholders become first-class citizens in their own agile transformation. Once cadence is established, we can then begin to refer to these participants, seriously and credibly, as "Delivery Teams".

In an agile way of working, Delivery Teams are encouraged to be self-organizing, and so their membership may initially be fluid before they stabilize. Each team should be able to complete its work in a timely fashion, and to a known and acceptable level of quality. This implies incorporating the required skills and resources into the team.

Examples of agile Delivery Teams include:

- Scrum Teams, often with a focus on project work
- Kanban Teams, often with a focus on “business as usual” work
- Scrumban Teams, exhibiting hybrid Scrum-Kanban characteristics
- DevOps Teams, with responsibilities that span product development and operational support.

Step 4: Draft and refine a Transformation Backlog

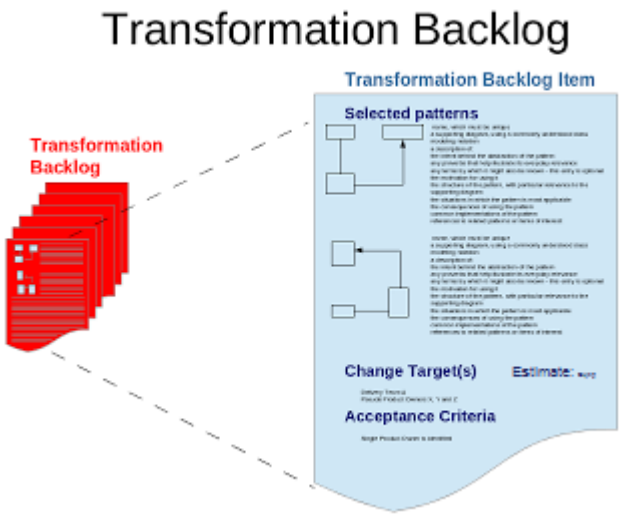
Executive sponsorship is an essential pre-requisite for agile adoption at enterprise scale. Only narrow tactical improvements – those which affect individual teams rather than the whole organization - are likely without the clear and unambiguous support of senior management.

This implies that enterprise transformation must be considered at a strategic level. Although a strategic vision for organization-wide agile practice is important, it is not enough. There must also be a clear means for implementing the vision if agile transformation is to succeed in practice.

The strategic vision for agile change is implemented using a Transformation Backlog. This is an ordered list of changes. Each change item on a Transformation Backlog ought to be structured according to an "Action Plan" for its implementation. A suitable plan can be structured in three parts as follows.

Selected patterns

Remember that change isn't done just for the heck of it. There must be a rationale, and in an agile transformation each change that is brought into effect should be expected to improve innovative potential in some way. Each item on the Transformation Backlog will therefore implement one or more agile patterns or practices that have been selected by a sponsored Rollout Team. Antipatterns may be included if they represent organizational impediments (dysfunctions) for removal. Antipatterns are best tackled if they are treated as epics, and subsequently broken down into the good patterns and practices which will eliminate them.



Target beneficiaries

Each Transformation Backlog Item must be applied to specific organizational components in support of a measured and controlled program of sponsored agile adoption. Coaching, training, and mentoring from the Rollout Team will facilitate this, as can support from peers. By the successful application of a change, each "target" for that change will demonstrate, empirically, the improved agile behaviors which the associated patterns and practices represent. Their ability to manage their own agile way of working will improve and, as they mature, less assistance will be needed and the Transformation Team can prioritize change for other areas of the enterprise. Any pattern or practice may be repeated across multiple items in the Transformation Backlog, as the same pattern may need to be applied in different contexts and at different times.

Note that the use of the word “target” can be misinterpreted or misrepresented as being confrontational. It may be necessary in some transformation initiatives to replace this with an alternative euphemism, such as “change beneficiaries”.

Now, although a Transformation Backlog Item can leverage multiple agile patterns and good practices, its application should be deliberately limited in scope. This is another way of saying that it should target as small an area as possible, and that at any one time the transformational Work In Progress should be correspondingly limited. It is better to clearly and unambiguously facilitate small improvements than to bite off more than the people who are affected can chew. To this end, the Action Plan should include an estimate which indicates the likely effort and complexity of implementing the change. A substantial change may preclude bringing other changes into progress before it is completed and retired.

Acceptance Criteria

Each Transformation Backlog Item must have clear Acceptance Criteria. These are the evidences that prove the change has been applied successfully. They are often expressed in terms of roles, demonstrable behaviors, events and their outputs, qualities of specific artifacts, and (ideally non-lagging) indicators and metrics. Identifying good Acceptance Criteria is hard and it is arguably the most challenging of transformation skills that one can develop. Understanding the intent, structure, motivation, applicability, consequences, and implementation of each pattern is instrumental to this ability. It requires the joint input of the sponsor, the backlog owner, and the coaches to frame acceptance criteria successfully, although as delivery teams mature they will increasingly demonstrate this capability themselves.

Action Plan Refinement

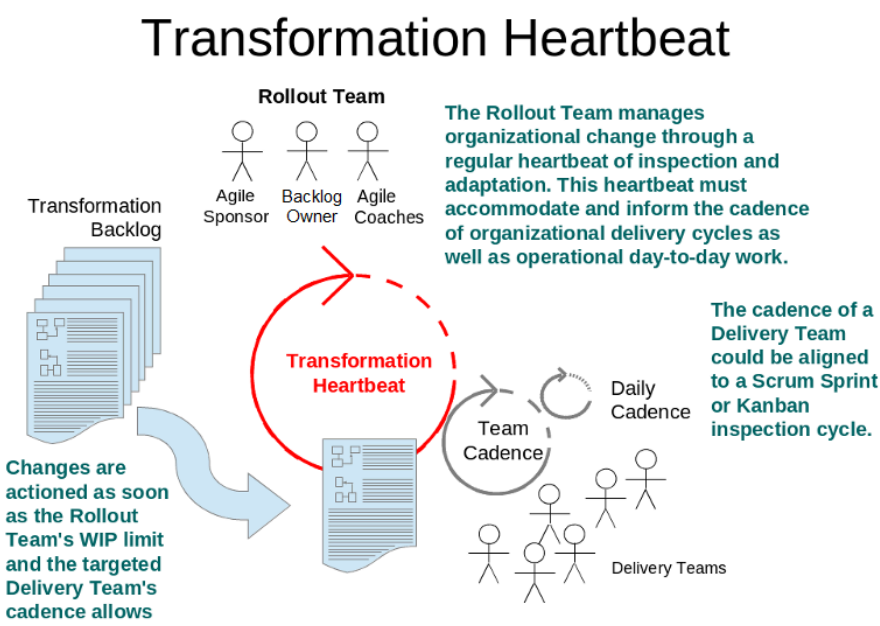
Action Plan Refinement is an ongoing activity during which the “Action Plan” is created and subsequently improved for each item in turn. The item's Action Plan describes what needs to be done in order to effect a change so that appropriate Acceptance Criteria can be met. . Remember that the greater part of this is likely to involve coaching and mentoring, and these can be expected to feature heavily in the plan. The plan for each change item will include a relative estimate of the effort required

for its completion. At any one time, the Transformation Rollout Team should action only as many items as they expect can be handled concurrently. Work In Progress must be deliberately limited.

Action Planning should be completed on a Just In Time basis for each Transformation Backlog Item, i.e. for each change, as and when its implementation becomes imminent. Before then, the item's plan may exist only in sketch. The highest priority items should therefore have mature plans that are sufficiently detailed for their respective patterns to be achieved, whereas lower priority items might be planned only at a cursory level with broad estimates that allow the backlog to be roughly sized. The overplanning of low priority items is potentially wasteful and must be avoided. As long as the items bring a sense of scale to the work remaining to be done, and allow it to be ordered, then that is sufficient for the purposes of refinement. This is because the value of each Action Plan can be expected to decay over time. Precociously specified details may no longer be appropriate once the item becomes a priority for implementation. It is also possible that the proposed change may be obviated by events and removed from the backlog altogether.

Step 5: Establish a cadence for transformation

A cadence is a regular, timeboxed cycle of no longer than a month. Each cycle provides an opportunity to assess productivity, and to inspect and adapt working methods in order to improve them. The Transformation Rollout Team facilitates this by drawing changes from the Transformation Backlog and coaching their adoption. Evidence is gathered timebox-by-timebox and the effect of the changes being effected can be gauged. Once a group values cadence with respect to delivery duties, we can begin to refer to it as a Delivery Team or at least as part of one. Delivery Teams can exist at project, program, and portfolio level. They may include teams with strategic management responsibilities as well as those that are more technically or operationally focused.



Each part of the enterprise that is to be involved in an agile transformation should make a contribution to product or service delivery in some way, and which helps in the release of value. If it does not make such a contribution, then there is little need to factor it into the agile adoption effort at all. Each target for change is therefore expected to contribute to delivery, and if it is to do so successfully then it must observe cadence, just as the transformation itself must observe cadence.

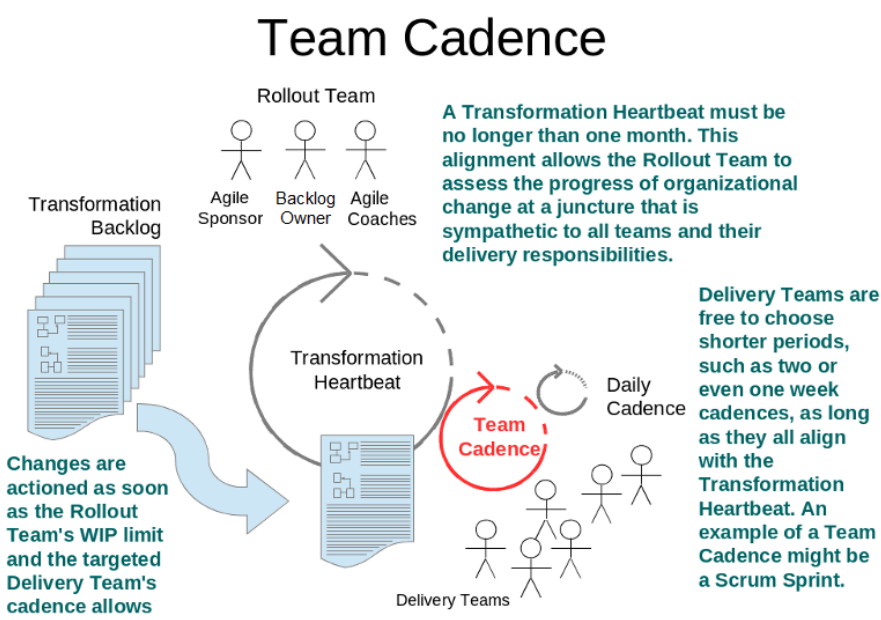
It is very important to consider the matter of cadence when rolling out an agile transformation. For one thing, we need to ensure that a suitable cadence is being observed by each Delivery Team. We must then verify that agile change is happening: that batches of work are kept small and manageable, that inspection and adaptation does indeed occur, and that

each team's deliverables are integrated where necessary and without delaying the release of value.

A Transformation Heartbeat should therefore be established and used for reference across the organization. It should be no longer than one month. Delivery Teams are free to choose shorter periods for their own cadences, such as two or even one week Sprints, as long as they all align with the Transformation Heartbeat. This organizational alignment allows the Transformation Rollout Team to assess the progress of enterprise change at a juncture that is sympathetic to all teams and their delivery responsibilities.

Step 6: Establish a team cadence for delivery

Agile development and delivery should be performed with respect to a certain cadence. As a minimum, this cadence must regulate the taking of productivity measurements and provide systematic opportunities for a team to inspect and adapt its process so as to improve it. This simple and effective approach is common to most Lean Kanban delivery teams, which assess their throughput and other measures, such as latency, over set periods of time. Other agile teams will batch related work for completion within each set period in order to achieve a significant product-focused goal. The “Scrum Sprint” is an illustration of the goal-driven approach in action. Success with regard to the goal - which is to say, the delivery of the batch and the mitigation of significant risk - provides a useful point of reference for inspection and adaptation. Sprints occur on a regular cadence of no longer than a month. At the beginning of each Sprint a batch of work can be planned, and at its conclusion a corresponding increment of functionality can be reviewed, demonstrated, and potentially released. The Scrum Team's process will then be subject to inspection and adaptation during an end-of-sprint Retrospective.



Each Delivery Team's cadence of inspection and adaptation should align at least once per month with the Transformation Heartbeat of the organization. This allows multiple teams to synchronize organizational deliveries, and gives the Transformation Rollout Team a regular indication of overall progress towards enterprise agile adoption. For example if an organization has a Transformation Heartbeat of 4 weeks, then teams might reasonably have cadences of 4, 2, or 1 week in length. The cadence of any given team would thus align every first, second, or fourth occurrence with the 4 weekly Transformation Heartbeat.

The Transformation Rollout Team should promote cadences that are appropriate for each individual Delivery Team. Any given team's cadence should be as rapid as efficient practice will allow. The timebox must be short enough to enable inspection and adaptation with minimum delay, yet long enough to permit the retrospective analysis of a significant sweep of team experience. If batches of work are planned into each timebox – as would be the case with Scrum Sprints for example – then

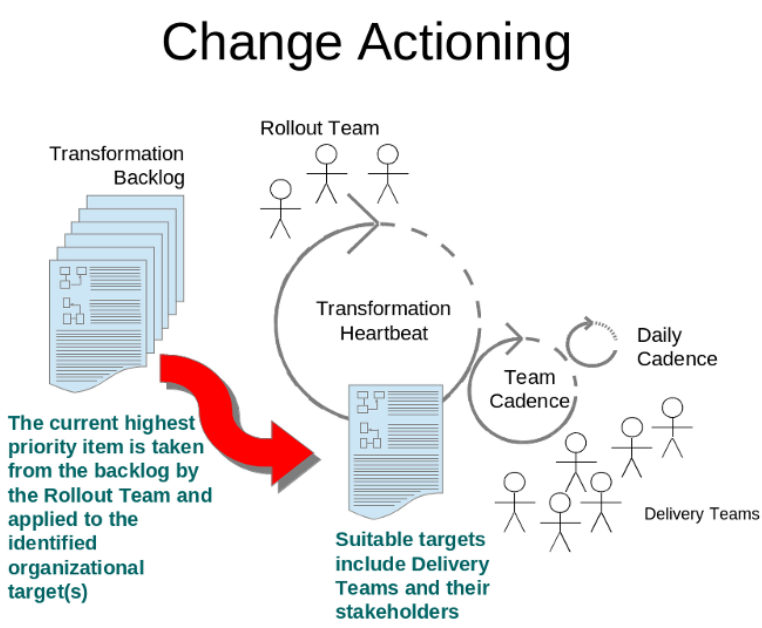
there is an additional consideration to bear in mind. In such cases the Team Cadence must be short enough to stop large batches of undelivered work from building up and depreciating in value, yet long enough to allow meaningful Sprint Goals to be achieved.

Choosing a suitable cadence for a team is hard, and experimentation may be needed before the right balance is found. When promoting a suitable cadence, the Transformation Rollout Team should also consider the ability of the Development Team to absorb items from the Transformation Backlog, and to inspect and adapt its practices accordingly. In the early stages of agile transformation this is likely to be more important than any other single factor.

Step 7: Action change

Agile change is encouraged and coached to the organization by the Transformation Rollout Team. It is their responsibility to ensure that each change is managed and measured, and realizes the benefits of appropriate agile patterns and practices. This is done by helping the target beneficiaries to understand the desired change in their behaviors and practices, the strength of executive sponsorship for change, and the expected outcomes that specific changes will bring. Coaching and mentoring are the watchwords here.

It is not necessary to explain (nor even to mention) the patterns themselves when assisting these parties through the process of change. In fact, “agile patterns” do not even need to form part of their vocabulary. Rather, they must be coached and mentored in the application of appropriate practices that allow each pattern to be realized and its Acceptance Criteria achieved.



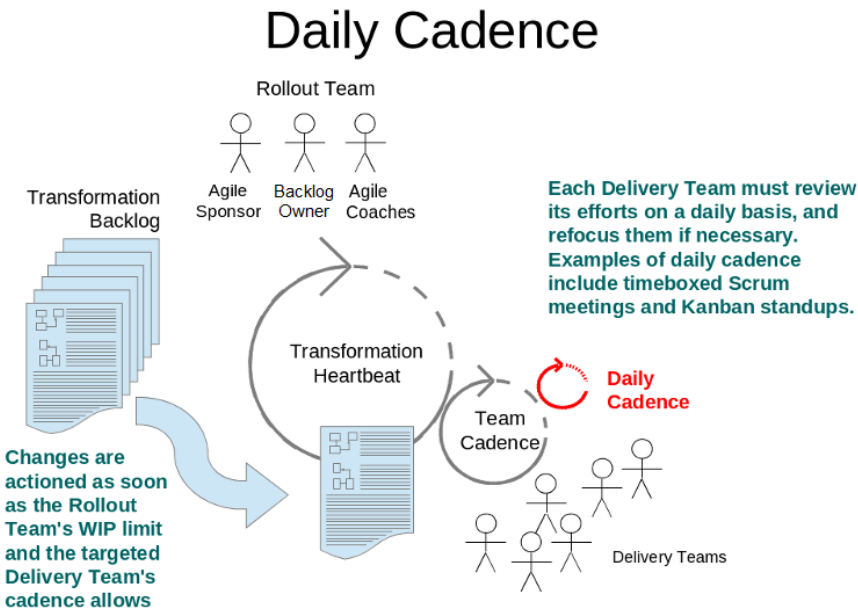
The Transformation Rollout Team actions change from the Transformation Backlog according to each item's order of priority and the Action Plan for implementing that change. A plan will typically involve the coaching and mentoring of teams, and the leveraging of appropriate patterns, so that the Acceptance Criteria of the change can be met.

Step 8: Keep a finger on the pulse, every day

Teams in an agile organization must review the work in progress and the value that is being accumulated and released, taking corrective action if necessary, as early and as often as is practicable. This stops waste from building up and prevents small errors from snowballing into larger ones.

A best practice is to hold brief and focused team meetings once per working day. Fifteen minutes is generally recommended. A timebox like this allows teams to plan their work for the day, and to revise any existing plans if needed, such as plans for meeting a Sprint Goal. This daily cadence of reviewing and refocusing allows for minor course

adjustments, and it helps to ensure the team meets its delivery responsibilities.



From the Transformation Rollout Team's perspective, the Daily Cadence is also an opportunity to assess...with minimum delay...the rollout of changes drawn from the Transformation Backlog. The Daily Cadence preserves the momentum for agile transformation. It can stop the agile initiative from drifting, or indeed from crashing due to a failure to escape organizational gravity. The Rollout Team should look for evidence that each team has made progress in applying the relevant Transformation Backlog Item, that the acceptance criteria are on course to be met, and that the associated improvements are in an increasingly better position to be leveraged. Even if a Rollout Team is working with a Delivery Team on a full-time basis in order to achieve such results, it is enormously valuable to review progress daily.

Suitable illustrations of daily cadence include timeboxed Daily Scrum meetings and Kanban standups or huddles. When done well, the application of several good agile patterns will be seen at work here, including timeboxing, inspection and adaptation, and teamwork.

Step 9: Verify outcomes

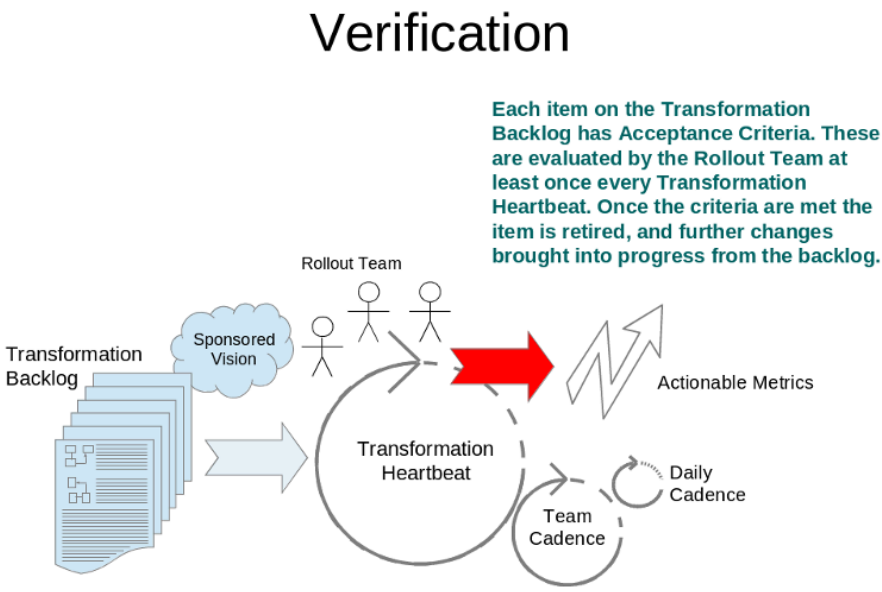
During enterprise agile transformation, changes must be coached to carefully targeted organizational areas, and with reference to good patterns and practices. It's important to have clear results in mind. Agile transformation is not something that is done purely for its own sake, but in order to achieve tangible benefits.

This means that the success of the transformation initiative should be measured objectively. What we specifically care about, at all times, is the delivery of value. To this end, each change on the Transformation Backlog must be qualified in terms that include the intent and motivation for using it, and the consequences that can be expected from its correct implementation. The hypothesis for implementing a change, and for leveraging the relevant good patterns and practices, can thereby be tested. Each change item in the Transformation Backlog is given acceptance criteria for verifying its successful adoption by a targeted party. The Transformation Rollout Team must consider two things: are these criteria being met, and are the expected benefits to the organization also being realized?

Verification addresses the first of these.

For example, the pattern “Limited WIP” is often applied in response to unconstrained demand on a team. A suitable acceptance criterion, for coaching a change that involves this pattern to a certain team, might be a WIP limit of 3. This should therefore be included in the Acceptance Criteria of the change item once it is placed on the Transformation Backlog.

The ability of the targeted Delivery Team to meet the Acceptance Criteria will be evaluated at least once every Transformation Heartbeat. Once a criterion has been demonstrably achieved to the satisfaction of the Transformation Rollout Team then the criterion is deemed to have been met, and the change item can be retired.



In the previous example, if the Rollout Team observe that a WIP limit of 3 is being consistently observed, they would be in a position to retire the change. Have the expected benefits to the organization materialized though? Regardless of whether or not the expected Acceptance Criteria have been satisfied, we must gauge the utility of implementing it. The Rollout Team must therefore look for actionable metrics which can inform them if the Acceptance Criteria were indeed appropriate, or if more work remains to be done by introducing further changes to the backlog.

Step 10: Harvest metrics

Each change in the Transformation Backlog must have acceptance criteria for gauging its successful application. The Rollout Team will verify whether or not these criteria are being met, and must do so at least once every Transformation Heartbeat.

Regardless of whether or not the expected Acceptance Criteria have been satisfied, however, we must understand the usefulness of implementing it. Are the expected benefits to the organization being achieved? The Transformation Rollout Team must therefore look for actionable metrics which can inform them if the Acceptance Criteria were indeed appropriate, or if other work must be done if the desired benefits are to be achieved.

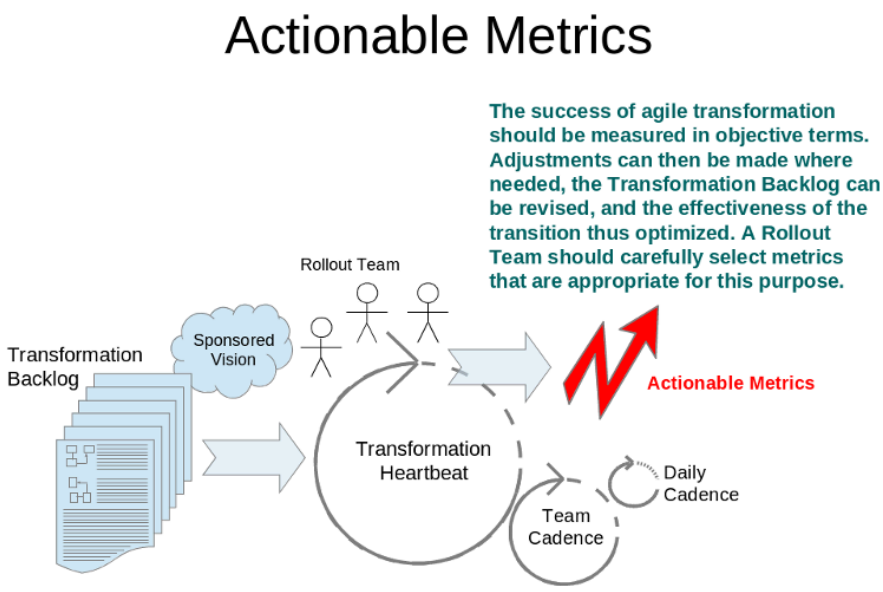
Suppose for example that a change involves the Limited WIP pattern. This pattern details the following consequences that can be expected from its implementation:

- value is delivered earlier
- there will be less stock-on-hand that will be subject to depreciation
- throughput and forecasting will improve
- for projects, the risk of failing to meet a Sprint Goal will be reduced as less work is likely to remain in progress by the end of the iteration

Any or all of these consequences might be used to define “actionable metrics” for gauging the successful application of a corresponding change in which WIP is limited. For example, if the Acceptance Criteria stipulated that WIP must be limited to 3 items, and this was indeed verified, the change may then be retired. However, it does not necessarily follow that any of the above benefits were in fact realized. Measurements will need to be taken in order to find out. We need empirical evidence as to whether or not value was delivered earlier, or depreciation was reduced,

or throughput rose, or progress towards a goal was better controlled. These metrics will inform us of how successful the change was and to take further action accordingly.

Actionable metrics should be evaluated for every change that is currently in progress. This must occur at least once every Transformation Heartbeat. These metrics can then be used to revise the plan for applying the change, and/or to make revisions to the Transformation Backlog.



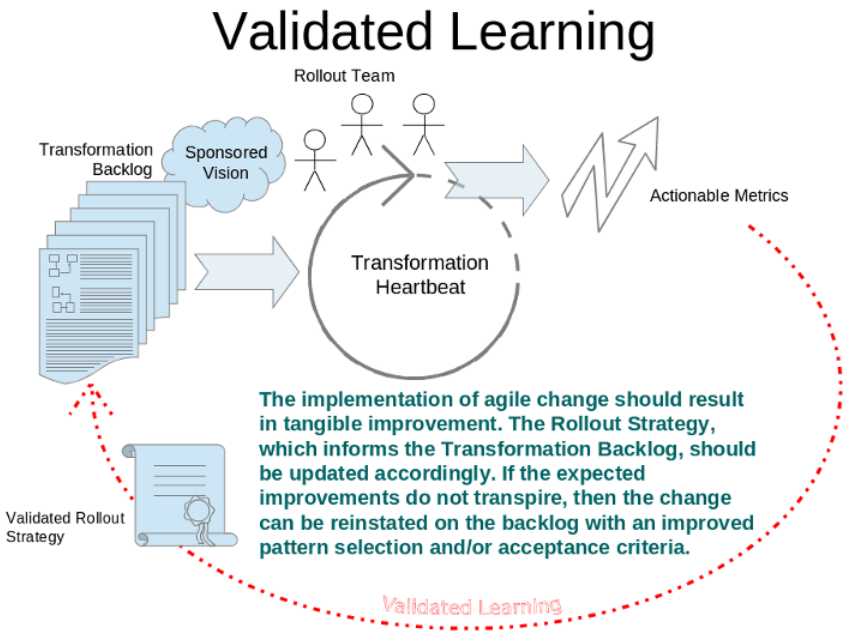
Once its Acceptance Criteria are met, the change ought to be retired, irrespective of whether or not the hypothesized benefits are attained. The next change item will then be drawn from the backlog by the Transformation Rollout Team and actioned.

Note that if the expected benefits have not been realized for a specific change after retirement, then a new item is typically needed. It may appeal to different patterns or implement them in a different way with a different action plan, and the Acceptance Criteria may need be revised.

The change will then be added to the Transformation Backlog and prioritized accordingly. This feedback loop, in which value and quality are subject to continuous improvement, is referred to as “Validated Learning”.

Step 11: Validate your learning

Validated Learning is the use of feedback to inform and improve future actions. It is important to improve an organization's ability to undergo enterprise agile transformation. This improvement is the primary responsibility of the Transformation Rollout Team at any given level or area of the business. Those teams decide what changes should go on their Transformation Backlogs and what the priorities should be. They decide where good agile patterns needs to be applied in regard to each change, and how and for what purpose, and what benefits may be expected from doing so.



Each change on the backlog is given Acceptance Criteria by the Transformation Rollout Team. These criteria stipulate the conditions that must be satisfied in order for the change to be implemented. Different change items may have different acceptance criteria even when the same patterns are being applied. For example, if the Limited WIP pattern is being coached to a particular Delivery Team, a suitable criterion might be that only 3 items can be in progress at a time. Other teams may be allocated different WIP limits. The more appropriate and better focused the acceptance criteria, the greater the chances are that the benefits of the pattern will be leveraged.

The Transformation Rollout Team must therefore determine whether or not the Acceptance Criteria are suitable for the intended target. This evaluation should be done on an ongoing basis, and at least once every Transformation Heartbeat for each change that is being rolled out. This determination is made on the basis of actionable metrics.

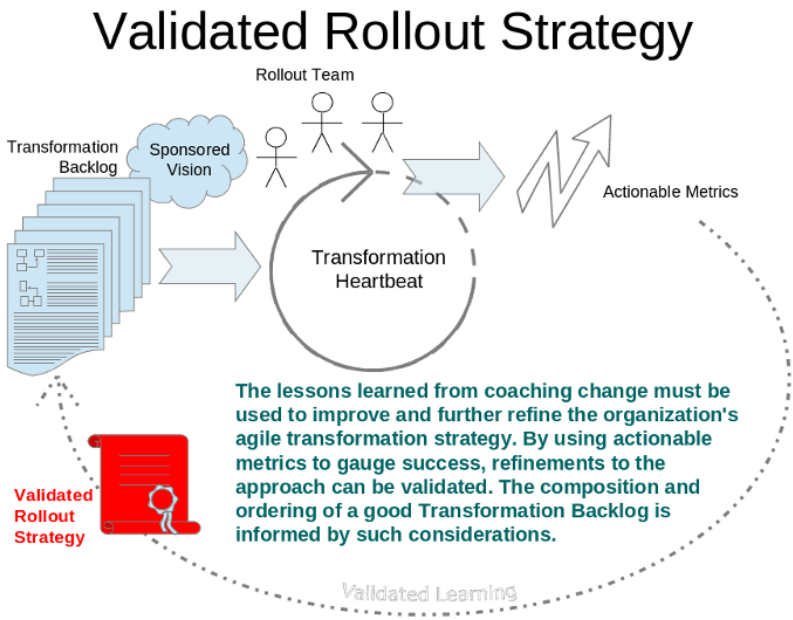
- If the metrics indicate that the expected benefits have been achieved then the change may be retired.
- If the benefits have not been achieved, regardless of whether or not the Acceptance Criteria have been met, then further work remains to be done. The Rollout Team can replan the Transformation Backlog, and the Acceptance Criteria for relevant changes may be improved.

Note that if the benefits have not been attained even though the Acceptance Criteria have been satisfied, then the change should still be retired. The lesson is that the Acceptance Criteria were inappropriate for the change target. A more appropriate change, with improved Acceptance Criteria and possibly implementing different patterns or practices, should be added to the backlog and prioritized.

Step 12: Validate your transformational rollout strategy

When a Transformation Backlog is first created, none of the change items will yet have been applied. The Transformation Rollout Team may not yet have had the opportunity to coach any good practices at all within the organization. No hard and direct experience of doing so may exist. This means that a transformation strategy may have to be initially formulated on the basis of:

- experiences gained elsewhere
- the Transformation Rollout Team's understanding of the organization's present state
- the vision for change that the transformation sponsors have articulated



The team should expect and plan to have valuable feedback no later than the completion of the first Transformation Heartbeat. Actionable Metrics should have been elicited at that point, and the content of the Transformation Backlog may be revised on the strength of the lessons

learned. This means that new changes can be added and their Acceptance Criteria can be shaped by recent experience, and different patterns may be exploited in different ways. Any entries that are no longer appropriate can be challenged and potentially removed. Other parts of the enterprise can perhaps be engaged in the change process, and the priorities given to each item in the Transformation Backlog can be reconsidered.

A validated transformational rollout strategy does not have to be a document. It can simply be the collected and shared experience of the Rollout Team throughout the agile transformation. The important thing is that it must be evaluated in terms of actionable metrics at least once every Transformation Heartbeat, and that it informs the revision of the Transformation Backlog. The validation or revision of the transformational strategy provides closure to each and every cycle of the validated learning loop.

Key Takeaway: A transformation must be agile in itself

If an agile transformation is to succeed, the associated patterns and practices must be implemented in a managed fashion. It's no use setting about change in a haphazard way, trying to do everything at once in the hope that “some of it sticks”. Essentially, transformation itself must be seen as an agile process, and an approach is therefore needed through which transformation can be galvanised. We must also recognise that large organizations are not typically start-ups, and the process of transforming them into more agile enterprises cannot interrupt the ongoing delivery of value. By applying an iterative-incremental framework and the principles of empiricism and backlog management, organizations can harness the agility and innovative potential normally associated with start-ups, but at enterprise scale.

---

*Cunningham, M. Agile Ottawa presentation, May 2012

Appendix 1: Agile Patterns of Method

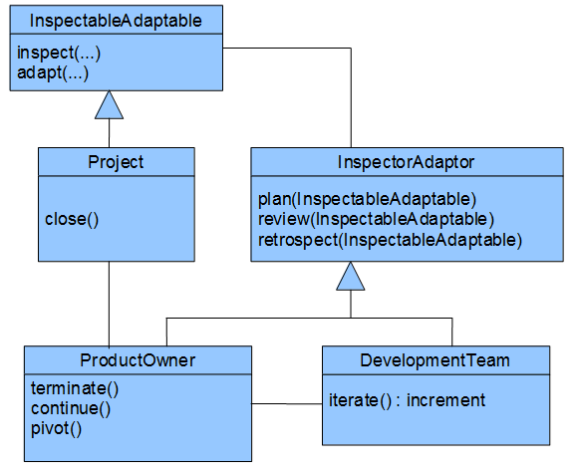
Controlled Failure

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

Proverbs:

- Don't throw good money after bad
- If you're in a hole, stop digging
- Failure breeds success

Controlled Failure



Also Known As:

- Fail Early Fail Fast
- Fail Fast Fail Often
- Fail Fast Fail Cheap

Motivation: It is important to avoid a continued project investment if it is unlikely to deliver sufficient value.

Structure: A transparent project will support inspection and adaptation. The Product Owner and the Development Team must be able to inspect and adapt the process. At the conclusion of each iteration, a potentially releasable increment must be delivered by the team to the Product Owner (PO). This increment will then be reviewed, , the development process considered in retrospect and improved, and future work planned. If the PO is not satisfied that sufficient value is likely to be delivered then he or she may terminate the project.

Applicability: 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.

Consequences: When a controlled failure is done well, the losses incurred by a

failing project should not exceed those resources consumed in the current iteration. However, any failure - controlled or otherwise - may have political repercussions.

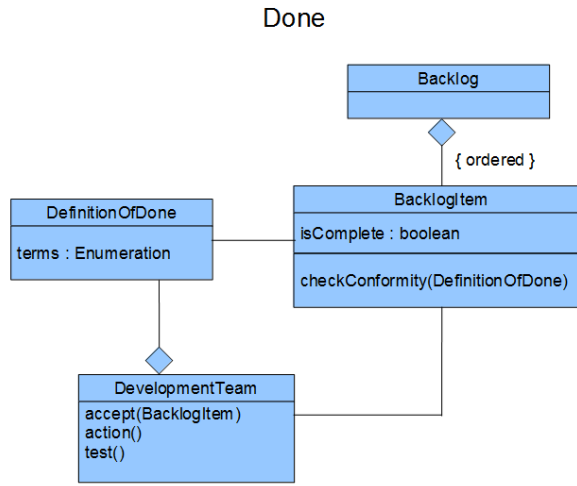
Implementation: All projects should be evaluated by the Product Owner with regard to whether or not the intended value is still likely to be delivered. Reviews, Retrospectives, and Planning sessions all represent suitable points at which a controlled failure may be initiated. It is irregular to implement a Controlled Failure before the current iteration has terminated and an increment is available for evaluation. The implementation of controlled failure requires courage. It can be tempting to continue with an unprofitable venture when resources have already been invested in it (see Death March). A controlled failure should secure the release of any value accrued to date, including as much as possible from the final iteration, identify and harness the lessons that must be learned, and free project resources for other more profitable activities.

Done

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

Proverbs:

- A stitch in time saves nine
- If a job is worth doing it is worth doing well
- The knot tied by a wise man cannot be undone by a fool



Also Known As:

- Definition of Done
- Tolerances

Motivation: The state of a product's completeness can be a source of misunderstanding between stakeholders. For example, work that has been developed but not integration tested may be considered to be complete ("done") by a delivery team. An end customer, however, may expect that a completed deliverable has also been integration tested. They may also have expectations

regarding documentation and end-user training that the development team do not satisfy. Similar discrepancies in understanding can also cause problems within delivery teams. If one developer holds to a different standard of completion than another, then the team will not be able to assert the quality of its deliverables.

Structure: Each backlog item is correlated to general terms of acceptance that apply to all. The development team will accept a backlog item, action it, and test it for compliance with each of those enumerated terms before asserting it to be complete.

Applicability: 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. The pattern can however be applied at any station in a workflow. For example, a requirement might need to meet user story INVEST criteria before it can be accepted into a Sprint Backlog.

Consequences: A consistent understanding of “done” helps ascertain a deliverable’s fitness for purpose. Inconsistencies in understanding can be expected to lead to confusion, rework, or the unmanaged accumulation of technical debt.

Implementation: 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. Some development teams use the terminology “Developer Done” to indicate that they have taken their work as far as possible, and “Done-Done” to mean that work is fit for potential release. For example, “Developer Done” may include unit testing, but not quality assurance testing. However, this usage suggests that the team do not wholly own their process, and that the silo anti-pattern might be in evidence.

Inspect and Adapt

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

Proverbs:

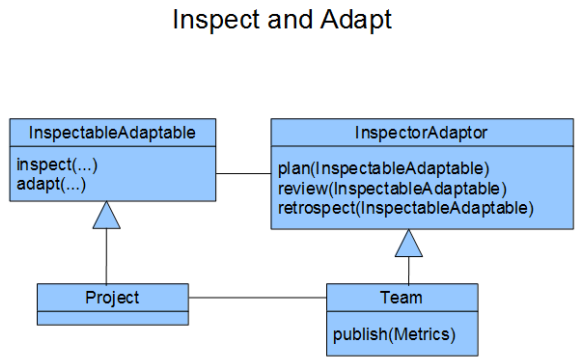
- Those who cannot remember the past are condemned to repeat it
- Work smarter, not harder
- Practice makes perfect
- An ounce of prevention is worth a pound of cure

Also Known As:

- Change Opportunity

Motivation: 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.

Structure: A transparent project will support inspection and adaptation. A team must be able to inspect and adapt the process. At the conclusion of each iteration, a potentially releasable increment must be delivered by the team to the Product Owner (PO). This increment will then be reviewed, the development process considered in retrospect and improved, and future work planned. The team will publish metrics that allow improvements to be quantified as well as qualified.



Applicability: Inspection and adaptation must occur as closely as possible to where the work is done. A team will inspect and adapt its products and processes at regular points so that the quality of delivery can be improved.

Consequences: Inspection and adaptation facilitates continuous process improvement. Teams that regularly inspect and adapt their processes demonstrate greater responsibility for their deliverables. Delegation of responsibility is thus encouraged, and improvements can be leveraged.

Implementation: 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. Feedback and the courage to adapt are core values within XP.

Iterate

Intent: Handle small pieces of work at regular intervals

Proverbs:

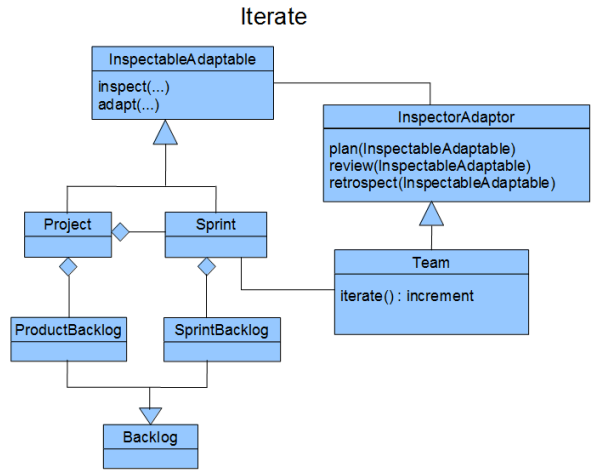
- Many small steps make a journey
- A rolling stone gathers no moss

Also Known As:

- Sprint

Motivation: Linear planning has a very limited capacity to support change, and defers the delivery of value beyond stage gates. The iterative rebaselining of plans minimizes the risk of change and the potential for waste, while also providing regular and frequent opportunities for the incremental delivery of value.

Structure: A project will have many Sprints (iterations). A team will tackle the project one Sprint at a time. They will select a Sprint Backlog of work from the Project's Product Backlog. The success of each Sprint can be determined by inspection, and working practices adapted so that future Sprints are more effective.



Applicability: Iterative development and delivery is a feature 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 an 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.

Consequences: Iterative development allows plans to be revised at regular time-boxed intervals. When accompanied with incremental releases of value, product depreciation is minimized and the risks associated with delivery are reduced.

Implementation: 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

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

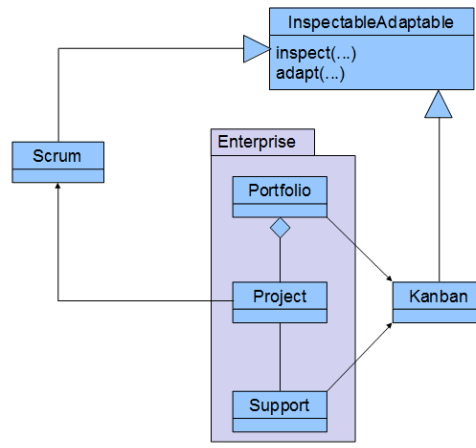
Proverbs:

- Different sores must have different salves

Also Known As:

- Agile Scalability Model

Kanban Sandwich



Motivation: An agile enterprise transformation affects the entire business. Kanban teams are best suited in technical support, while projects are more likely to be run in Scrum fashion. At the business level we often find "project pipelines" that become Kanban-like once Agile practices are embedded in the boardroom.

Structure: Portfolio and Support levels are both run as Kanbans, while Project level work is run in Scrum fashion. Both methods support inspection and adaptation.

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

Consequences: Use of this pattern may help to facilitate release planning if the delivery cycles are synchronized.

Implementation: This pattern can be found in the Scaled Agile Framework. It can also be correlated to the agility@scale approach where such issues have

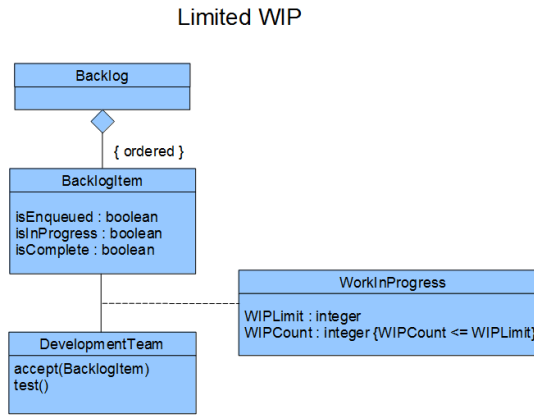
been framed in the context of DevOps and a business-driven project pipeline that supports architectural visioning.

Limited WIP

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

Proverbs:

- Don't bite off more than you can chew
- Givers have to set limits because takers rarely do



Also Known As:

- Lean Pull

Motivation: The pressure upon teams of finite size to respond to unconstrained demand can lead to excessive work in progress. The result is that the delivery of any one item becomes delayed as attention shifts to another, and the value invested is given an opportunity to depreciate. By limiting the amount of work that can be progressed at any one time, this problem is mitigated and the problem is reduced to one of backlog prioritization and quality of service.

Structure: Each backlog item that is currently being worked on by a development team will count as Work In Progress, and the total count will not exceed the established WIP limit.

Applicability: 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.

Consequences: The limitation of WIP will allow value to be delivered earlier, and there will be less stock-on-hand that will be subject to depreciation. Throughput and forecasting will improve. For projects, the risk of failing to meet a Sprint Goal will be reduced as less work is likely to remain in progress by the end of the iteration.

Implementation: Limited WIP is a key feature in all Kanban and Lean configurations. 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. 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. A limitation of WIP is therefore also considered to be good (if not essential) practice in Scrum.

Pivot

Intent: Realign a project to fit a better goal

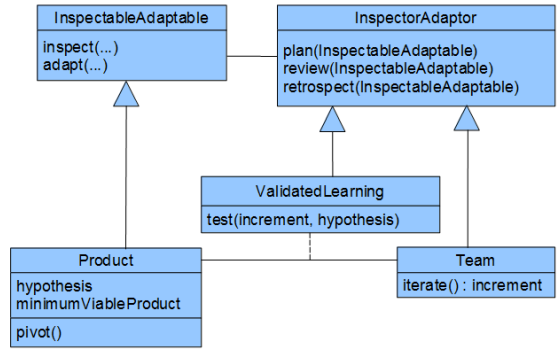
Proverbs:

- When one door shuts another one opens
- Strike while the iron hot

Also Known As:

- Strategic agility
- Structured course correction

Pivot



is

Motivation: The choices made for a project are subject to the information that is available at the time. A structured course correction may be needed that allows a new hypothesis involving different choices to be tested. These choices can involve customer requirements, the delivery strategy including technology and architecture decisions, and the model for growth.

Structure: A team delivers a product increment at the end of each iteration. The increment can test a new hypothesis by revising the Minimum Viable Product (MVP) and subjecting the results to inspection. The Product can then be adapted (pivoted) to this new MVP if the hypothesis is shown to be valid.

Applicability: Pivoting is applicable for innovative environments where the right product choices need to be established quickly.

Consequences: Pivoting allows improved products to be brought to market rapidly while leveraging value already earned. Risk is constrained to the construction of a revised MVP that allows a new market hypothesis to be validated. Customers may benefit from disruptive innovation in the marketplace.

Implementation: Pivoting is a key feature of the Lean Startup movement. It has not yet become an established feature in other agile methods.

Red-Green-Refactor

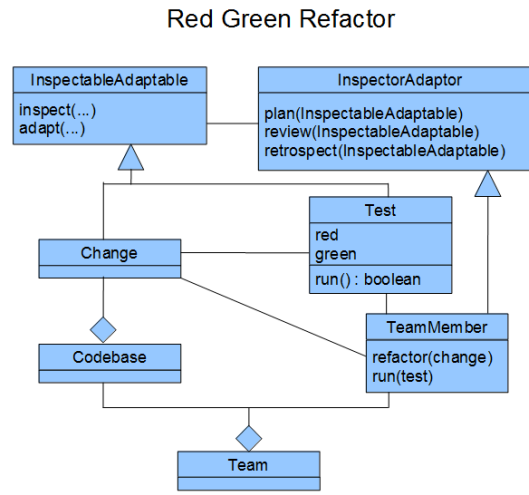
Intent: First prove that an unmet need has been met without breaking anything else, then optimize the solution

Proverbs:

- If it ain't broke don't fix it
- Make it work, then make it fast

Also Known As:

- Test Driven Development
- Test First
- Fail First



Motivation: Assert that each code change is necessary and fit for its intended purpose

Structure: 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 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. Previously written tests should also be run before any change is committed, as other tested code may have been compromised by the modification.

Applicability: Test Driven Development is applicable to all forms of development. It is also applicable for maintenance and repair. For example, if a flaw is discovered a test can written that expresses the desired behavior. The test will fail, and a fix then introduced, at which point the test will pass.

Consequences: Test Driven Development is a significant culture change for many teams and throughput may initially drop as the practice beds in. Introducing TDD may be controversial in organizations with a separate QA testing function. However, productivity generally increases in line with

improvements in code quality, and reduced rework can be expected. The practice also encourages a better understanding of requirements and their acceptance criteria. An automated test environment is recommended to gain the full benefits of TDD, since all tests should be run before any change is committed.

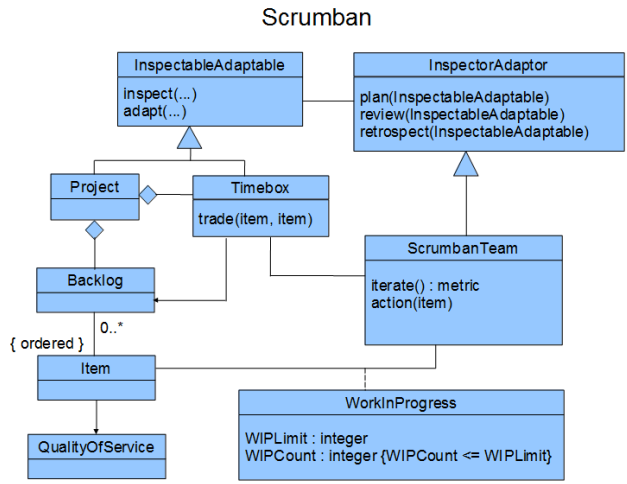
Implementation: TDD is an integral part of the Extreme Programming method. It is also found in some Scrum, SAFe, and DSDM implementations. It is widely aspired to as a best practice in agile development. Most implementations of TDD are done in a unit testing context. Behavior Driven Development is a variation that uses business-focused acceptance criteria as the tests.

Scrumban

Intent: Handle a mixture of project and unrelated work, in an agile manner, by implementing a hybrid of Scrum and Kanban rules.

Proverbs:

- What is not urgent must be done quickly in order to take care of the urgent things calmly
- Cut your coat according to your cloth



Also Known As:

- Fast Track (when used in a Scrum context)

Motivation: An organization might not have the resources to dedicate a project team to the development of a product. These constraints often lead to Scrum Teams being given work to do that is unrelated to their Product Backlog and/or which may compromise their Sprint Backlog. This work can include maintenance (BAU) activities or emergency incident response.

Structure: A project supports multiple iterations each of which is time-boxed. The team will plan backlog items into each time-box, and with the intention of actioning them at some point in the corresponding iteration. Only a limited

number of those planned items can be work in progress at any one time. Each item has a defined Quality of Service that determines how it will be actioned. Items that were not on the backlog, and which were not planned into the time-box, may be done within the time-box if they have a Quality of Service that mandates their immediate attention. However, a suitable backlog item must be traded out of the time-box in order to accommodate it.

Applicability: Scrumban is a compromise to the problem of limited resources and competing demands for project and unrelated work. It therefore finds wide application across the IT industry.

Consequences: Scrum is predicated on the ability of a team to plan their work for a time-box. Pre-empting that plan with other (unplanned) work may therefore compromise a team's ability to meet their Sprint Goal. All stakeholders must therefore be informed if such pre-emption will be allowed to occur.

Implementation: Many organizations that claim to have implemented Scrum are in fact allowing the quality of service to vary by means of a fast-track lane or similar, and are thereby implementing Scrumban.

Single Piece Flow

Intent: Minimize stock-on-hand to one item so as to deliver value more quickly and reduce waste

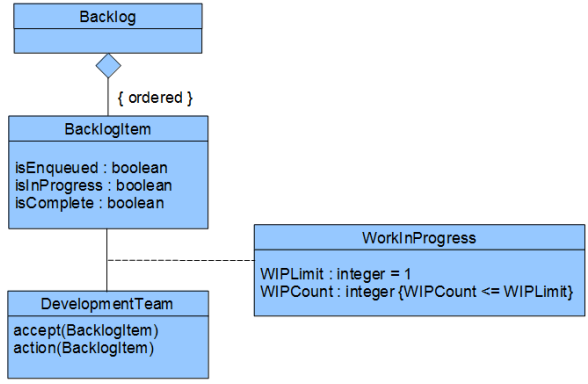
Proverbs:

- It's best to do one thing at a time

Also Known As:

- One Piece Flow

Single Piece Flow



Motivation: In theory, the most efficient way to work is one item at a time. This minimizes the costs associated with maintaining inventory (stock on hand) and also brings value to market more quickly, thereby allowing a return on investment to be leveraged sooner.

Structure: A development team accepts and actions only one backlog item at a time. The Work In Progress is limited to one.

Applicability: Single Piece Flow should be considered in situations where the value stream needs to be optimized as far as possible. Note that for Single Piece Flow to be viable, it must be possible for multiple developers to work on one item simultaneously and without causing each other impediment.

Consequences: Single Piece Flow, when applied in the right situation, can result in a faster time to market, an earlier return on investment, reduced inventory costs, and rapid feedback cycles. However, it can also lead to waste if developers are allowed to get in each others way.

Implementation: Single piece flow can be implemented for Lean Kanban teams of small size, where developers will not cause each other impediment by working on the same item. It is rarely implemented in Scrum development teams, since they are expected to formulate their own plan for the delivery of a Sprint Backlog, the release of value being deferred until the end of a Sprint and not before.

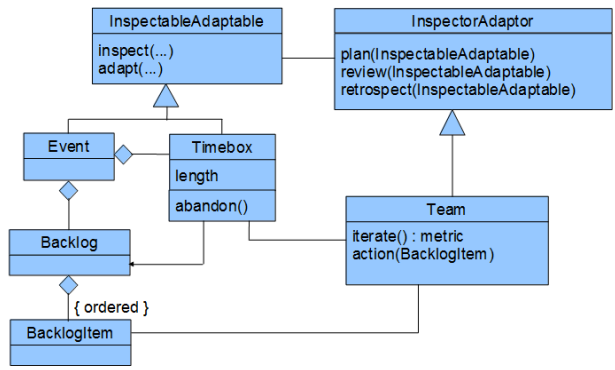
Timebox

Intent: Limit the time that is available for a task in order to avoid procrastination

Proverbs:

- Work expands to fill the time available (Parkinson's Law)
- Time is not so important as making time count

Timebox



Also Known As:

- Time Limited Activity

Motivation: Work that does not have a clear deadline is likely to be delayed, while focus on the completion of core requirements may be lost. If the time available is limited, it helps to provide a focus, and increases the chances of essential scope being addressed within the allotted time span.

Structure: A time-limitable event, such as a meeting or project, will have a backlog of items for actioning. A team will plan to action certain items from the backlog in an iterative manner. Each iteration will be time-boxed to be of a certain length. At the end of the time-box, the success of the iteration can be reviewed, and the process followed can be retrospectively adapted. Each time-boxed iteration will produce metrics that can be inspected and compared for this purpose. The early delivery of value does not shorten a time-box; it will terminate at the predetermined time. Note that it should only possible to abandon a time-box in very exceptional and clearly defined circumstances.

Applicability: In agile practice, time-boxing is appropriate for any activity that is expected to deliver a result of value. It would be highly irregular in any agile way of working to conduct an activity that is not time-boxed.

Consequences: The time-boxing of an activity can increase focus on the delivery of value. However, it also incurs the risk that not all planned items will be actioned within the allocated time. The prioritization of backlog items is therefore important.

Implementation: In Scrum, Sprints are time-boxed to a length of no more than one calendar month. Reviews, Retrospectives, Planning Sessions, and Daily Stand-ups are also time-boxed.

Trade

Intent: Change the content of a backlog without changing the overall size

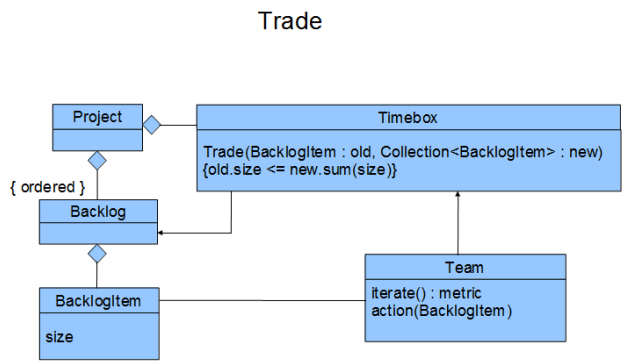
Proverbs:

- Fair exchange is no robbery

Also Known As:

- Quid Pro Quo

Motivation: Allow on-the-fly changes to the work planned for completion in a time-box



Structure: A backlog that has been negotiated for completion in a time-box can contain multiple backlog items. Each one of those items must be sized. The team will be prepared to trade items in and out of the backlog even after the time-box has commenced, as long as the item has not yet been actioned. Traded

items must be of equivalent size and the magnitude of the team's commitment will not be altered. The team wholly own their time-boxed backlog, and no trade can be made without their understanding and approval.

Applicability: Trading is applicable when unactioned requirements are de-scoped after an iteration commences, and when other requirements of equivalent size would be more valuable. The Sprint Goal should not be compromised as the result of making such a trade.

Consequences: The trading of items usually incurs some cost; an old backlog item of size X may have to be traded for an item of size (X – y) in order to accommodate the overhead of replanning. Trading can imply weak product ownership and/or weak iteration planning. Trading project requirements for unrelated work often indicates weak or split product ownership. Commitment-based planning is incompatible with scope trading, as the commitment will by definition change.

Implementation: Scrum has replaced commitment-based planning with a forecast-based view of a Sprint since 2012. As such, in-Sprint scope trading can be said to be supported in Scrum. DSDM allows "should have" and "could have" requirements to be traded out-of-scope in the event that "must have" requirements prove to be more demanding than estimated.

Appendix 2: Agile Patterns of Responsibility

Management by Exception

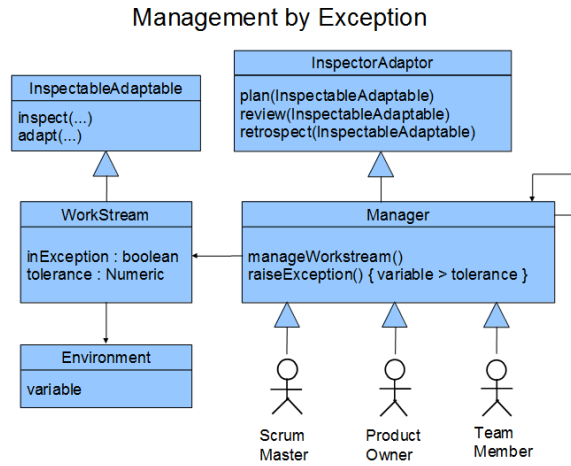
Intent: Delegate authority to act within specified tolerances, and only involve others if those tolerances are exceeded

Proverbs:

- First try and fix it yourself

Also Known As:

- Escalation (especially in a line management context). Note that the term Management by Exception is sometimes used to describe fire-fighting.



Motivation: Localize responsibility, avoid micro-management and the unnecessary seeking of approval

Structure: A workstream (such as a project) can only operate within certain tolerances. Tolerances might be constraints on time, cost, resource, quality, scope, or any other variable within a workstream's environment. The conformance of a workstream to those tolerances is measurable by inspection. A manager (such as an agile team member) will have the authority to inspect and adapt his or her workstream within the prescribed tolerances. If the tolerances are exceeded then an exception will be raised.

Applicability: Management by Exception is suitable for teams with clearly demarcated responsibilities, operational tolerances, or constraints.

Consequences: Management by Exception empowers parties to operate within certain tolerances. Raising an exception transfers responsibility and accountability to others should those tolerances be exceeded. However, it is usually most efficient to resolve problems as close to the source as possible. It is therefore important that the allocated tolerances do not unduly limit an agile team's freedom of operation.

Implementation: Agile projects usually exhibit tolerances on scope rather than time, cost, resource, or quality. Team empowerment is recognized in DSDM, Scrum, and XP. All support Management by Exception.

Peer

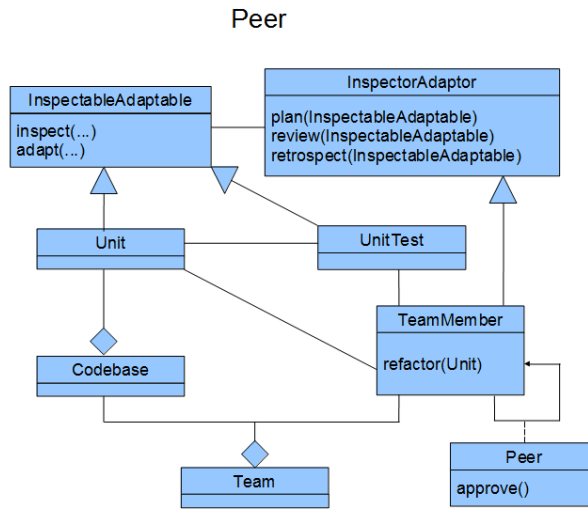
Intent: Remove skill silos, encourage team responsibility and accountability

Proverbs:

- A problem shared is a problem halved
- Another pair of eyes often helps

Also Known As:

- Cross-Skilled Developer



Motivation: The quality of work done by individual team members is the responsibility of the team. Recognizing fellow team members as peers is critical if there is to be effective team ownership and quality control.

Structure: A team has a codebase and multiple team members, any of which may be a peer to another. Each unit of work that is done by a team member is validated by a unit test, and must be inspected by a peer before it can be committed to the codebase. Units of work, and their tests, can be inspected and adapted by team members, including those in the role of peer, in order to maintain quality control.

Applicability: The assignment of peers to team members is appropriate when the team as a whole cannot exercise effective and consistent jurisdiction over the work of all members. This is often the case when each team member is allowed

a discrete piece of work, such as a user story or task, to action. The assignment of peers can become unnecessary when team collaboration is an entrenched practice; this can be the case with Single Piece Flow.

Consequences: The usage of peers can improve the consistency of work products, and provides a mechanism for team oversight of constituent members and quality assurance. Reduced rework is likely. Since peers are by definition reviewing the work of others, this can lead to frictions including the perception that individual performance is being criticized. Training is required in order to overcome this. There is also a danger that the use of peers will be seen as waste and the duplication of effort.

Implementation: Peers are most commonly found in two contexts. Peer Review is the ad-hoc assignment of a peer for the purposes of checking the conformance of work items to their acceptance criteria and the team's Definition of Done. This is quite common in a Scrum setting. Pair Programming is the systemic use of peers for development and delivery; a peer will continually collaborate with a fellow on the actioning of work. Pair Programming is most commonly associated with Extreme Programming and Test Driven Development, though it may also be found in Scrum and DSDM implementations.

Product Ownership

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

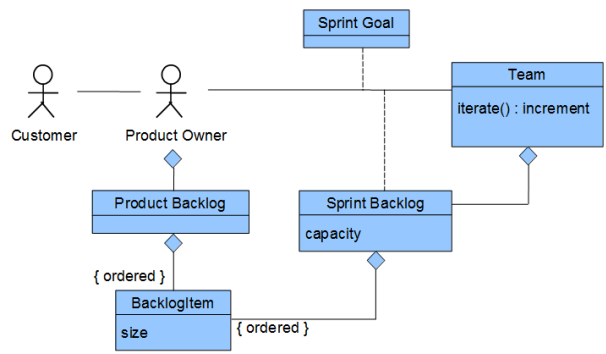
Proverbs:

- Too many cooks spoil the broth
- A single wringable neck

Also Known As:

- Onsite Customer
- Business Owner

Product Ownership



Motivation: Explaining customer requirements to technical teams, and liaising with them on the delivery of value, is a skilled job. Customers cannot be expected to have these skills and technical teams cannot be expected to make business decisions regarding product value. A single voice is needed who, as an

empowered liaison, can fulfill this role and perform duties to both client and team.

Structure: A Product Owner will wholly own an ordered backlog of product requirements. Each backlog item will represent value to the customer and the Product Owner will be in a position to explain each one to the team, and to rewrite it if necessary in order to improve clarity. The team will then size each item and the Product Owner will order the backlog in the customer's best interests. The Product Owner and the Team will negotiate which items should be brought into the Sprint Backlog for the next iteration in accordance with a suitable Sprint Goal.

Applicability: Product Ownership is key to a collaborative way of working, and as such it is applicable to all agile methods

Consequences: Product Ownership encourages teamwork. It gives customer and technical teams a common goal and provides both with an incrementally emergent schedule in the form of a Product Backlog. This requires the availability of someone with appropriate skills, availability, and authority. Such people can be difficult to resource and Product Ownership by proxy has become common. Failure to resource a suitable Product Owner or proxy implies that product value cannot be represented and that the Business Case should be reconsidered.

Implementation: 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

Intent: Vary the way backlog items are expedited and handled

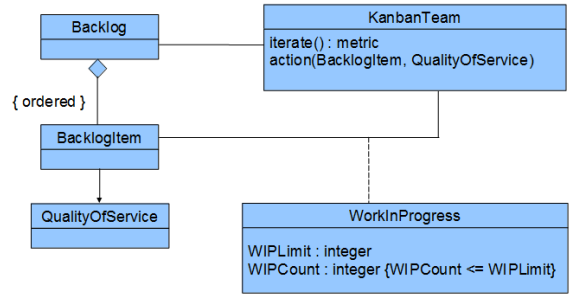
Quality of Service

Proverbs:

- Some are more equal than others

Also Known As:

- Fast tracking (when used in a Scrum or



Scrumban context)

Motivation: Not all items on a backlog are necessarily subject to the same terms of service. Examples include enhancements and defect fixes, the impact or severity of which may vary. The quality of service provided to those backlog items must therefore be allowed to vary as well. Note that this motivation typically applies to Business As Usual workstreams such as technical support. The quality of project work should be invariant.

Structure: A team selects which items from a backlog to action according to the prescribed Quality of Service and only then will the order of items be considered. The Work In Progress limit may be revised to support the terms of service of a selected item.

Applicability: Quality of Service is applicable to situations where the handling of work varies by context. 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.

Consequences: Quality is no longer invariant. The order of backlog items only partly indicates their relative priority since their Quality of Service is also considered.

Implementation: Quality of Service is allowed to vary in most Lean-Kanban implementations. These teams provide Business as Usual work in a support context. This pattern is not found in DSDM, XP, or Scrum as these methods hold quality to be invariant. Hybrid Scrum-Kanban implementations that allow the Quality of Service to vary (for example by supporting a “fast track” or “expedite” class of service) are widespread.

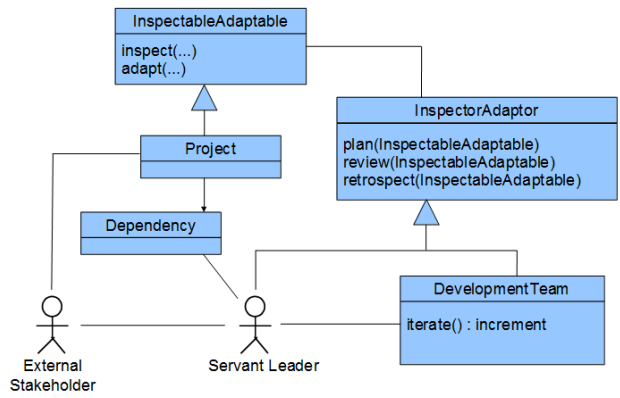
Servant Leadership

Intent: Protect a team from distraction and impediment, and facilitate their progression towards their goals.

Proverbs:

- The utmost form of respect is to give sincerely of your presence
- Sow a thought and you reap an action, sow an act

Servant Leadership



and you reap a habit
- Noblesse oblige

Also Known As:

- Scrum Master (this term is not used exclusively in a Scrum context)

Motivation: If a team is to self-organize in order to meet a goal, they must be permitted the freedom to do so, and provided with the guidance and support they need. This implies that a manager is needed who can smooth their progress and help them to make and meet their commitments. In short good leadership is required, and the best leaders serve their team…the team does not serve them.

Structure: A Servant Leader helps a team to 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.

Applicability: Servant Leadership is applicable to all agile methods. The term Scrum Master is commonly used to describe this role, and is no longer restricted to Scrum alone.

Consequences: Servant Leadership can involve a wider remit of organizational coaching. If impediments originate from other departments, there may be political barriers to their removal.

Implementation: The canonical implementation of Servant Leadership is the Scrum Master. 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.

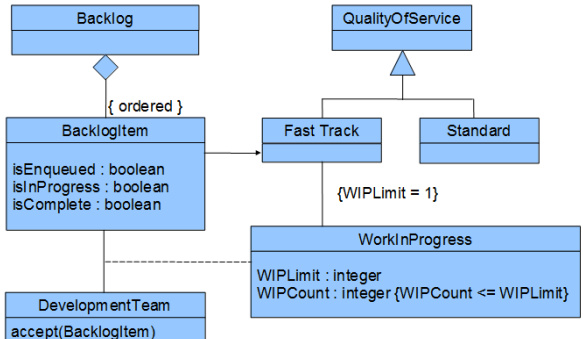
Swarm

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

Proverbs:

- It's best to do one thing at a time
- Many hands make light work

Swarm



Also Known As:

- Single Piece Flow (when used in context of Quality of Service)

Motivation: In theory, the most efficient way to work is one item at a time. Critical defect fixes can require this high level of as and when they arise.

Structure: A development team accepts and actions only one backlog item if the associated Quality of Service demands that such prioritization be given. The Work In Progress is limited to one.

Applicability: 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.

Consequences: Swarming, when applied in the right situation, can result in a faster response time and a more rapid turnover of defect fixes and small changes. However, it can also lead to waste if developers are allowed to get in each others way.

Implementation: 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. It is rarely implemented in Scrum development teams, since they are expected to work on the delivery of a Sprint Backlog for the achievement of a Sprint Goal, and not to vary the quality of service of individual backlog items.

Teamwork

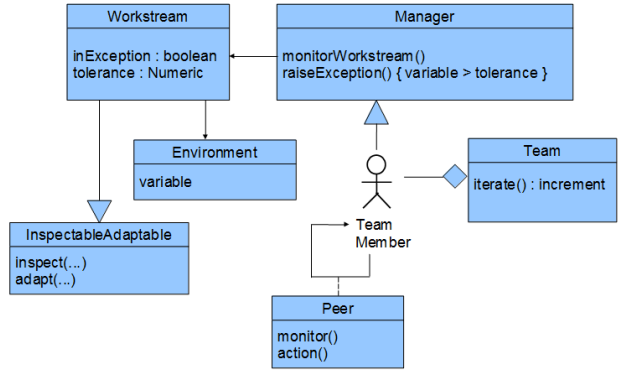
Intent: Get people to collaborate on work being actioned so that increments are developed as efficiently as possible

Proverbs:

- Many hands make light work
- None of us is as smart as all of us

Also Known As:

Teamwork



- Collaboration

Motivation: There is value to be found in co-operation and self-organization. Quality improves through peer review and inspection and rework is consequently reduced. Efficiencies of scale can be leveraged as management overhead is less for a self-organizing team than it would be for individual workers.

Structure: A team member does not work in isolation, but as part of a team consisting of peers. Each member can therefore action work and monitor the progress of work actioned by others. Team members collaborate in iterative and incremental delivery. They are in a position to monitor their workstream and jointly inspect and adapt the process they jointly follow. They can also raise exceptions by consensus if the tolerance of appropriate variables is exceeded.

Applicability: Teamwork is not only applicable but essential to all agile methods.

Consequences: Teamwork removes skill silos. In its most efficient implementation all team members will be cross-trained. They will be able to “go to where the work is” and action it regardless of the skills required. The removal of these silos can be politically challenging, as some workers are constrained in terms of what they are allowed to do, or simply do not wish to be cross-trained.

Implementation: Scrum recognizes both a Development Team, without skill silos, and a larger Scrum Team that includes a Scrum Master and a Product Owner. XP values teamwork greatly and pair programming is a key feature of this. DSDM is more complex and there are around a dozen business, technical, and supporting roles that make up its collaborative model.

Release Orchestration

Intent: Ensure 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.

Proverbs:

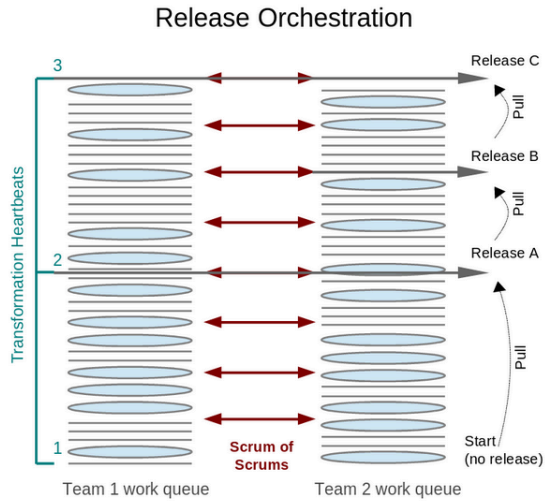
- No one can whistle a symphony, it takes an orchestra to play it

Also Known As:

- Release Management

- Release Planning. Note that the emphasis in agile Release Orchestration is to minimize release planning, and associated batch sizes, as far as possible.

Motivation: Delivery Teams should collaborate between themselves and release more often to better support demand (consumer pull). Releases may then occur as often as teams and their Product Owners are able to arrange.



Structure: An organization determines that it will release value at a certain cadence, with a view towards eventually supporting Release on Demand. This cadence, or transformation heartbeat, is a regular inspect-and-adapt opportunity for gauging the success of enterprise agile adoption and the lean delivery of value. Delivery Teams will collaborate (by means of a “Scrum of Scrums” or similar body) to ensure that releases do in fact occur on each heartbeat. Product Owners will exert pull for release on a more frequent and self-organized basis.

Applicability: Release Management is applicable to all agile methods, although very lean ones may automate much of this through continuous integration and deployment (Release on Demand).

Consequences: Value is delivered more frequently and work done has less opportunity to depreciate.

Implementation: Holding a “Scrum of Scrums” between Delivery Teams is one way to orchestrate releases. DevOps and Lean Kanban teams may use a Release on Demand model. During agile transformation, releases should be aligned at least once per organizational inspect and-adapt-cycle (transformation heartbeat).

Appendix 3: Agile Patterns of Representation

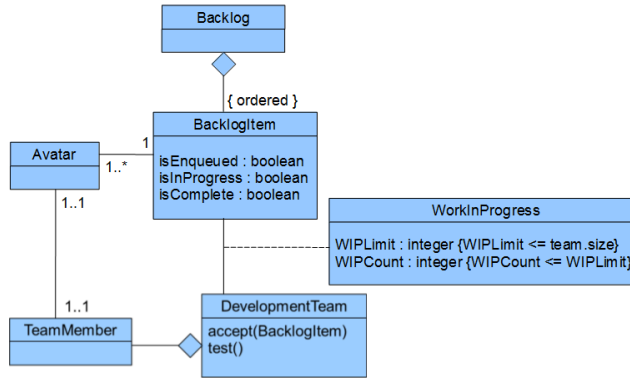
Avatar

Intent: Have a signal on an information radiator that indicates who is working on what

Proverbs:

- In a transparent world it is best to be transparent
- Honesty is the best policy

Avatar



Also Known As:

- Tag
- Icon

Motivation: Misunderstandings can arise as to who is currently responsible for work that is in progress. A means is needed for showing that work is actively being done and by whom. Queries about that work can then be directed to the appropriate team member. Conversely, work that is no longer clearly represented can be actioned and completed before additional work is drawn from a backlog.

Structure: Enqueued items are taken off the backlog by developers, at which point they become work in progress. A developer will immediately attach his or her avatar to the item to indicate that they are progressing it. Once the item is complete, they will remove their avatar from it, and are free at that point to progress another item from the backlog assuming that the Work In Progress limit would not be exceeded. Note that each developer has only one avatar, and can therefore only take off one backlog item at a time before completing it. The Work In Progress limit cannot therefore exceed the number of developers. Developers are allowed to collaborate on the same item; it is therefore permissible for an

item to have multiple avatars. No item that is in progress should ever be without an avatar.

Applicability: 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.

Consequences: Avatars reduce the potential for confusion in the team regarding who is progressing work. They also encourage the limitation of WIP since each developer only has one avatar to attach to an item.

Implementation: 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

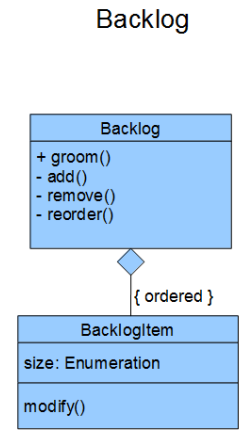
Intent: Enumerate all work to be done in the form of an ordered list

Proverbs:

- When everything is a priority, nothing is a priority
- First come, first served

Also Known As:

- Prioritized Requirements List



Motivation: 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.

Structure: A backlog will consist of multiple backlog items, all of which will be placed in order. Each item will be sufficiently clear and unambiguous to be given a size. Items can be added or removed from the backlog, or re-ordered within it, a process known as backlog grooming. Grooming also allows backlog items to be modified if necessary.

Applicability: Backlogs and the associated principles of queue management are applicable to all agile methods.

Consequences: 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. A less reactive management style is cultivated and "firefighting" is reduced. Backlogs require good management and clear ownership. Establishing this discipline can be a problem for many organizations.

Implementation: 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.

Epic

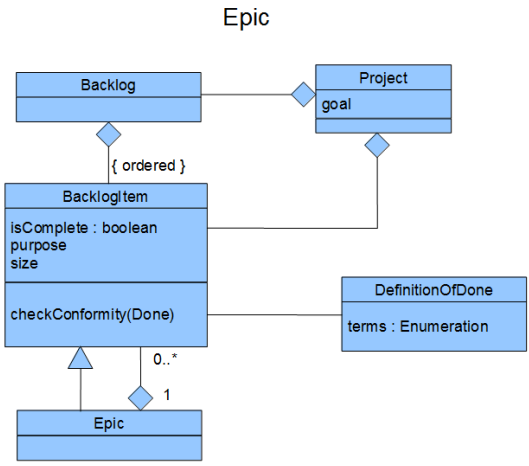
Intent: Allow certain items on a backlog to be jointly or provisionally represented in a grouped way

Proverbs:

- Enough is as good as a feast
- Half a loaf is better than no bread at all

Also Known As:

- Theme (although this is more commonly used to describe a grouping of epics)



Motivation: Requirements may represent business value without being detailed or focused enough to be actionable User Stories. A way is needed to capture this scope so that actionable stories can be represented, even though they may not yet have been written.

Structure: A project backlog will consist of multiple backlog items, some of which may be nested. An item that may potentially nest one or more other items that are related by some common purpose is termed an Epic. An Epic will be

complete when all of its member items conform to the terms of the Definition of Done. All backlog items, including Epics, must have a size.

Applicability: Epics are applicable whenever a grouping construct is needed for Product Backlog Items (PBI's). They may also be used as a placeholder for items that have not yet been identified, but for which an over-arching business purpose is clear.

Consequences: Epics can be difficult to size since important details may be absent. They represent scope that may be associated with significant risk. They must generally be broken down further before a team can accept the relevant work into a Sprint Backlog.

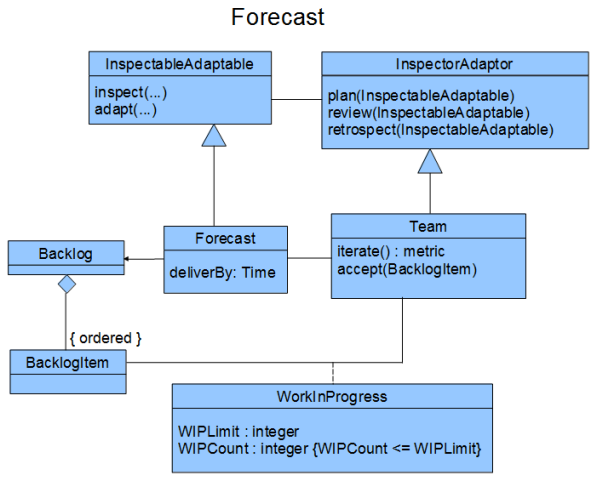
Implementation: Epics are widely used to represent User Stories at a high level in a Product Backlog. DSDM, Scrum, and XP commonly use epics as a placeholder grouping construct for undefined project requirements. Epics are less commonly associated with Lean-Kanban, at least in a Business As Usual context, since they do not align to the small and repeatable changes that typify such workstreams.

Forecast

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

Proverbs:

- If we are to understand the future we must look at the past
- Forewarned is forearmed



Motivation: 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.

Structure: A team has an ordered backlog of items to deliver. The team only allows a limited number of items to be work in progress at any one time. The

velocity or throughput can be calculated by inspection and a forecast made regarding the likely time of delivery for items remaining in the backlog. Metrics are captured iteratively on a timeboxed basis. The method used to derive these metrics and make such forecasts can be adapted by the team in order to improve their usefulness.

Applicability: Forecasting is used in most agile methods in order to determine the likely deliverables by the end of a given timebox.

Consequences: Forecasts can be mistaken for commitments and it is important to qualify expectations accordingly. The accuracy and value of forecasts diminishes the further into the future they extend.

Implementation: 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

Intent: Deliver a potentially releasable piece of work at the end of each iteration

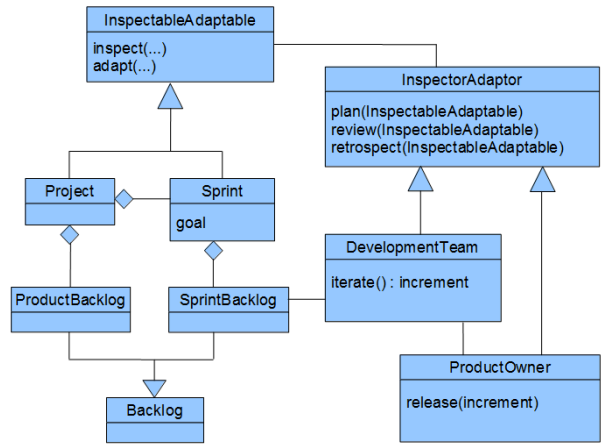
Proverbs:

- Actions speak louder than words
- The proof of the pudding is in the eating

Also Known As:

- Prototype (in the context of evolutionary prototyping)

Increment



Motivation: There is a risk that when a system is delivered it will not prove fit for purpose, or that certain features will be delivered too late for a maximum return on investment. This risk can be managed by the regular and timely delivery of system features that have been usefully grouped, and which are potentially releasable in themselves.

Structure: 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 which 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.

Applicability: 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. It is less appropriate for Business As Usual work consisting of discrete and isolated changes.

Consequences: Incremental release implies that work will be batched, and that each constituent item may not be delivered prior to that release. Items that could have been delivered earlier than the end of the Sprint will therefore have their return on investment delayed.

Implementation: Incremental release is a core feature of Scrum, XP, and DSDM. It is less commonly found in Lean Kanban and DevOps approaches, where continuous integration and deployment techniques are rather more likely to be in use.

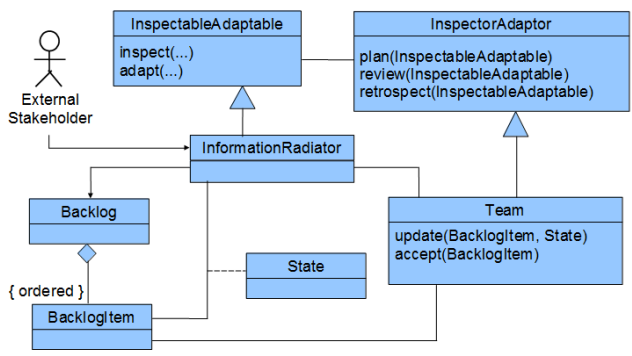
Information Radiator

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

Proverbs:

- Honesty is the best policy
- Tell the truth and shame the Devil
- Better to light a candle than to curse the darkness

Information Radiator



Also Known As:

- Andon

Motivation: 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.

Structure: A team will have an ordered backlog of work items. 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.

Applicability: Information radiators have their origins in Lean Kanban where "andon" signals are used to flag impediments and other status indicators to a team. They have subsequently found application by in agile methods and their use is widely accepted.

Consequences: Information radiators usually require team discipline if they are to be kept up-to-date. If they are not updated regularly, their value will decrease and team members will be even less likely to maintain them.

Implementation: The most common information radiator is a Kanban board or task board. Variants of these can be found in use by most agile teams.

Metrics

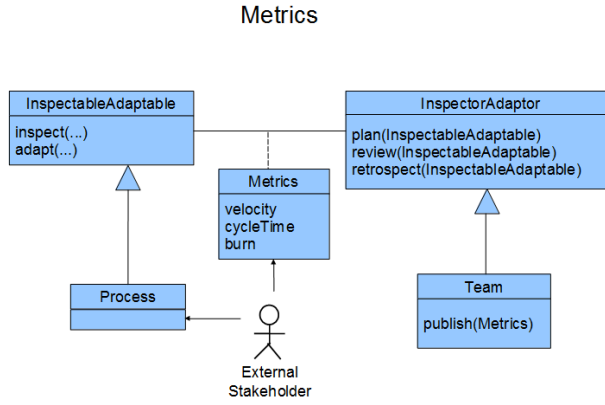
Intent: Provide non-subjective data so that informed decisions can be made

Proverbs:

- You can't manage what you don't measure

Also Known As:

- Stats



Motivation: Decisions made on the basis of

subjective judgment, and which do not involve any objective measurement, are likely to be poor. It is important that a team has an objective means to gather quantitative data about its process so it can inspect, adapt, and provide meaningful forecasts.

Structure: A team will publish its own metrics, including velocity, cycle time, and burn rate. These metrics are used to help them inspect and adapt their process. The metrics are visible to external stakeholders in the process who can then make appropriate forecasts.

Applicability: Metrics are important to the inspect-and-adapt capability of all agile teams, and to stakeholders who may wish to make forecasts based on that data.

Consequences: Knowing what to measure is important. Certain objective measures are not always useful or fit for purpose. Vanity metrics are measurements which show a situation in an unreasonably favorable light, even though the measurements themselves may well be accurate.

Implementation: Metrics can be derived from forecasts (such as estimates) or actuals (such as number of items actioned in a timebox). Scrum teams typically provide a Sprint Burndown of estimated hours remaining, and a Product Burndown of estimated User Story points remaining. Kanban teams will often provide metrics in the form of actuals only, such as cycle time.

Minimum Viable Product

Intent: 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

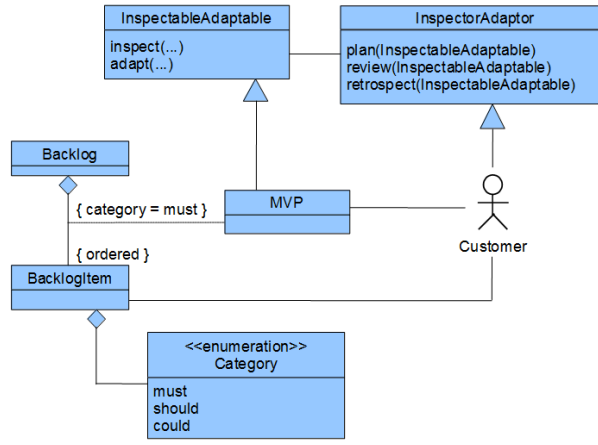
Proverbs:

- You Ain't Gonna Need It
- Keep It Simple Stupid (KISS)
- Test the water first

Also Known As:

- Minimum Usable Subset
- Must-Haves

Minimum Viable Product



Motivation: There are two separate and incompatible motives behind the delivery of Minimum Viable Product (MVP). Firstly, there is the desire to have an early Return On Investment (ROI). A complete product does not necessarily have to be delivered in order to be usable. A minimum set of features may prove sufficient to establish an adequate revenue stream. Alternatively, there may be a desire to test a hypothesis about the business environment or market. By delivering a minimum set of features that allows the hypothesis to be tested, lessons can be validated and wasted effort can be minimized. The establishment of an early ROI should not be a critical outcome if the testing of such an hypothesis has priority.

Structure: A backlog will consist of multiple items that are placed in a certain order for delivery. In addition to its order in the backlog, each item will be categorized in terms of its importance as a must-have, should-have, or could-have requirement. The must-have requirements will represent the Minimum Usable Subset of items needed for the product to either harvest a suitable Return On Investment from customers, or to allow important lessons to be learned about the customer market.

Applicability: The delivery of a Minimum Viable Product is appropriate when the provision of a subset of product features would be useful in its own right.

Consequences: The provision of a Minimum Viable Product is time constrained. Either a Return On Investment must be harvested as soon as possible, or a hypothesis must be tested and lessons learned before time and effort is wasted. By definition, the constituent items in a Minimum Viable Product are all must-have requirements. As such there can be no flexibility of scope. The risk of not delivering an MVP within a specified timebox is therefore high.

Implementation: Minimum Viable Product is not supported in Lean Kanban implementations except in the context of testing a hypothesis about the business environment or market, such as split A/B testing. An MVP representing collated requirements for a minimal marketable product is less viable where continuous integration and deployment is in use, and where each completed item is promoted to live without batching or staging. Minimum Viable Product is not recognized in Scrum, although a Development Team and a Product Owner may still plan for the delivery of a such an increment. DSDM recognizes must-have, should-have, and could-have requirements, and the delivery of MVP (called Minimum Usable Subset) is expressly supported.

Proxy Product Ownership

Intent: Authorize a suitable stand-in should a Product Owner be indisposed

Proverbs:

Proxy Product Ownership

- There are no small parts, only small actors

Also Known As:

- Stand-in

Motivation: A Product Owner will often have multiple responsibilities and can be accountable for many products. He or she may not have the time to represent a product to a Development Team and might not be co-located with them. A proxy is needed who can represent product ownership adequately on the PO's behalf, and liaise with a Development Team on an operational day-to-day basis.

Structure: A Product Owner authorizes one or more Proxies to represent product ownership to one or more Development Teams. Each Development Team will only have one Proxy, such that product ownership is unambiguously represented. Each Proxy will be able to explain and prioritize the items on the Product Backlog to a Development Team, to negotiate a Sprint Backlog with them, to review, accept, and release the increments produced every iteration, and to liaise with the Product Owner as necessary and in a timely manner.

Applicability: Proxying is useful when product ownership is clear, but the Product Owner is not able to represent the product at an operational level. Proxying cannot be done for a notional or unclear Product Owner, or when product ownership is weak, divided, or absent.

Consequences: Proxying allows a Development Team to deliver meaningful increments of value without the direct involvement of the Product Owner, who is thereby freed for other activities. Note that the Product Owner remains fully accountable for the product, and for its Return On Investment, even when a Proxy is in place. The Product Owner will need to liaise closely with his or her Proxies in order to ensure that product interests are competently represented. Note also that although proxying is a viable model, it is difficult to implement well and is often associated with certain antipatterns (e.g. Too Busy, Unresolved Proxy). There is a further danger that proxying will degenerate into product ownership by committee, the symptoms of which are weak decision making and unclear accountability.

Implementation: In Extreme Programming, the Customer Representative may effectively represent product ownership by Proxy. In DSDM, the Business Ambassador, Business Visionary, Business Analyst, and Business Advisor each proxy in certain respective matters for the Business Owner. Scrum does not address the matter of product ownership by Proxy.

Relative Sizing

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

Proverbs:

- Acorns compare their height with each other
- Everything is relative

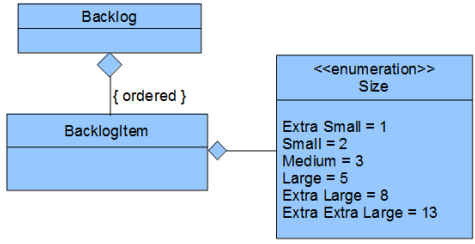
Also Known As:

- Relative ordering

Motivation: Time-based estimates are unreliable, and are apt to become even more so at larger magnitudes. A way to size backlog items is needed that is independent of time or other absolute measures.

Structure: A backlog consists of an ordered list of items. Each item has a size. The size will be expressed in relative terms. These terms may be descriptive, such as small, medium or large. Arbitrary numbers that do not reflect any particular unit of measure can be mapped to the terms being used. Items are compared to each other; i.e. this is bigger than that, or smaller, or roughly the same size, and are thereby given a size in the scheme.

Relative Sizing



Applicability: Relative sizing is useful when teams are unable or unwilling to create time-based estimates.

Consequences: Relative sizing can be a difficult concept for teams to understand, as there is no time-based unit of measure. “Relative effort” is often the best way of explaining how items should be compared. Understanding how to size using arbitrary numbers often proves difficult, as there is a common prejudice towards mapping numbers to hours or days. For this reason it is often best to use descriptive terms alone (e.g. small, medium, large).

Implementation: 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.

Value Stream

Intent: Understand the value being added at each stage of a process with a view to eliminating waste

Proverbs:

- There is nothing quite so useless as doing with great efficiency that which should not be done at all
- More haste less speed
- Efficiency is intelligent laziness

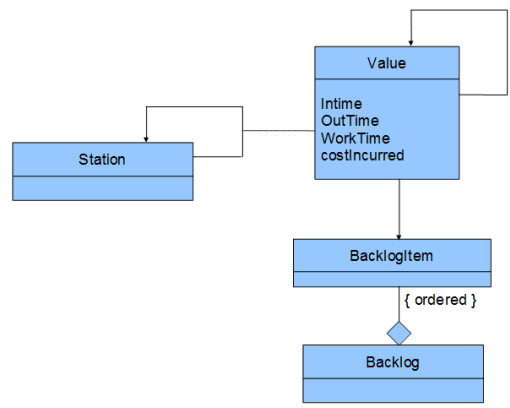
Also Known As:

- Value Chain
- Workflow

Motivation: Wasted effort is often hidden in established working practices. Transparency is needed. It has to be clear if the backlog of work is being turned into valuable product in the most efficient way possible, and where any blockages lie.

Structure: A backlog consists of many items which are placed in order. Items are removed from the backlog and progressed through various stations of work, and a stream of accumulating value is thus created. However, the time spent working on an item in each station in the value stream will incur a cost. These costs subtract from the value and are open to inspection.

Value Stream



Applicability: The modeling of a value stream is applicable to all industrial sectors that face competitive or regulatory pressures for greater efficiency.

Consequences: Understanding the value stream highlights occurrences of waste, such as bottlenecks or the introduction of unplanned work. There can be political repercussions associated with waste removal, as the practices that lead to it may be culturally normed. Waste often accumulates around the edges of linked value streams, such as the outputs and inputs between collaborating teams.

Implementation: A team's value stream is often represented as a Kanban or Scrum board. Bottlenecks and other impediments are thereby made transparent, and the stream can be inspected and adapted by the team. Work In Progress is often limited in order to reduce stock-on-hand and to improve throughput.

Appendix 4: A Demonology of Agile Antipatterns

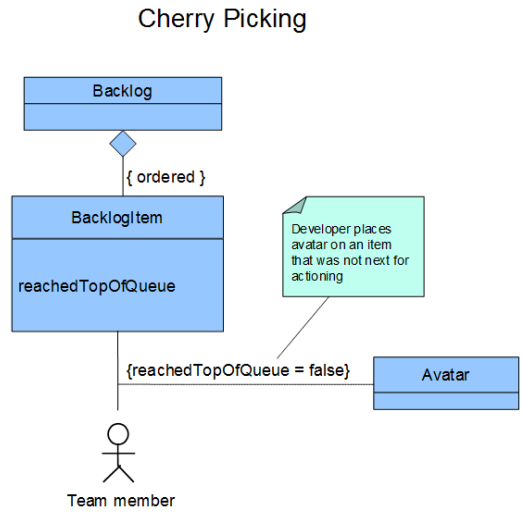
Cherry Picking

Intent: Select the most desirable item from a backlog without reference to priority or plan

Proverbs: Glory built on selfish principles is shame and guilt

Motivation: Team members are often tempted to select those items from a backlog which they expect to be the most gratifying for them, or the easiest to do.

Structure: A team member lays claim to an item that was not the highest priority for actioning. The claim is made by placing his or her avatar on the item. They may assert ownership either while the item is still in the backlog or by bringing it into work in progress.



Applicability: Cherry picking often occurs in teams where the members are not cross-trained. Members may regard certain items as “their” work rather than the team’s work, and try to lay claim to them accordingly. Backlogs items that have been insufficiently groomed for actioning may be avoided by team members who attempt to cherry-pick others.

Consequences: If cherry-picking is left unaddressed, it will reinforce skill silos and reduce team cohesion, cross-functionality, and trust.

Implementation: Cherry picking can occur in Kanban. There is a danger of confusing this anti-pattern with Quality of Service, since different terms of service can imply that certain backlog items are preferentially treated. Cherry picking can also occur in Scrum, where the selection of an item by a developer is not in accordance with the team’s Sprint Plan. This anti-pattern is associated less with

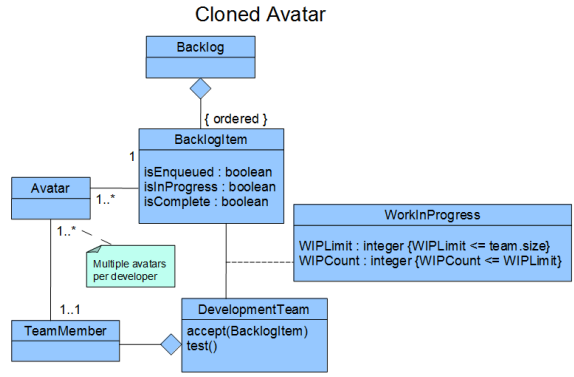
XP since pair programming limits the opportunity for individual developers to cherry pick.

Cloned Avatar

Intent: Increase the Work In Progress beyond 1 item per team member

Proverbs: You can't be in two places at once; Good and quickly seldom meet

Also Known As: Split Avatar



Motivation: 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. They can then claim to those exerting the pressure that work is underway.

Structure: A Development Team member accepts a backlog item on behalf of the team, and thereby brings it into Work In Progress. His or her avatar is placed on the item to show that responsibility has been taken for its progression. Before the item is complete, another enqueued item is taken off the backlog and brought into progress. The team member then either places a duplicate avatar on the work (cloned avatar) or spans both items with a single avatar (split avatar).

Applicability: Avatar cloning or splitting is common where product ownership is poor. A good product owner establishes clear priorities . Pull will be exerted on work that is currently in progress rather than unstarted work. As such, team members will neither be expected nor encouraged to break WIP limits.

Consequences: When team members clone or split their avatars they are typically breaking WIP limits. Each team member should only have one avatar and as such the team WIP limit should be no greater than the number of team members. Exceeding the WIP limit leads to an increase in batch sizes, reduced throughput, and an increased depreciation of work partially completed.

Implementation: It is common to find an avatar straddling two or more items in progress on a Scrum or Kanban board, or for items to be placed in proximity to one avatar so as to imply simultaneous progression (split avatar). Some teams

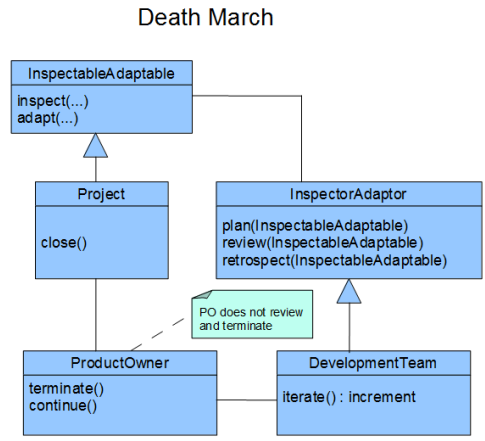
normalize this dysfunction by mistakenly allowing team members more than one avatar (cloned avatar).

Death March

Intent: Continue with a failing project so as to defer its outcome

Proverbs: Kick the can further down the road; hold tight and pretend it's a plan; the Emperor has no clothes

Also Known As: Escalation of Commitment



Motivation: It can be politically difficult to cancel a struggling initiative if it has already absorbed significant resource. 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.

Structure: A development team notionally delivers increments of value to a Product Owner on an iterative basis. The development process and project are transparent and are theoretically subject to inspection and adaptation. However, for political reasons the Product Owner cannot terminate the project even though satisfactory value is not being delivered. Instead, the project continues until either the expected delivery date or resources run out.

Applicability: Death march projects tend to be high profile and high risk, and are associated with weak product ownership. The risks will not be managed by the iterative and incremental delivery of value. Instead, any increments produced will accumulate. The associated risk will also accumulate until it is too late for mitigation.

Consequences: A death march project will severely impact team morale and staff turnover is likely to be high. Resources will be wasted on a failed project, when it would have been better to cut losses earlier and fund alternative investments.

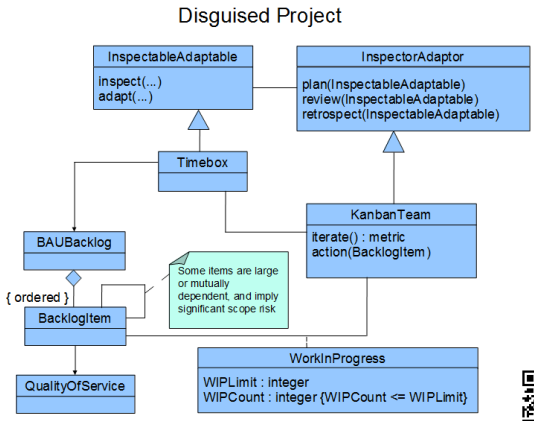
Implementation: Death marches are very common in the public sector due to a weak understanding of agile practice, cultural barriers against iterative and incremental release, and the fear of losing face as a result of early project

termination. They may also occur in the private sector if project investments are similarly politicized, or if ongoing delivery and the de-risking of scope is similarly undervalued.

Disguised Project

Intent: Try to get substantial work done without having to set up and fund a project

Proverbs: If the Devil is going to disguise himself, it will be as a monk or a lawyer



Also Known As: Scope Creep (in a Business As Usual context)

Motivation: 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.

Structure: A Kanban Team will action items that are drawn from an ordered backlog, each of which may require a certain quality of service. Work In Progress limits will be observed, and metrics will be used to inspect and adapt the team’s working practices on an iterative and time-boxed basis. The backlog is expected to consist of “Business As Usual” changes that are small and repeatable in nature. However, large or mutually dependent work items will have been placed on it that do not represent BAU work, and which are associated with significant scope risk.

Applicability: Disguised projects are commonly found where there is a separation between operational and capital budgets. It is also found where there is inadequate control over the parameters of BAU work, or over the threshold that must be reached to trigger project inception.

Consequences: A disguised project will compromise the ability of BAU teams to handle genuine “Business As Usual” work. Throughput will be reduced due to the size and scale of the work items, and the associated scope risk is likely to be hidden.

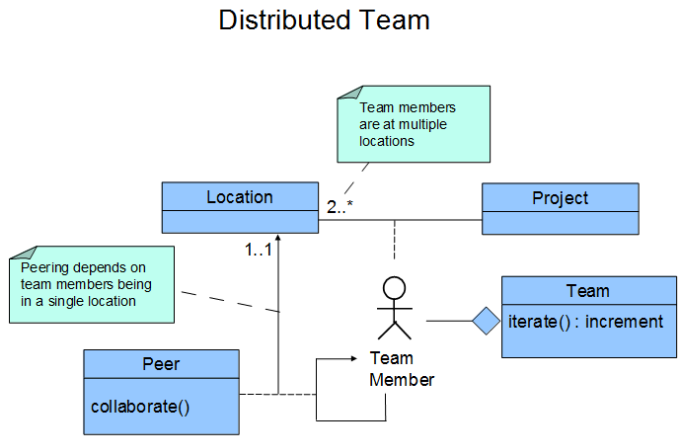
Implementation: Business As Usual work is typically handled by Kanban teams, and as such they are most likely to encounter disguised projects on their backlog. Scrum teams tend to be organized in terms of project delivery and so this work is unlikely to be disguised. Scrumban teams might be aligned to a project while simultaneously providing BAU support, hence they are also at risk of encountering disguised projects on the BAU stack.

Distributed Team

Intent: Claim the benefits of agile practice without co-locating team members

Proverbs: United we stand, divided we fall; 90% of communication is non-verbal

Also Known As: Non co-location; Split Team; Dislocated Team



Motivation: 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.

Structure: A team will try to release project increments on an iterative basis. However, the team members will be spread across multiple locations, and peer collaboration will be correspondingly impaired.

Applicability: Teams in large organizations are particularly susceptible to having non-colocated members. This is because such bodies are often geographically distributed themselves or outsource work to distant third parties.

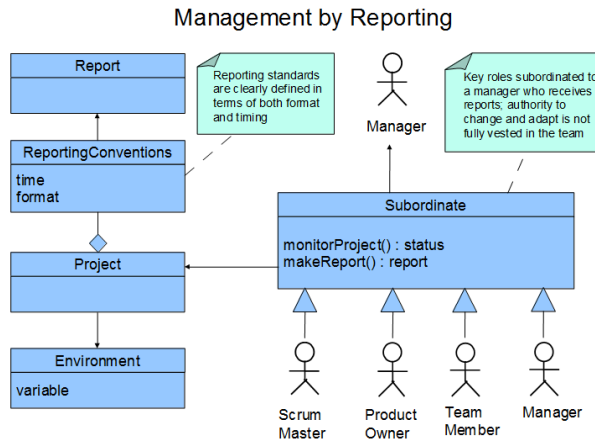
Consequences: Agile practice is heavily dependent on individuals and their ability to interact with each other. Teams with low WIP limits are particularly impacted by the effects of distributed membership, as each item in progress will require collaborative actioning. There is an increased risk of skill silos emerging as a result of non-colocation, since the opportunities for pairing and cross-training will be reduced.

Implementation: Scrum teams may have some of their functions outsourced or even offshored. Testing is one common example.

Management by Reporting

Intent: Limit liability by restricting any action taken to the official communication of status

Proverbs: The only thing necessary for the triumph of evil is that good men should do nothing; If you want to kill any idea in the world, get a committee working on it



Also Known As: Management by Committee (in the context where multiple parties are involved)

Motivation: Highlight issues and concerns, while deferring the risk of taking action to others…or at least to a later date. The hope is to avoid personal exposure should there be negative consequences from taking action.

Structure: A manager monitors a project and its environment, and reports on status to another management authority in accordance with established reporting conventions.

Applicability: This anti-pattern is common in bureaucracies with a stage-gated culture or process, especially those that are heavy with middle management. It

can often be found in large organizations that are transitioning to agile ways of working.

Consequences: Action should be taken as close as possible to the time and place of a triggering event…i.e.by team members. Failure to do this results in delay and increases the risk of either inappropriate action or no action at all. If team members are insufficiently empowered (or insufficiently trained) to inspect and adapt the team process, then the associated risks will be high. Management by Reporting is a contraindication to Agile and Lean practice.

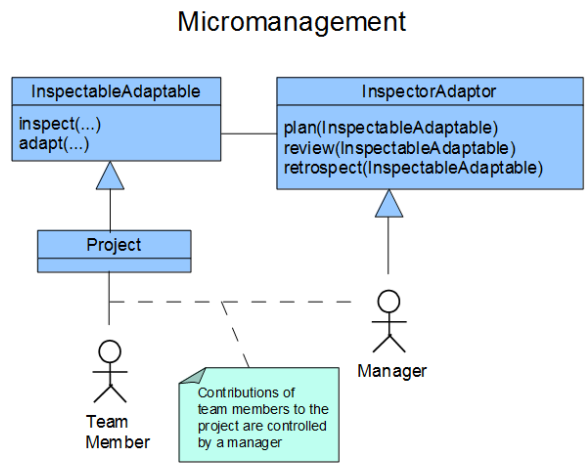
Implementation: The most common implementation of this anti-pattern is a vestigial Project Manager. Typically repositioned as an adjunct to an agile team and as gatekeepers to a legacy stage-gated process, such managers may relay or transcribe the status that has been communicated by Scrum Masters and/or Product Owners.

Micromanagement

Intent: Manage the details of a project directly rather than through the offices of the appropriate team

Proverbs: You have to know when to let go; lack of accountability equates to lack of trust

Also Known As: Control Freakery



Motivation: 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.

Structure: A team member's involvement with a project is controlled by a manager, such that the member has little or no autonomy. The ability of the team member to inspect and adapt working practices is compromised where authority in operational matters is vested in the manager.

Applicability: Micromanagement is a common issue with managers who have been promoted from the field, and who wish to retain a degree of operational control. This dysfunction can also affect managers with a vision or strategy about which they feel strongly, and who exert an inappropriate amount of oversight on operational details.

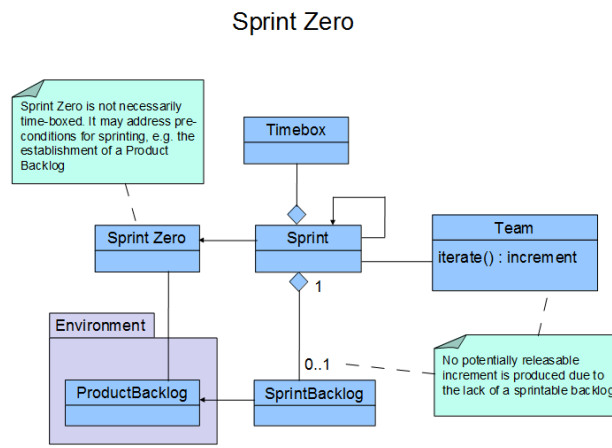
Consequences: Micromanagement creates a bottleneck in decision making and leads to waste being incurred. Actions may not be taken in a timely manner and could be absent altogether, poorly-informed, or rushed. Since inspection and adaptation of the team's process is not vested in them, their accountability and responsibilities within the value chain will be compromised. Note that micromanagers can be quite selective in the concerns they choose to interfere with, and may continue to delegate matters they do not wish to address. Micromanagers who selectively interfere with people's work are likely to create inconsistencies in quality, and to infuriate and alienate the affected team members.

Implementation: Organizations with a command-and-control management culture often suffer from micromanagement. This problem can be carried into an agile transformation. As such, the implementation of servant-leadership often requires that the micromanagement anti-pattern be eliminated first.

Sprint Zero

Intent: Set up an agile project, while contextualizing unplanned initialization overheads in agile terms.

Proverbs: The first step is always the hardest



Motivation: Agile methods typically have an operational delivery focus. They do not always address strategic or tactical matters such as the resourcing and initialization of projects. This deficit may go unnoticed by agile development teams which have a duty to deliver value as quickly as possible. As such, these teams are often tempted to start iterative and incremental development before business or other important stakeholders can engage in the value stream. When the first

sprint fails to deliver business value, it is then contextualized and dismissed as “Sprint Zero”.

Structure: A team attempts to sprint. However there is no Product Backlog from which a Sprint Backlog can be drawn and completed within the associated time-box. Other significant elements of the project environment may also be missing. This first attempt at an iteration does not therefore result in a potentially releasable increment of value. Work done by the team in trying to overcome these matters is correlated to a so-called Sprint Zero. This pseudo-sprint does not necessarily match the original time-box, since the initialization work is unplanned and unsized.

Applicability: This anti-pattern is often found in organizations where product ownership is weak and/or portfolio management is poor. The assignment of teams to a project will not be synchronized with other elements of project initiation, such as the resourcing of a Product Owner, the drafting of a backlog, the availability of development or test environments, or the approval of a project vision or business case.

Consequences: A poorly controlled project may commence with an ill-defined Sprint Backlog or unforeseen dependencies, and the team will need to resolve these impediments before value can be delivered. Most or all of this work may be unforeseen and unplanned. Retroactively fitting such project initiation tasks into a time-box may create the illusion of control, but no value will have been delivered in this pseudo-Sprint and any associated metrics are likely to be deceiving. Also, the wrapping of work post-facto does not represent the genuine time-boxing of activities, and can indicate a poor grasp of agile practice by the stakeholders concerned.

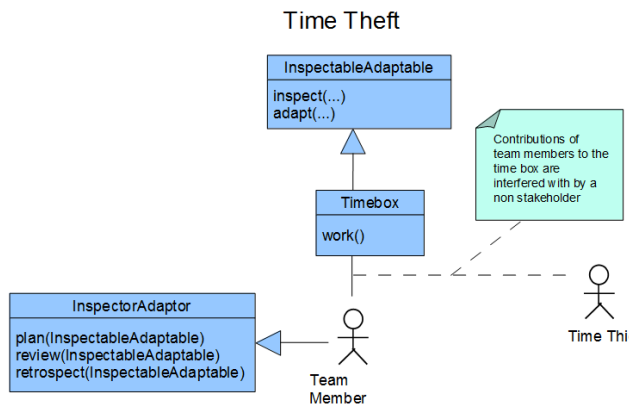
Implementation: Sprint Zero is not recognized as a tenable construct in any agile methodology, although the anti-pattern is frequently encountered and is commonly assumed to be a valid practice.

Time Theft

Intent: Coerce third parties into assisting you without negotiating their time

Proverbs: Time is the coin of your life, be careful lest others spend it for you

Also Known As: Waste (in the context of unplanned work)



Motivation: 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.

Structure: A team member plans to do timeboxed work. All work completed within the timebox is inspectable and subject to review. A time thief approaches the team member during the timebox, involves them in unrelated activities, and thereby interferes with the team member's ability to complete the work satisfactorily in the time available.

Applicability: Time theft can happen in environments where stakeholders are used to approaching team members directly, or where product ownership is weak and backlogs are not rigorously managed.

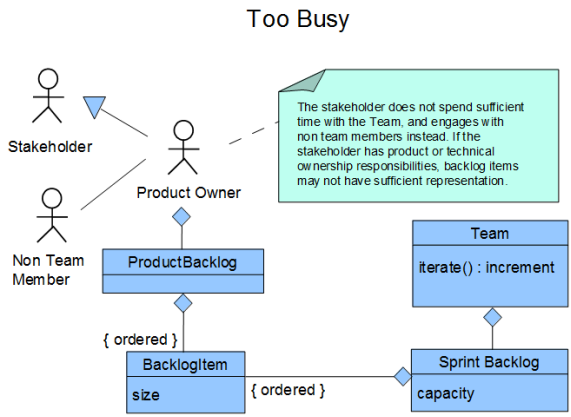
Consequences: Time theft reduces the ability of teams to deliver planned work in a Sprint or other timebox. Incremental releases will be put at risk. Inspection and adaptation of the team process may be compromised if time theft is unrecorded.

Implementation: Agile methods such as Scrum reduce the opportunities for time theft by introducing servant leadership roles such as a Scrum Master. The person fulfilling such a role must protect the team against unwarranted interference and coach its members to beware of time thieves. It should be noted that the incentive to commit time theft can only be eliminated once all parts of an organization subscribe to agile practices, including backlog management. Teams must also have a shared view of release planning so that they can manage their dependencies in a timely fashion.

Too Busy

Intent: Ignore your duties as a stakeholder in order to perform other activities that you consider more pressing

Proverbs: It's not about having time, it's about making time; if you don't make time for your wife, someone else will



Also Known As: Absentee Product Owner, Assigned Product Owner

Motivation: 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.

Structure: A stakeholder in the product (such as a Product Owner or architect) fails to liaise with the team at a critical time. During an iteration in which one or more of items with a stakeholder dependency are drawn from the Product Backlog and planned into the team's Sprint Backlog, the stakeholder's input will be needed so that the increment can be delivered. However, the stakeholder abdicates this responsibility in order to perform unrelated activities.

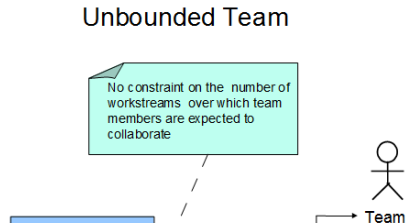
Applicability: Clear product ownership (and sometimes technical direction) can be difficult to establish, especially when products or services affect multiple stakeholders or cross organizational boundaries (e.g. middleware components). Each stakeholder may believe they have higher priorities outside of the team and its work. The strategic importance of the product or service may only be apparent if such weak stakeholders are viewed in aggregate. This can result in product ownership and/or architectural duties being assigned to managers who do not have a sufficient level of individual interest in the outcome, and who consequently do not afford their role the time it deserves.

Consequences: If the stakeholder does not provide the assistance expected, dependent backlog items may not be completed as per their acceptance criteria. The Sprint Goal may be put at risk and an increment of functionality might not be delivered. Alternatively, the team may try to negotiate the impediment by consulting with other stakeholders instead. The interests of such stakeholders may not coincide with those of the absentee, whose own interests might thus become compromised.

Implementation: In many organizations product ownership or architectural authority is vested in senior managers. These managers can be stakeholders in multiple concerns that are in progress or being planned. They may not be sufficiently interested in a particular project to afford it the time that the role demands.

Unbounded Team

Intent: Use team members for work that is outside of their remit or declared purpose



Proverbs: What we imagine is order is merely the prevailing form of chaos

Also Known As: Firefighting

Motivation: 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.

Structure: A team member is assigned to a workstream but there is no limit to the number of assignments that can be made. Members can thus be expected to work on multiple concerns and this can include both project and business-as-usual work.

Applicability: Most agile methods, including Scrum, do not constrain team members to single workstreams. To this extent the teams are susceptible to becoming unbounded. It should be noted however that agile teams are self-organizing. Self-organization restricts the ability of managers to assign members to different workstreams in an arbitrary manner.

Consequences: Unbounded teams make commitment-based planning difficult or impossible. Team membership is unclear, and the time that members can spend on a particular workstream may be unreliable. Sprint goals can be compromised and the quality of forecasting will be poor. The morale of the team will suffer as a result of poor cohesion, a failure to deliver, or an inadequate sense of purpose. Inefficiencies due to task-switching are often a symptom of the unbounded team problem. The reassignment of team members by external authorities implies that the team is not self-organizing and therefore not agile. Reassignment of team members to multiple workstreams can also imply that portfolio and program level management is unfit for purpose.

Implementation: The Unbounded Team antipattern often occurs when organizations succumb to firefighting. Firefighting happens when inappropriate plans are made or the organizational response to change is poorly managed. The problem is also associated with reactive management styles and with incompetent managers who commit team members without appropriate consultation.

Unbounded Timebox

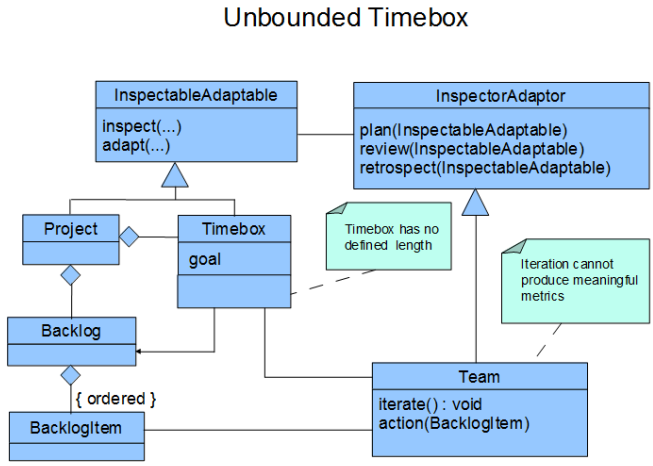
Intent: Allow an indeterminate amount of time for the completion of a task

Proverbs: To protract a great design is often to ruin it; Procrastination is the art of keeping up with yesterday

Motivation: The time-boxing of incremental delivery requires the specification of intermediate goals. Time-boxes should be of the same length so that progress can be assessed regularly on a goal-by-goal basis. However, teams can be tempted to try and extend an available time-box if, in so doing, it 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.

Structure: A team plans to action certain items from an ordered backlog, and to do so within a time-boxed project iteration. However, inspection of the time-box shows that the goal is unlikely to be met. The time-box is then revised to be of indeterminate length such that it will only complete once the intended goal has been completed.

Applicability: All agile methods including Scrum, DSDM, and XP mandate that time-boxes be of a fixed length. They cannot be extended. Any project in which time-boxes have become unbounded is not being run in an agile manner.



Consequences: When a time-box is unbounded the ability to inspect and adapt the delivery process is compromised. Reviews, retrospectives, and future planning sessions will be deferred until an arbitrary point when the time-box completes. Useful and comparable metrics will no longer be produced iteration by iteration. Progress towards the completion of a project can no longer be gauged in such conditions and risks will be difficult to assess. Incremental deliveries will not occur as forecast, and an early return on investment will not be possible.

Implementation: Unbounded time-boxes tend to occur when product ownership is weak and insufficient pull is being exerted upon a team and their backlog for the regular delivery of valuable increments.

Uncommitment

Intent: Allow unplanned work into a timebox.

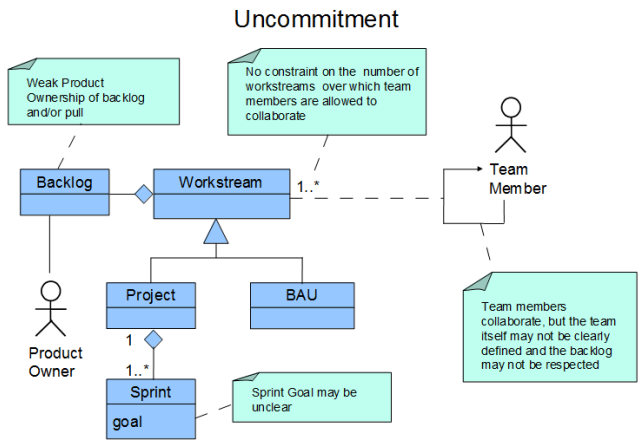
Proverbs: The sluggard who does not sow in season does not reap at the harvest

Motivation: 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. Product Owners who do not value the Product or Sprint Goal can themselves be tempted to trade unplanned work into a Sprint Backlog to the detriment of value produced.

Structure: Team members collaborate with each other in a workstream. The workstream may encompass either Business As Usual activities or project work delivered in multiple sprints with sprint goals. However, product ownership will be weak and any sprint goals may be unclear. Team membership may be ill-defined and/or the backlog may not be respected by them.

Applicability: Clear goals and the careful management of time-boxes are central to agile methods such as Scrum and DSDM. Business As Usual work can encourage uncommitment in so far as there may be no time-boxed goal to commit to. As such, Kanban teams can face unique challenges regarding product ownership, the replanning of work, and team commitment to any forecasts made.

Consequences: A Product Owner who does not value the product will have little incentive to help teams formulate Sprint goals, or to exert pull on a team for delivery. Team purpose and/or discipline will therefore be compromised. Sprint Goals may not be met, or the value delivered may be less than that expected.



Implementation: Uncommitment can take either of two forms: uncommitment by the team to a forecast backlog of work, or uncommitment by a product owner to the product being delivered. The result is the same in either case: the planned backlog will be compromised and the expected value is unlikely to be delivered.

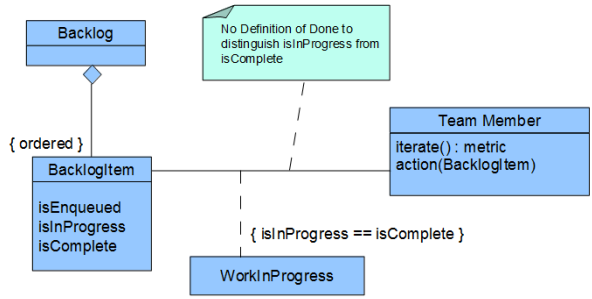
Undefined Done

Intent: Progress work as quickly as possible and without sufficient regard for quality

Proverbs: Nothing is done while something remains undone

Motivation: Establishing a Definition of Done formalizes the means of by which work is accepted. Customers may have little incentive to formally accept work as being complete, as to do would give them no advantage while also reducing their opportunity to demand rework, or to extend the original scope under the guise of rework. Some development team members may, for their part, object to the improved transparency that a Definition of Done brings to their working practices.

Undefined Done



Structure: Team members iterate over an ordered backlog of items. However, there is no clear separation between work that is in progress and that which is complete. Multiple items are actioned simultaneously and WIP is not limited. Customers may be partially in receipt of some work that is in progress.

Applicability: Authoring a Definition of Done is particularly hard when the type of work varies, such as in technical support. However, a specification of “done” is still possible if it is expressed in terms of review and signoff. Customers may face penalties or restrictions if they fail to approve work in a timely manner.

Consequences: Pull is eliminated and throughput is reduced. Incremental releases cannot be made as it is not know what work is “done”. Metrics cannot be elicited, and it will be unclear if Sprint Goals are being met.

Implementation: Scrum expressly requires a Definition of Done to be in place, although other agile methods may be less rigorous in this matter.

Unlimited WIP

Intent: Start new work before current work is completed

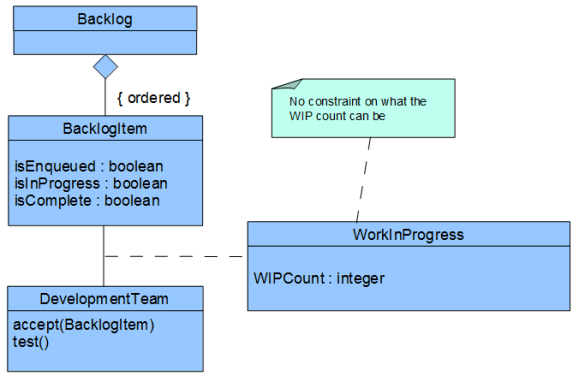
Proverbs: More haste less speed

Also Known As: Unbounded Commitment

Motivation: When faced by 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.

Structure: A development team accept items that have been enqueued in an ordered backlog. There is no limit to the number of items that are allowed to become work in progress.

Unlimited WIP



Applicability: Unlimited WIP is common in reactive environments where product ownership is weak and/or scattered, such that no effective backlog prioritization occurs.

Consequences:Having no limit on inventory increases batch sizes and reduces the prospects of incremental release. Work depreciates while it remains in progress, and the timely delivery of value is compromised.

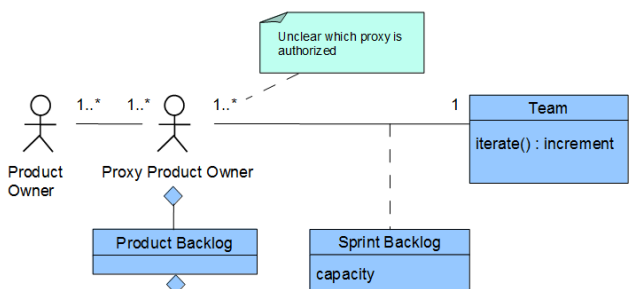
Implementation: Lean Kanban methods explicitly limit Work In Progress. Note that WIP limits may be applied to each Kanban states where work may become enqueued. Scrum does not strictly require WIP limits to be enforced as batch sizes are constrained in the form of Sprint Backlogs. Unlimited WIP is discouraged in all agile methods.

Unresolved Proxy

Intent: Conceal weak Product Ownership by distributing it

Proverbs: That which is owned by everyone ends up being owned by no-one

Unresolved Proxy



Motivation: 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.

Structure: A development team tries to negotiate a Sprint Backlog with a Product Owner. However, the PO is represented by multiple proxies and it is unclear which one has authority over the Product Backlog and the backlog items which are to be negotiated.

Applicability: Unresolved proxies can be found where Product Owners do not exercise authority or control over the Product Backlog or how it is represented. They are especially common in projects where multiple stakeholders consume a service, such as that which may be provided by shared components or middleware.

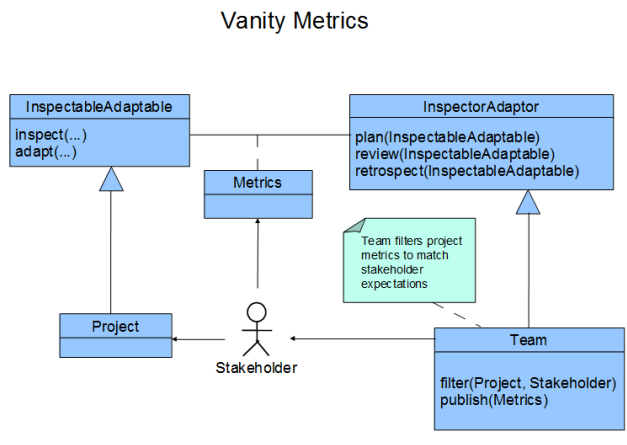
Consequences: Sprint Planning will be compromised as it is unclear which proxy represents the Product Owner and has authority to negotiate a Sprint Backlog. Increments are likely to be of low quality if the team do not know who to consult with regarding the approval of work. The Definition of Done may be unclear.

Implementation: Scrum does not provide for proxy product ownership, although it is widely used and misapplied in a Scrum context. DSDM provides for several business roles which can reduce to unresolved ownership of a backlog (Prioritized Requirements List) unless discipline is maintained. XP mandates that the team liaise with a clear customer role and hence proxying is effectively forbidden.

Vanity Metrics

Intent: Use only the most favorable metrics in order to strike a posture

Proverbs: Fine words butter no parsnips



Motivation: Team members can be tempted to show their efforts in the best light rather than in a more objective one. They may cherry-pick data in support of this position so as to encourage the further

commitment of stakeholders.

Structure: A team review the progress of a project they are working on by inspecting the metrics they have gathered. They then filter these metrics down to a reduced data set which gives the most positive interpretation of project status to stakeholders.

Applicability: Vanity metrics can be used to entice third party investors. They are also often used within organizations to justify projects, and to protect and extend budgets.

Consequences: If vanity metrics are used to influence decision making, then those decisions will be taken with an unrepresentative view of the real status of a project. Any forecasts based on this information may be unfit for purpose, and the opportunity to secure advance warning of any problems may be lost.

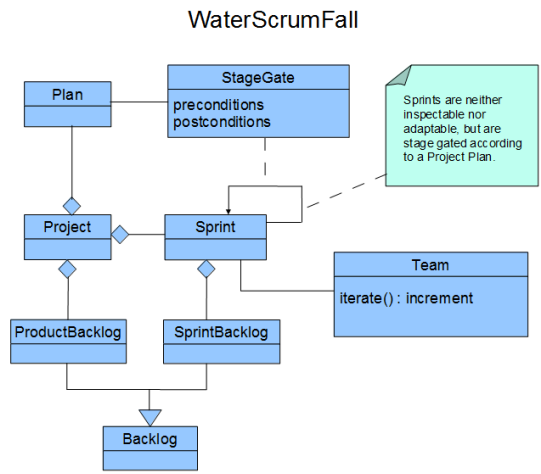
Implementation: Vanity Metrics are explicitly dealt with in the Lean Startup approach and the use of actionable metrics is favored.

WaterScrumFall

Intent: Attempt to leverage Scrum benefits while retaining a stage gated approach

Proverbs: A dog cannot serve two masters; cr*p or get off the pot

Also Known As: Scrum-But, Agile In Name Only (AINO)



Motivation: 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 effect a compromise in which an iterative development approach is encapsulated within a development stage. In so doing they hope to leverage the benefits of agile practice, whilst not actually changing the organization's delivery approach or the terms of reference that are comfortable to stakeholders.

Structure: A team attempts to sprint in order to produce increments of work that are drawn iteratively from a Product Backlog. However, the sprints are

constrained by the pre and post conditions of stage gates that are determined by a project plan, and the value delivered is correspondingly reduced.

Applicability: Waterscrumfall is widely used to constrain agile practice to a development stage. Requirements analysis and delivery are held to be outside of the remit of agile teams.

Consequences: The stage gated culture of an organization remains unchanged. Definitions of Done are weak because they do not cover the value stream through to release. Potentially releasable increments of value are not delivered and work continues to be batched into large shipments. The ability to inspect and adapt the engineering process is compromised due to delayed feedback. The potential to gain from early returns on investment is lost. Reputational damage can occur to agile practice since stakeholders may believe, incorrectly, that agile methods are in fact being applied.

Implementation: This anti-pattern has been widely implemented across the public sector, but it can also be found in commercial organizations, particularly those at enterprise scale. Various agile scaling frameworks (e.g. Agility Path) have been developed in an attempt to provide remedy.