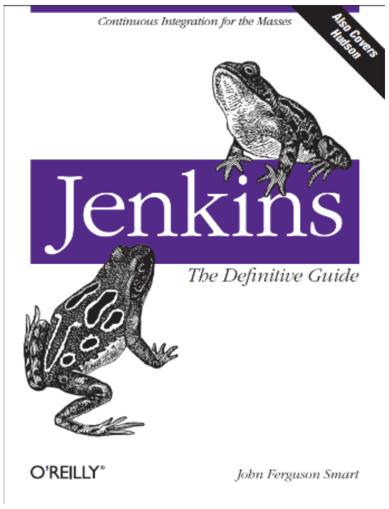
The Seven Phases of Introducing Continuous Integration into Your Organization

by John Ferguson Smart MVB · Aug. 10, 11 · Agile Zone

Reduce testing time & get feedback faster through automation. Read the Benefits of Parallel Testing, brought to you in partnership with Sauce Labs.

Continuous Integration is not an all-or-nothing affair. In fact, introducing CI into an organization takes you on a path that progresses through several distinct phases. Each of these phases involves incremental improvements to the technical infrastructure as well as, perhaps more importantly, improvements in the practices and culture of the development team itself. In the following paragraphs, I have tried to paint an approximate picture of each phase.

(From 'Jenkins: The Definitive Guide')



Phase 1—No Build Server

Initially, the team has no central build server of any kind. Software is built manually on a developer's machine, though it may use an Ant script or similar to do so. Source code may be stored in a central source code repository, but developers do not necessarily commit their changes on a regular basis. Some time before a release is scheduled, a developer manually integrates the changes, a process which is generally associated with pain and suffering.

Phase 2—Nightly Builds

In this phase, the team has a build server, and automated builds are scheduled on a regular (typically nightly) basis. This build simply compiles the code, as there are no reliable or repeatable unit tests. Indeed, automated tests, if they are written, are not a mandatory part of the build process, and may well not run correctly at all. However developers now commit their changes regularly, at least at the end of every day. If a developer commits code changes that conflict with another developer's work, the build server alerts the team via email the following morning. Nevertheless, the team still tends to use the build server for information purposes only—they feel little obligation to fix a broken build immediately, and builds may stay broken on the build server for some time.

Subscribe

Phase 3—Nightly Builds and Basic Automated Tests

The team is now starting to take Continuous Integration and automated testing more seriously. The build server is configured to kick off a build whenever new code is committed to the version control system, and team members are able to easily see what changes in the source code triggered a particular build, and what issues these changes address. In addition, the build script compiles the application and runs a set of automated unit and/or integration tests. In addition to email, the build server also alerts team members of integration issues using more proactive channels such as Instant Messaging. Broken builds are now generally fixed quickly.

Phase 4—Enter the Metrics

Automated code quality and code coverage metrics are now run to help evaluate the quality of the code base and (to some extent, at least) the relevance and effectiveness of the tests. The code quality build also automatically generates API documentation for the application. All this helps teams keep the quality of the code base high, alerting team members if good testing practices are slipping. The team has also set up a "build radiator," a dashboard view of the project status that is displayed on a prominent screen visible to all team members.

Phase 5—Getting More Serious About Testing

The benefits of Continuous Integration are closely related to solid testing practices. Now, practices like Test-Driven Development are more widely practiced, resulting in a growing confidence in the results of the automated builds. The application is no longer simply compiled and tested, but if the tests pass, it is automatically deployed to an application server for more comprehensive end-to-end tests and performance tests.

Phase 6—Automated Acceptance Tests and More Automated Deployment

Acceptance-Test Driven Development is practiced, guiding development efforts and providing high-level reporting on the state of the project. These automated tests use Behavior-Driven Development and Acceptance-Test Driven Development tools to act as communication and documentation tools and documentation as much as testing tools, publishing reports on test results in business terms that non-developers can understand. Since these high-level tests are automated at an early stage in the development process, they also provide a clear idea of what features have been implemented, and which remain to be done. The application is automatically deployed into test environments for testing by the QA team either as changes are committed, or on a nightly basis; a version can be deployed (or "promoted") to UAT and possibly production environments using a manually-triggered build when testers consider it ready. The team is also capable of using the build server to back out a release, rolling back to a previous release, if something goes horribly wrong.

Phase 7—Continuous Deployment

Subscribe

Confidence in the automated unit, integration and acceptance tests is now such that teams can apply the automated deployment techniques developed in the previous phase to push out new changes directly into production.

The progression between levels here is of course somewhat approximate, and may not always match real-world situations. For example, you may well introduce automated web tests before integrating code quality and code coverage reporting. However, it should give a general idea of how implementing a Continuous Integration strategy in a real world organization generally works.

From http://weblogs.java.net/blog/johnsmart/archive/2011/08/08/seven-phases-introducing-continuous-integration-your-organization

The Agile Zone is brought to you in partnership with Sauce Labs. Discover how to optimize your DevOps workflows with our cloud-based automated testing infrastructure.

Topics:

It's Time for Forums to Die

by Matthew Casperson MVB · Oct 15, 16 · Agile Zone

Learn more about how DevOps teams must adopt a more agile development process, working in parallel instead of waiting on other teams to finish their components or for resources to become available, brought to you in partnership with CA Technologies.

When I first had a post flagged on StackOverflow as being “closed as off topic,” I was kind of annoyed. I felt that my question was genuine, and ironically, at over one hundred thousand views, it was also the most popular question I have asked on StackOverflow. But those are the rules, like it or not.

Fast forward a few years, and I found myself trying to articulate some of the advantages Spring had over using the JBoss implementation of Java EE.

It seemed that any time I had a problem with Spring, a simple Google search revealed a StackOverflow post that laid out the same problem and the solution.

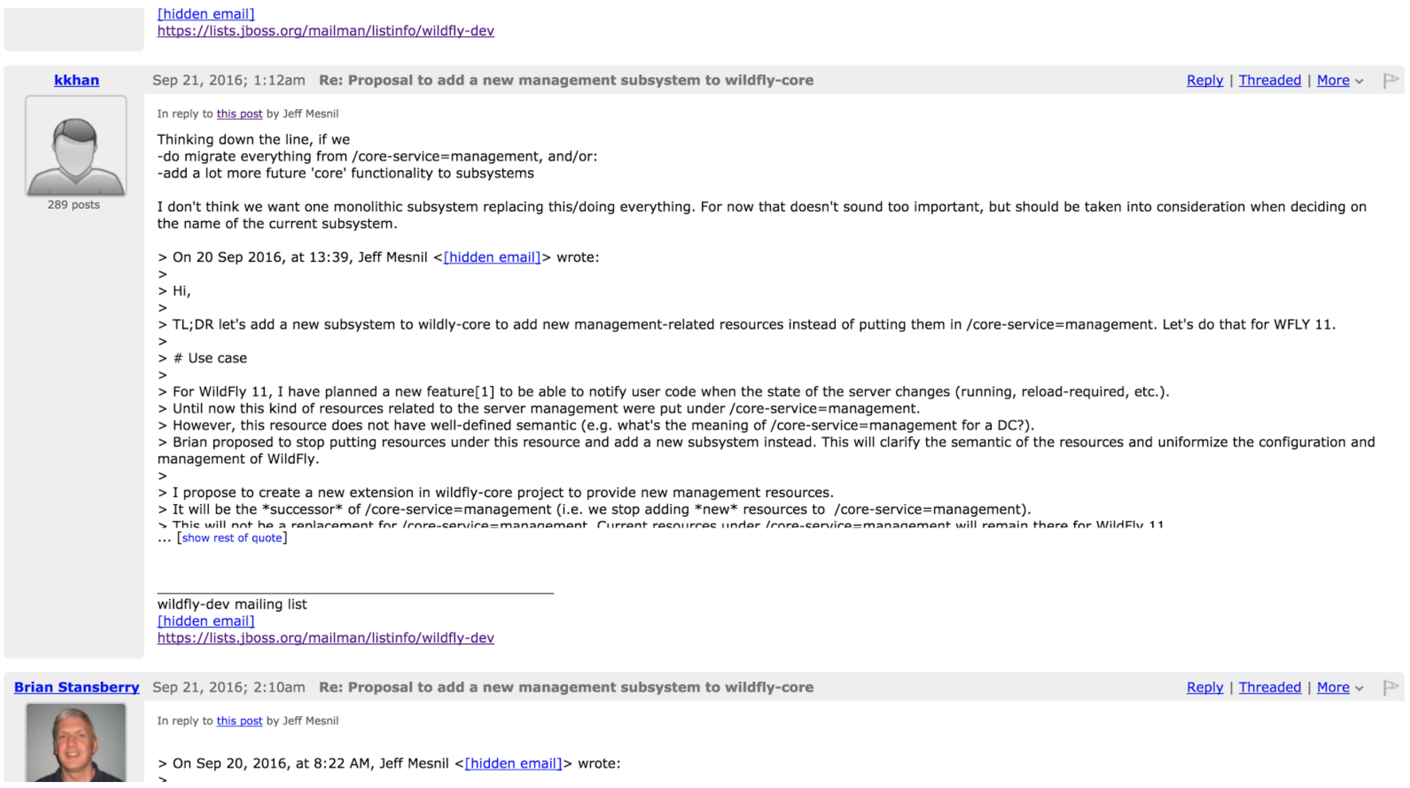
I couldn’t really say the same about the JBoss libraries. More often than not, I would be taken to a forum post that I would have to read top to bottom in order to work out if the issue being discussed was related. Every time I saw a link to the JBoss forums, I sighed, knowing just how much work I was going to have to do to get anything useful out of the content.

And don’t get me started on mailing lists. I have yet to see a pleasant way to browse mailing lists, what with all their inline quotes, signatures, and awkward threading displays.

Subscribe

Hello, the 90's are calling and they want their Frontpage web design back.

wildfly-dev mailing list



[hidden email]
https://lists.jboss.org/mailman/listinfo/wildfly-dev

kkhan Sep 21, 2016; 1:12am **Re: Proposal to add a new management subsystem to wildfly-core** Reply | Threaded | More

289 posts

In reply to this post by Jeff Mesnil

```
Thinking down the line, if we
-do migrate everything from /core-service=management, and/or:
-add a lot more future 'core' functionality to subsystems

I don't think we want one monolithic subsystem replacing this/doing everything. For now that doesn't sound too important, but should be taken into consideration when deciding on
the name of the current subsystem.

> On 20 Sep 2016, at 13:39, Jeff Mesnil <[hidden email]> wrote:
>
> Hi,
>
> TL;DR let's add a new subsystem to wildly-core to add new management-related resources instead of putting them in /core-service=management. Let's do that for WFLY 11.
>
> # Use case
>
> For WildFly 11, I have planned a new feature[1] to be able to notify user code when the state of the server changes (running, reload-required, etc.).
> Until now this kind of resources related to the server management were put under /core-service=management.
> However, this resource does not have well-defined semantic (e.g. what's the meaning of /core-service=management for a DC?).
> Brian proposed to stop putting resources under this resource and add a new subsystem instead. This will clarify the semantic of the resources and uniformize the configuration and
management of WildFly.
>
> I propose to create a new extension in wildfly-core project to provide new management resources.
> It will be the *successor* of /core-service=management (i.e. we stop adding *new* resources to  /core-service=management).
> This will not be a replacement for /core-service=management. Current resources under /core-service=management will remain there for WildFly 11
... [show rest of quote]

_______________________________________________
wildfly-dev mailing list
[hidden email]
https://lists.jboss.org/mailman/listinfo/wildfly-dev
```

Brian Stansberry Sep 21, 2016; 2:10am **Re: Proposal to add a new management subsystem to wildfly-core** Reply | Threaded | More

In reply to this post by Jeff Mesnil

```
> On Sep 20, 2016, at 8:22 AM, Jeff Mesnil <[hidden email]> wrote:
>
```

Despite my initial displeasure at having StackOverflow posts flagged as off-topic or conversational, I now appreciate the genius behind the decision. While conversational posts are a great way to support the first person to ask a question, they are a very poor way to support the next thousand that Google the same problem. Not even the ability to move a helpful answer to the top of the post seems to make forums as instantly useful as a StackOverflow post.

I think it is time to accept that, for developer related issues, the QA format so zealously promoted by StackOverflow has won, providing much better support for modern software development:

"Traditional software dev was like farming. You bought your tool stack and got busy. Now we're more like foragers."

For software development support, it is time to put traditional forums, Wikis, and mailing lists to pasture, because the needs of the many people Googling a problem outweigh the needs of the few who fill entire pages with streams of consciousness as they go from problem to solution. In the end, your customers will thank you.

Discover the warning signs of DevOps Dysfunction and learn how to get back on the right track, brought to you in partnership with CA Technologies.

Topics: SOFTWARE DEVELOPMENT, STACKOVERFLOW, FORUMS, AGILE

Subscribe

Agile Transformation in Practice: Part V

by Ian Mitchell MVB · Oct 14, 16 · Agile Zone

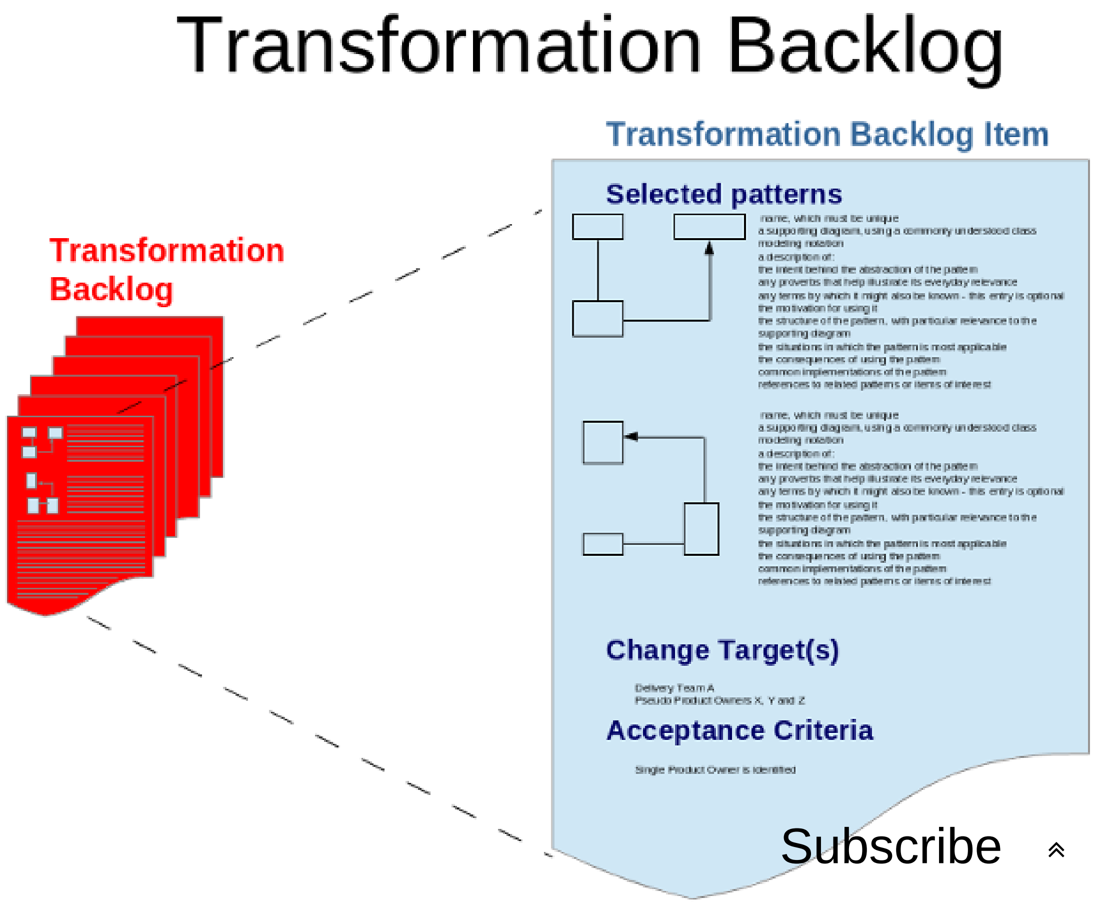
Reduce testing time & get feedback faster through automation. Read the Benefits of Parallel Testing, brought to you in partnership with Sauce Labs.

Transformation Backlog

Executive sponsorship is an essential pre-requisite for Agile adoption at enterprise scale. Only narrow tactical improvements – those that 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 (similar terms and constructs include Change Backlog, Pattern Backlog, and Practice 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.


Transformation Backlog
Transformation Backlog
Transformation Backlog Item
Selected patterns
name, which must be unique
a supporting diagram, using a commonly understood class modeling notation
a description of:
the intent behind the abstraction of the pattern
any proverbs that help illustrate its everyday relevance
any terms by which it might also be known - this entry is optional
the motivation for using it
the structure of the pattern, with particular relevance to the supporting diagram
the situations in which the pattern is most applicable
the consequences of using the pattern
common implementations of the pattern
references to related patterns or items of interest
name, which must be unique
a supporting diagram, using a commonly understood class modeling notation
a description of:
the intent behind the abstraction of the pattern
any proverbs that help illustrate its everyday relevance
any terms by which it might also be known - this entry is optional
the motivation for using it
the structure of the pattern, with particular relevance to the supporting diagram
the situations in which the pattern is most applicable
the consequences of using the pattern
common implementations of the pattern
references to related patterns or items of interest
Change Target(s)
Delivery Team A
Pseudo Product Owners X, Y and Z
Acceptance Criteria
Single Product Owner is identified

Subscribe

1. 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.

2. 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. As they mature, less assistance will be needed and the Transformation Team can prioritize change in 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 that 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.

3. Acceptance Criteria

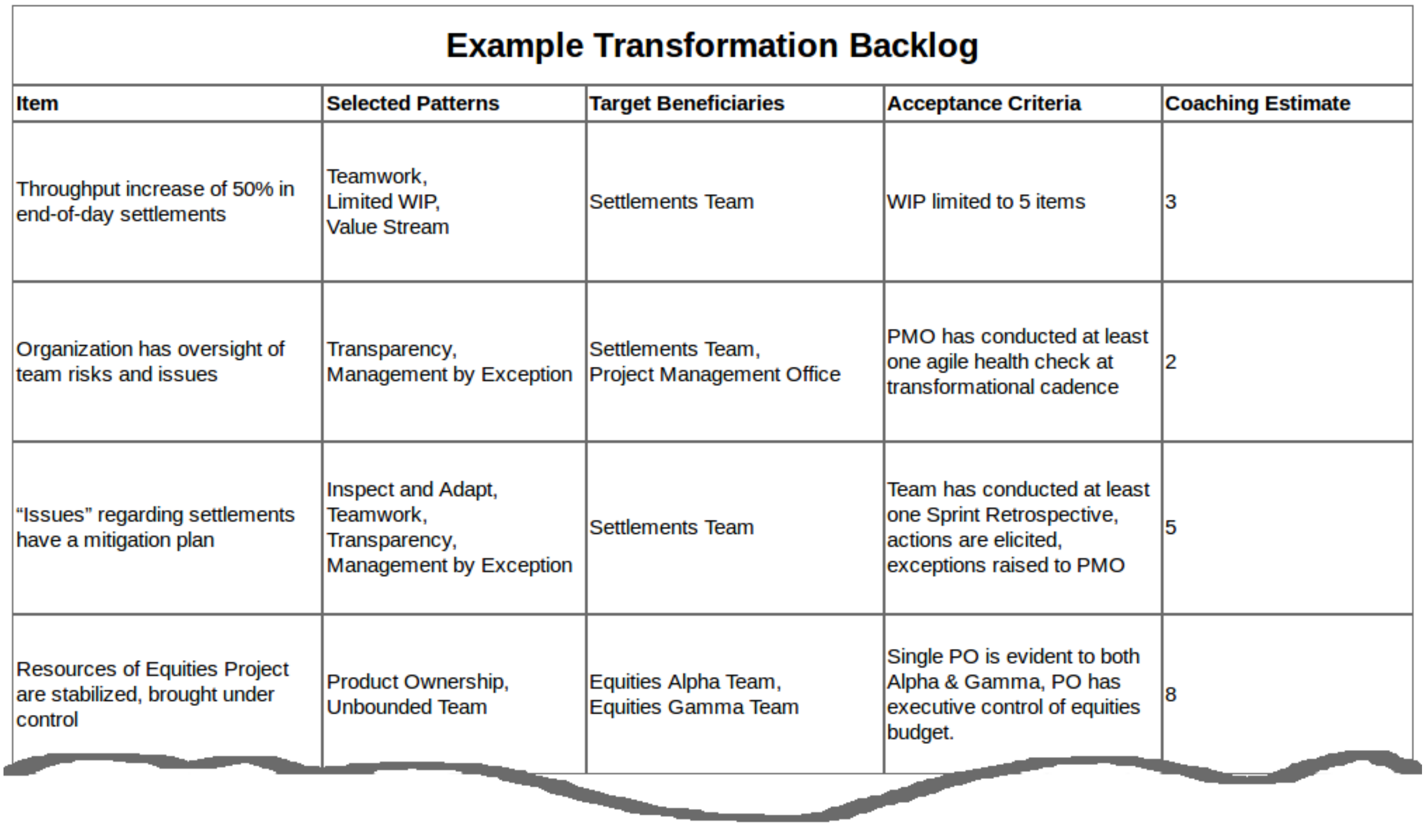
Each Transformation Backlog Item must have clear Acceptance Criteria. These are the evidences that prove that the change has been applied successfully. They are often expressed in terms of roles, demonstrable behaviors, events and their outputs, qualities of specific artifacts, and (ideally non-lagging) 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

Subscribe

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 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 a 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.



Example Transformation Backlog

Item	Selected Patterns	Target Beneficiaries	Acceptance Criteria	Coaching Estimate
Throughput increase of 50% in end-of-day settlements	Teamwork, Limited WIP, Value Stream	Settlements Team	WIP limited to 5 items	3
Organization has oversight of team risks and issues	Transparency, Management by Exception	Settlements Team, Project Management Office	PMO has conducted at least one agile health check at transformational cadence	2
"Issues" regarding settlements have a mitigation plan	Inspect and Adapt, Teamwork, Transparency, Management by Exception	Settlements Team	Team has conducted at least one Sprint Retrospective, actions are elicited, exceptions raised to PMO	5
Resources of Equities Project are stabilized, brought under control	Product Ownership, Unbounded Team	Equities Alpha Team, Equities Gamma Team	Single PO is evident to both Alpha & Gamma, PO has executive control of equities budget.	8

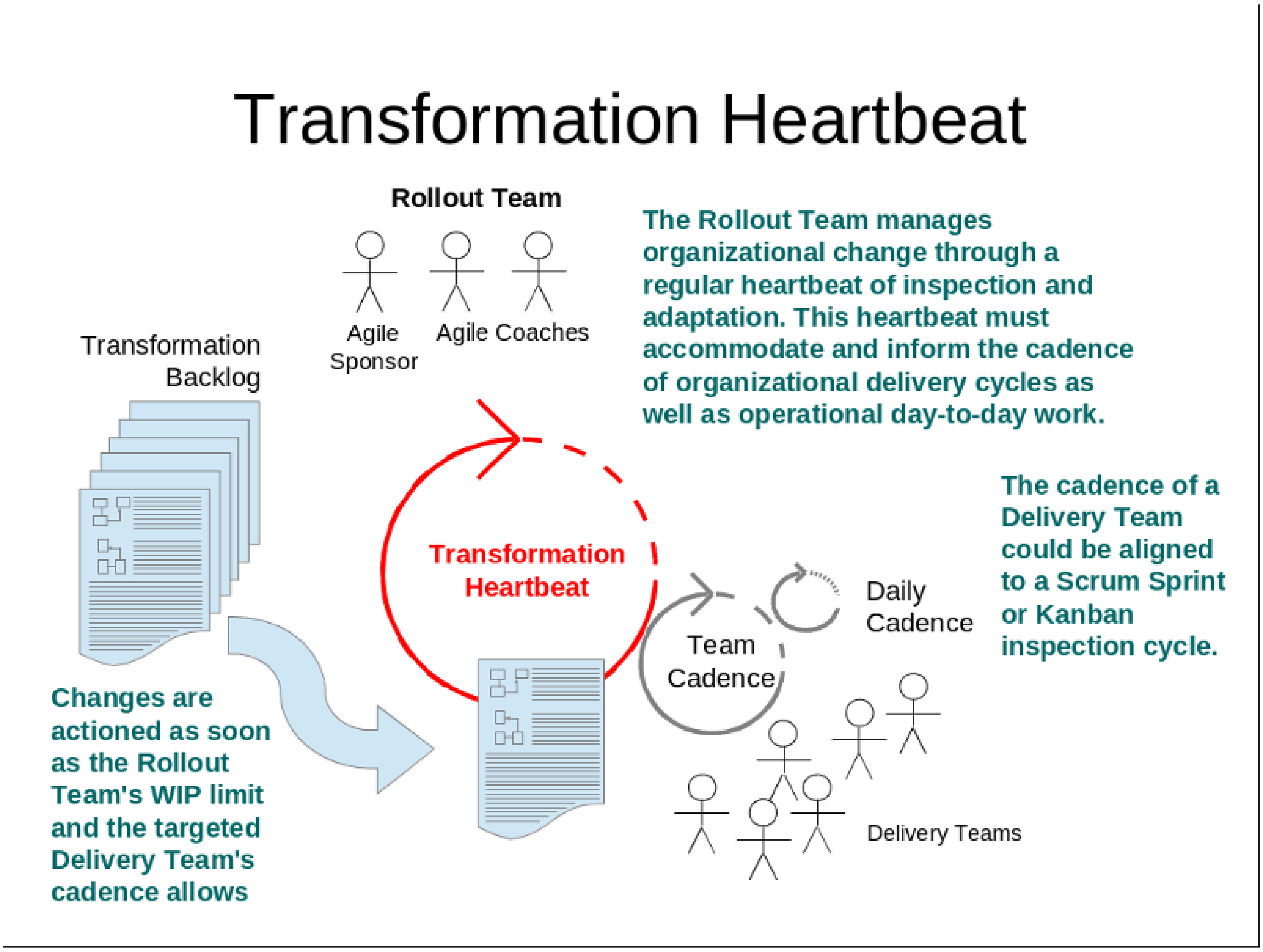
Transformation Heartbeat

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. The 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 the project, program, and portfolio level. They may include teams with strategic management responsibilities as well as those that are more technically or operationally focused.

Subscribe

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.


Transformation Heartbeat
Rollout Team
Agile Sponsor
Agile Coaches
The Rollout Team manages organizational change through a regular heartbeat of inspection and adaptation. This heartbeat must accommodate and inform the cadence of organizational delivery cycles as well as operational day-to-day work.
Transformation Backlog
Transformation Heartbeat
The cadence of a Delivery Team could be aligned to a Scrum Sprint or Kanban inspection cycle.
Daily Cadence
Team Cadence
Changes are actioned as soon as the Rollout Team's WIP limit and the targeted Delivery Team's cadence allows
Delivery Teams

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 do 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.

Topics: AGILE TRANSFORMATION, AGILE, TRANSFORMATION BACKLOG, ENTERPRISE

Subscribe