Aligning HP® ALM to Scrum & SAFe

Elite Partner

Software

Northway Solutions Group
9005 Overlook Blvd • Brentwood, TN. 37027 • Ph: 866.611.8762 • Fax: 866.611.8762
www.northwaysolutions.com

Table of Contents

Overview – Challenges Aligning HP® ALM to Agile

The Scaled Agile Framework (SAFe) is a fully scalable set of agile best practices used at an enterprise scale. It is often selected when it is clear that Scrum, while effective for a small organization with fairly independent teams, is not scalable for a large enterprise with multiple delivery teams. The framework is public domain, and contains processes and tools that help organizations define effective agile at scale.

At the heart of SAFe is the work delivered at the team level. At this level, SAFe looks very similar to the traditional agile Scrum processes. There is a team backlog, and sprints (or iterations) are planned in the traditional fashion. Daily standup meetings ensure that progress is being made during the sprint. There are also the key roles of agile, Product Owner, Scrum Master, and team. User stories are created and managed in the team and sprint backlogs. Sprints can be managed in a very manual fashion using sticky notes on a Scrum Board. Some teams may opt to leverage an agile management tool, such as HP® Agile Manager. It is important that the stories are available to both the development and the testing team members during the sprint.

This is where HP® Application Lifecycle Management (ALM) can provide tremendous value to the Quality Assurance team. Testing teams are typically very well versed in using ALM to do traditional waterfall centric testing, but they struggle to figure out how to leverage ALM in an agile environment. We often find that there are several contributing factors to this challenge.

1) A lack of understanding of basic agile principles
2) A lack of understanding about the role of QA in agile projects.
3) A lack of understanding of agile testing techniques.
4) An attempt to use a waterfall ALM configuration to deliver agile work.
5) A lack of ALM configuration and administration expertise.

This paper will briefly address these challenges and provide some recommendations that can help the testing team improve their value to the agile delivery. The recommendations simply include observations and practices that we have found effective for many of the engagements delivered for our clients. We hope they will be of value to you.

A lack of understanding of basic agile principles

Agile is nothing like traditional waterfall methodologies. Traditional projects last anywhere from three (3) months to two (2) years and have extremely long phases. The success of each phase depends on the correct completion of the preceding phase. Waterfall projects are typically undertaken by very large teams with a large focus on defining the solution completely up front in a set of business requirements. Testing in this environment happens as part of a phase of the waterfall lifecycle. Test cases prepared earlier to verify a specific requirement or set of requirements are executed, and defects are provided back to development. HP® ALM lends itself very well to the waterfall processes. Releases and rounds of testing can be defined in the tool. By default ALM has basic the requirement types most commonly associated with waterfall projects, and default reporting is very waterfall centric.

What happens is that these waterfall concepts don't translate easily to agile, and in particular, Scrum. Testing is no longer a phase completed after "code freeze", but is something that happens at an individual story level when the story is completed by the developer. Testing may not even be accomplished by an "independent" testing function, but by a developer other than the one who developed the story. Where teams get into trouble with agile using ALM is trying to force fit the concepts of waterfall to their agile projects. They attempt to treat testing as a phase within a sprint rather than a task on a story. They wait until all of the stories for a sprint have been completed and then begin their round of testing. In other words, they turn agile into mini waterfalls.

A lack of understanding about the role of QA in agile projects

The traditional role of QA may have to change in an agile environment. While there may be some regulatory reasons that an organization may have to retain an "independent" testing team, for most organizations this may not be the case. Does this mean there is no place for the QA team? Absolutely not! What it means is that organizations need to reassess how and when they are using "independent" testers. We have one client who still imbeds testers onto the scrum team. They participate in the sprint as just another role on the team; much like a BA or a DBA would be specialized skills on the scrum team. This is particularly useful when an agile team is new and transitioning from waterfall to agile. It retains some of the discipline of a waterfall approach, but moves into agile behaviors. However, the tester must become an agile tester rather than a traditional waterfall tester. This means that they now focus on testing at the story level rather than at the completed feature level. This will typically require a more technical tester as they are testing closer to the production of the code, rather than just at the completed feature level.

Another option for leveraging QA in the agile paradigm is to abstract the "independent" team out of the scrum team entirely. What QA becomes is an assurance function that has the responsibility to review the results of testing to ensure that the product was adequately tested. They may provide an Independent Integration Verification & Validation (IIVV) function, which conducts an independent round of testing for completed sprints. This IIVV would focus on verifying that the definition of done for the Potentially Shippable Increment (PSI) was achieved. This is accomplished often in parallel with the sprints, but one sprint removed, or is accomplished in a hardening sprint close to the end of the release. In any event the core testing responsibility for the project is accomplished within the scrum team. The QA team is the "independent" check and balance. This is a strategy that has proven very effective for decades in Aerospace. Engineering groups design and build the flight software, while a separate team of engineers in the same group test the software under the watchful eye of the Quality Assurance team. Finally, an "independent" IIVV function performs final testing to validate the solution meets the needs of the launch.

A lack of understanding of agile testing techniques

In traditional waterfall testing we are writing a test case to verify the correct implementation of a very specific functional requirement. The test is very specific and is aimed at verifying a very narrowly focused requirement. In agile, there are very few narrowly focused requirements. You have a user story in the form of:

As a … <*role*>

I need to … <*do something*>

So that … <*some result*>

User stories by the very nature are very high-level. They lack the very specific detail you would have in a set of functional specifications. Yes, you can write very low level stories that are intended to define some feature, but typically they are still not to the same level of granularity as detailed functional specifications. This lack of specificity means that the test cases you write will very likely be much more generic for agile testing. While you may have discussions on the scrum team about the more specific requirements (such as, wire frames), often you will need to define a more simplified test scenario. Some may be worried about this, but remember the entire concept of agile is that you deliver smaller pieces faster, and you adjust to change quickly. If we are getting good stories, then we should still be able to get good test cases. Agile assumes a lower level of test case detail and a higher level of exploration to verify the story has been met. We will talk about how ALM can be used to better manage that exploratory testing later in the paper.

Your goal coming out of every sprint should be to have a set of more detailed test cases than when you entered the sprint. I believe a more important goal should be to have a set of test cases that can be readily automated. This means that sufficient detail has to exist for an automation resource to convert that manual test case into an automated script.

An attempt to use a waterfall ALM configuration to deliver agile work

It should be clear by now that waterfall and agile are nothing alike. Just like you should not use a hammer to drive a screw, you shouldn't attempt to use your waterfall configured ALM project to do agile projects. The agile entities are completely different between the two. While entities vary, the table below is a good representation of some of these differences.

WATERFALL ENTITIES	AGILE ENTITIES
Project	Release
Phases	Sprints
Business, Functional, Technical Requirements	Epics, Features, User Stories
Unit, System, Integration, UAT Test Cases	Story, and UAT Test Cases
Defects	Defect Task/Defect Story

These are just a few of the differences between the entities in scrum versus waterfall. As you can see, treating agile artifacts and entities as waterfall is not a viable solution for managing an agile project. As a clear example of the differences we can discuss how defects are handled. In traditional waterfall, defects are captured and then assigned out to development for repair. Once repaired, a build is prepared, and then the defect is retested and closed if resolved. In agile, a defect is associated to a specific story and is simply a task for the story. It doesn't count as a defect if it is resolved within the sprint as it is just a task. If the defect isn't resolved during the sprint, it gets placed into the team backlog as a defect story. It is no longer just a task on a story, but a story in and of itself. It gets prioritized in the backlog and for a sprint. When it gets slated into a sprint it gets tasked out just like any story. Developers create code to repair the defect and the tester either creates or executes an existing test to complete the story. The defect story is then accepted by the Product Owner. This is not traditional testing like waterfall. It takes different behaviors in the tool. The tool should be configured to fit agile practices so that the tool doesn't become a barrier to effective testing in agile.

A lack of ALM configuration and administration expertise

One of the biggest challenges I see with teams trying to leverage ALM for agile, is the lack of an internal resource that has the knowledge and skills in ALM to properly configure the tool to meet the needs of the organization. Companies purchase HP® ALM and they may have received some basic training in the tool, but they don't avail themselves of advance administrative training. They find one of their more technical testing resources and assign them as the ALM Site Administrator. The resource is then left with no training, and only has the Administrator Guide to figure out how to get the most out of the solution. The company then wonders why the tool doesn't live up to the expectations of leadership.

HP® ALM is a very powerful and feature rich toolset designed to manage virtually all aspects of the test management lifecycle, and as such it requires a very high level of expertise to manage. When companies purchase the tool they often desire to "go it alone". They want to ensure that they have ownership of the tool, so they attempt to configure it on their own. While I applaud the desire to take ownership of their purchase, they are not being responsible with the large investment that they have made in the solution. They are actually limiting the value they are getting out of their investment. A leader once told me “*a good idea, poorly delivered, becomes a bad idea*”. What he was telling me is that no matter how good your intentions, if you don’t follow them up with the right action, you will get squat! Too many companies spend large sums on ALM only to try to figure it out themselves. What they end up with is a mess, and a loss of confidence in the tool.

If you invest your capital in a “world class” tool, be prepared to invest in the right resources to ensure that your solution is effective and adds value to your organization.

Aligning HP® ALM to agile processes

The primary focus of this paper is to help teams identify how to effectively align HP® ALM to their agile processes. When customers purchase a tool, unintentionally they are intellectually purchasing the demo. They sat through countless presentations from the vendor where a highly polished version of the tool was demonstrated. There was realistic data and structure to the demo so that the client could see what was possible with the tool. Soon after the purchase the client realizes that they have purchased an empty shell. When they engage a consultant to come in an implement the tool you almost always hear the client say those four little words that every consultant loves to hear: “*out of the box*”. The client indicates that they don’t want any customization and they want to use the product just as it comes from the vendor. In other words, they want it “*out of the box*”, but just like the demo they saw! Let me help you understand what “*out of the box*” means. In ALM terms this means installing the tool with no configured fields, no lists, no workflow, no favorite views, no folders, no structure, no anything.

The best analogy I can provide is this. Using ALM “*out of the box*” is like opening a brand new box of LEGO® brand building blocks. There is a great picture on the outside of the box of a completed project, but when you open the box all you find are the basic components to build the solution. You can put those building blocks together in countless configurations, each one producing a different result. None of which are wrong. The only thing that is absolute is that just taking the LEGO® brand building blocks out of the box and doing nothing with them is absolutely useless! You have to build some configuration of those blocks to get to any meaningful result. The same holds true for HP® ALM.

You have to define a process, including fields, lists, and behaviors in order for the tool to provide meaning. This holds true for each module of ALM, not just the defect module. It doesn't make sense to try and leverage the "*out of the box*" business requirement type to manage user stories. (1) They don't have the same metadata fields at all. User stories have fields like Acceptance Criteria, and Story Points just to name a couple. (2) Another fundamental difference is the way that stories are completed. Stories are aligned to sprints over the duration of a project rather than being delivered up front as part of a business requirements artifact. Furthermore, (3) the states of a story are aligned with the delivery of the story, such as: Defined, In Progress, Completed, and then Accepted; the statuses of a business requirement are often aligned to the status of the requirement in the business requirement document, such as: Proposed, Reviewed, Approved, Retired. In order to align ALM to agile, you have to make some configuration updates, and put the building blocks together in a slightly different way.

The HP® ALM solution doesn't come "*out of the box*" with a default set of best practice entities and processes. It is up to the client to define them for themselves and then to configure ALM to support those processes. Make sure that you plan to engage an HP® ALM solution consultant to help you implement your methodology in the tool. If you don't have a methodology, then you may not gain the value from the tool you projected. It will be important to first define your methodology. We always recommend that you do this before you purchase a tooling solution to ensure that the solution meets with the needs of your process. It also gives you the ability to refine and improve your processes before you commit them to a digital solution. If you purchased to tool before you defined your process, and then make sure that you engage a very experienced process consultant who understands not only testing, but also understands the ALM tool and how to configure it.

Agile Organization and Structure

There are two major components to utilizing ALM, (1) the configuration of the platform & project, and (2) the setup of the working structure within an ALM project. I will soon be sharing how to configure and set up an agile project template, but first wanted to focus on the operational use of an agile project template. When ALM is configured to have user stories, and features, and other agile structures such as reporting, it is still not totally ready for operational use. You have to understand how to leverage the agile configuration for the successful execution of a project.

If we start with the Management module of ALM, we need to first define how releases and cycles will be leveraged under Releases. In traditional testing, we would typically define the release as the beginning and end of the project, and then define each round of testing as one or more cycles. In SAFe and Scrum it is not exactly the same. We have to make the choice about what the release defines. The release could define some Potentially Shippable Increment (PSI) or it could define a quarter in the case of SAFe. Fundamentally, it is still a PSI as it is a quarterly PSI, however, there could have been multiple value drops during that quarter. In either event, we would define the release and then leverage cycles to define our sprints/iterations. We can then assign individual stories to the appropriate sprint.

In addition to adding stories to the sprint, we would define scope for the release. In traditional waterfall projects, this is accomplished by defining some feature or function in the release, and then defining the business, functional and technical requirements that are a part of that feature. You would then define the scope items for each round of testing. In agile, you could still define the features as scope items, and then associate the stories that satisfy that scope. Another option is to create each sprint as a scope item, and then associated the stories that are committed in that sprint. The advantage is that now management reporting within Project Planning & Tracking is aligned to the execution of sprints, so you can track sprint progress toward the overall release. Either way is acceptable depending on your focus.

Don’t forget to also define how you plan to leverage Libraries to manage your stories and testing assets across projects. In SAFe, work is sent to a team on a train. While the entire train may be executing their work in a single ALM project space, the assets they are producing may be for applications that impact multiple trains. You will have to devise a mechanism for setting baselines and then sharing those completed assets. Having ALM application project spaces that are simply asset repositories is an effective solution. A SAFe train would produce testing assets during the execution of a release. Upon completion of the release they would baseline the test assets and then share them with the appropriate application space. Each train could then go to the shared asset repository to get the latest version of tests they may need. While this sounds simple, it is a very complex process, and will take a coordinated effort and some key data fields to make effective.

Agile Project Planning & Tracking

I wanted to spend just a moment covering some agile considerations for Project Planning & Tracking in the Management module. In SAFe and Scrum it is all about the stories. The default Key Performance Indicators (KPIs) in ALM are not aligned to agile at all. They are focused on a test centric approach rather than a story centric approach. In waterfall it is all about the testing, as demonstrated by a phase dedicated to testing the completed product. In agile it is all about the story, and testing is simply a task that needs to be accomplished to say the story is complete and ready for acceptance by the Product Owner. In your agile template you will either create new KPI measures or you will have to modify the existing measures to be story focused. While some of the traditional testing and defect related KPI measures can be used (e.g. Test Cases Created, Test Cases Executed, Defect Severity, etc.), you will need to create some story and feature KPI metrics to meet your needs. Some examples are provided here.

- Stories Ready – the percentage of stories in the release backlog that are Defined
- Stories Defined – the percentage of stories assigned to the sprint that are Defined or greater.
- Stories Covered – the count of stories that have test cases that cover them
- Stories Complete – the percentage of stories that are in a Direct Cover Status of Passed

While this isn’t an exhaustive list of KPI metrics, you can see they are focused on the story rather than the test. As will all KPI metrics, you will have to determine the KPI criteria for each of these measures. What Project Planning & Tracking allows is the management of the release delivery from a quality viewpoint. Stories don’t become complete until they have successfully passed all of the associated test cases. It allows the team to manage based on KPI’s rather than on “gut” feel, and is one of the most compelling, but unfortunately least used capabilities of HP® ALM. I believe that the reason it isn’t leveraged is that teams are so busy just trying to get stuff done that they don’t feel they have the time to manage by objective. It does take significant effort up front to define the KPI metrics, but once that investment is made it is easy to leverage them going forward. If you have created a default release structure in your template then each quarterly release can be built from that master release structure. This reduces the amount of work to kick off the next quarterly release.

Agile Features & Stories

User stories are not exclusively an agile concept; however, they are almost always associated with Scrum. The requirements hierarchy for SAFe is fairly straightforward:

- ***Epic*** – An Epic usually defines some major delivery of value that will take multiple quarters to deliver. In a traditional sense, this is equivalent to a project. In ALM this may be tracked as field (e.g. Project). In the Release module, a folder with each quarterly release would represent the Epic.
- ***Feature*** – Epics are made up of one or more features that can be delivered within a single quarter. Features show up as scope within Project Planning & Tracking, and as a requirement type.
- ***User Story*** – Features are made up of user stories. Stories are defined to be small enough that they can be fully completed (define - build - test) in a single sprint. Stories show up as scope within Project Planning & Tracking; are associated to a specific sprint (cycle) under a quarterly release; and, are captured as story requirements. User stories are traced to the feature they define.
- ***Task*** – User Stories have one or more tasks that define the activities that must happen in order to complete the story. Tasks should be specific and small enough that progress can be readily reported at each scrum meeting. Tasks may or may not be captured in ALM as a unique requirement type. If ALM is used to capture them, they would not have coverage turned on, but would be traceable to the story.

Each unique requirement type in ALM can have a different set of fields, lists, and behaviors. Since Features and Stories have some fields in common, but not all, a unique type should be established for each. Let’s look at one example, user story. The key fields for a story are as follows:

FIELD/LIST	DESCRIPTION
Summary	***Default Field*** – The Summary field is common for all types and is the Name field renamed.
Description	***Default Field*** – The Description field is common for all types and is leveraged as is.
Status	***Default Field*** – The Status field is common for all types and is the Reviewed field renamed. The list associated with status changes based on the requirement type.
Author	***Default Field*** – The Author field is common for all types and is the Assigned To field renamed.
Priority	***Default Field*** – The Priority field is common for all types and is leveraged as is.
Target Release	***Default Field*** – The Target Release field is common for all types. The field is set to read only as the selection of a Sprint automatically sets the value.
Target Sprint	***Default Field*** – The Target Sprint field is common for all types and is the Target Cycle field renamed.
Story Points	***Custom Field*** – The Story Points field is a number field, which becomes required at the point the Story reaches the status of Defined.
Acceptance Criteria	***Custom Field*** – The Acceptance Criteria field is a memo field, which becomes required at the point the Story reaches the status of Defined.

Some optional fields you might want to consider are:

FIELD/LIST	DESCRIPTION
Story Kanban	***Custom Field*** – The Story Kanban field is a list field that enables the define-build-test team to move the story through a virtual Kanban board. Phases might include: Define, Develop, Test, and Done
Ready	***Custom Field*** – The Ready field is a Yes/No list used to indicate that the story Kanban state is complete and ready for the next team member to "pull" that story into their work queue.

In ALM you have the ability to define not only the layout of the fields and which fields are required, but you can also define when the fields appear. It doesn't make sense to display Date Created, and the Direct Cover Status on a new requirement entry form. You know who you are! Real estate is precious on these forms, so only display what is essential for the logged in user at that time. Also, make good use of pages on forms. Having a second tracking tab is a great way to track information that is needed, but not essential to the work. Making forms easy to use by an end user is anything but easy for the designer. You have to have some basic foundational knowledge of how to develop custom workflow in ALM. The recommendation I give to all new ALM administrators I train is to always start with a blank canvas. Every time you open or move to an entity, hide all fields; make them not read only; and make them not required. This ensures that you always know where you are starting. You can then set the form layout based on the entity type and the logged in user. By doing so, you are making the user experience truly custom for their role.

If you are using integration with other agile tools, make sure that you know which tool is the system of record, and whether or not you want one way synchronization or bi-directional synchronization. You may also want the entity to only be authored in one solution. If so, then the record would be read only in ALM and only some fields could be updated (e.g. Status, Sprint, and Direct Cover Status), but more on this later in the paper.

Agile Test Design & Execution

As I look at waterfall versus agile test case development, there aren't fundamentally a lot of differences in the basics of test cases. You will have very specific test cases aimed at verifying a single story, and you will have end-to-end test cases that verify some scenario through a system. What fundamentally changes is the level of detail. As I stated earlier, the level of detail in a user story is more organic and less defined than the traditional waterfall requirements. What it drives is very close communication and collaboration within the define-build-test team. The tester needs to understand in a much more intimate way the direction that the solution is taking. They will need to elaborate test cases, just as the team is elaborating the solution from the story. Test cases will tend to be less specific in their writing and seem almost like the test steps are verification objectives. For example, the following feature may be developed for your sprint.

As a Client Service Representative

I need to be able to open a new banking account

So that customers can start banking with Biz Bank

As you see this feature is fairly high level. It would likely need to be broken down into multiple stories. One of which might be as follows:

As a Client Service Representative
I need to be able to access a new account screen
So that I can set up new customers

This story is only one of many, and the scope is limited to only being able to access a new account screen. It doesn't cover the creation of that account from that screen. That would be a separate story. The developer will define some tasks to create the new screen including how to access the screen, and the fields and widgets that are on the screen. As a tester, you would need to create a test case, *Access Account Screen*, to verify access to the new account screen. It would involve navigation, and screen layout. The specific layout may come in the form of a user experience design or wireframes. Your test step may be simply to verify the navigation path, and a second step to verify the layout of the screen to the wireframes.

What you end up with is a very high level set of test cases that are highly dependent on the knowledge of the tester. It assumes that the tester has been an intimate part of the team (co-located). It also assumes that the knowledge is maintained over time. Where you find weakness is in team turnover. As team members move on the tests begin to make less sense to the remaining team. This is especially true when you factor in distributed teams, such as offshore team members. They may struggle to understand very high level test objectives, rather than detailed procedural test steps.

This is where HP® Sprinter can be a key differentiator in your success. Sprinter is a relatively new addition to the traditional Quality Center testing tool. Much like the Manual Runner capability already within ALM, Sprinter is a manual testing solution that runs outside of ALM. What makes Sprinter different is the ability of the tool to capture your testing actions and results while you are doing your test. For example, if you were to run your generic *Access Account Screen* test in Manual Runner, ALM would keep track of your pass and failed steps, but wouldn't know how you accomplished the actual steps. What Sprinter is able to do is to capture your keystrokes and keyboard entries, so that while you are testing every action is being recorded along with the screens. Where the real power comes in is after you complete the test. You are able to view every step that you completed during the test and then you can push those steps to ALM as either a manual or an automated test script. What this does for teams is provide the ability to be more “free” with test case definition and then capture the detail during execution for future regression testing, and automation. Sprinter has many more advantages, that we just don’t have the time to address here, but what Sprinter can do is bring detail to agile.

Agile Defect Management

In agile, the days of a defect review or triage board are no longer needed. Testers are now expected to get up out of their seats, walk over to the developer, and talk. Defects don’t have to be triaged, they just become tasks on the story that need to get completed before the story can be marked as Completed. Just as with the Requirements module, it is important to build a custom defect form based on the fields that are important for your team to track agile defects. The fields, lists, and behaviors require planning and development to make the defect record simple and effective.

As I described earlier, defects aren’t discovered within a testing phase, they are either tasks to user stories or they become stories themselves. How you manage them is based on where you are in the Sprint they were discovered. Defects that are discovered during a sprint are always just tasks on the story. It doesn’t become a defect story until it can’t be resolved before the sprint completes. In that case, the defect becomes a story on the product/team backlog. It gets prioritized along with all other stories by the Product Owner and finds its way into some future sprint.

This may all sound very trivial, but believe me when I tell you that this takes some real thought and planning to make sure that the defect module behaves the way you desire. If you are using HP® Agile Manger, you will need to make sure you have adequate integration between the tools to insert the defect in the proper backlog. You will then need to know what you are going to do to establish a final disposition for the defect record in ALM. We recommend that the defect record in ALM close as Deferred at the point it is moved into HP® Agile Manger. Defects that are resolved during the course of a sprint are closed as Resolved. This enables the tracking of good metrics.

Agile Reporting

After spending all of the time getting the project to align better with agile, don't forget to create meaningful graphs and reports that are aligned to agile. I can't count the number of times I have taught ALM and Quality Center to clients, and even my own employees. What I often see is teams trying to use reporting developed from the Excel version of their test cases created years prior. They attempt to fit their old reporting ways to their new testing methodology. I also see teams struggle to meaningfully pull information from ALM into paper test reports. What I always encourage teams to do is to align their reporting to the way testing is supposed to happen in the tool. The percentage of test cases executed doesn't really tell you anything! What you should be reporting on is the number of requirements that are passed or failed, and maybe the number of test instances that have passed and failed. What we are trying to accomplish with testing is verifying the requirements have been met, so I don't understand why they only focus on execution metrics! It doesn't matter if you execute 1,000 tests if your requirement is failed. It doesn't matter if you planned to execute 10 tests and you only complete 9. What matters is whether or not the testing covered all of the requirements, and if they all passed.

When you move into an agile project, then the reporting focus changes to stories that are pass or failed within a specific sprint. Your testing horizon is two weeks rather than an entire release. Make sure that you are defining metrics that help you manage testing against a story within a given sprint. Yes you will want to be able to roll up the individual sprint reports into the overall release, but that is just a release version of the sprint report. Create dashboards for the individual sprints and a dashboard for the overall release. It will help you manage testing more effectively.

One additional capability of ALM is the capability to create what are called Project Reports. The client has the ability to generate branded test reports right out of ALM and consume them as a web page, a PDF, or as a MS Word document. You can set the style, contents, and branding to match the existing company templates. This enables the team to produce test reports as a direct output of ALM rather than copy and pasting data into a document. The report could have multiple summary sections that are pulled directly from ALM data, such as requirements coverage, test execution, and defect data. It can also embed the graphs you define into each section of the report, making a truly custom report. An even more agile approach would be to generate the report only when requested. Because you can set a filter to select the report for a specific sprint or release, you could generate the report on demand.

Integrating HP® ALM with Agile Tooling

One of the greatest efficiencies to be gained is the implementation of a highly integrated suite of agile tools. There are a variety of tools in the marketplace that can provide efficiency in the agile space. There are distinct suites of tools that focus on efficiently managing the artifacts and work streams of the agile process. HP® Application Lifecycle Management integrates with a variety of agile and development solutions.

- Agile Suite – end-to-end agile management, including quarterly release management, iteration (sprint) management, and basic artifact management (e.g. Epic, Feature, User Story, and Tasks).
- Architecture Suite – management of architecture design, including conceptual, logical, and physical designs.
- Testing Suite – full end-to-end agile test management, including test case development and execution using a 100% automation strategy.
- Development Suite – fully integrated development suite of tools that focused on delivering continuous integration.
- Release & Deployment Suite – automated release and deployment of code into controlled environments (e.g. test, staging, disaster recovery, and production).

These state-of-the-art suites of tools promise high efficiency and are designed to enable the agile practitioner to be more effective. A highly ambitious undertaking for an organization is the full integration of each suite of tools using a synchronization solution such as Tasktop™ Sync. With integration comes simplicity for the consumer, but high complexity for the team that has to manage the integration. Imagine having a user story that is assigned to a specific sprint automatically show up not only on the developer's environment, but also in ALM. Integration enables the seamless movement of artifacts through the agile lifecycle without forcing teams to take manual steps to move artifacts around. If you have used Quality Center for any period of time you cringe when you think about exporting requirements from Microsoft Excel into QC. It is tedious and time-consuming work that is critical to ensure end-to-end traceability. With synchronization you can eliminate that manual step and have the stories automatically move from the agile suite to the testing suite and have statuses automatically update in both solutions.

I would like to provide a caution about synchronization. It is not free. There is a cost associated with developing and implementing synchronization. License costs for the synchronization layer can be costly. The technical knowledge to complete the integration may require some high end consulting. HP® has included Application Lifecycle Intelligence (ALI) as part of the ALM platform, and it continues to improve and include more system integrations. It may be the solution you need to integrate your agile enterprise.

Summary

Whether you're new to agile or have been doing agile in your organization for years, you can always improve how you leverage HP® Application Lifecycle Management. We have discussed some recommendations and best practices we have discovered supporting our clients that we sincerely hope you will find useful. While this paper focused on HP® ALM, the same principles would apply to the Quality Center edition of the solution. Of course, some of the features in ALM are not fully available in QC, but the majority of the best practices would still apply. As I often say:

"Technology only rates the speed at which you get to the crap you produce"

If you have bad practices and you attempt to put them into ALM or any other tool, you will only realize that you have faster crap. Make sure that you are challenging the practices that you have in an unwavering desire to continuously improve. Let HP® ALM become the foundation for your new agile best practices.

About the Author

Brian Copeland is currently the Practice Director for Northway Solutions Group. With nearly 30 years of senior level experience in the software development industry specializing in organizational transformation and development, Brian has been instrumental in the testing of critical business systems, from mission critical applications to commercial software.

References

Leffingwell LLC. Copyright 2014, Scaled Agile Framework, http://scaledagileframework.com

About Northway Solutions Group

Northway Solutions Group is a HP Elite Solutions Partner and Reseller for HP Software products Northway employs certified consultants with real-world experience who provide long-term solutions to the toughest business challenges. This includes providing training and implementation service of HP Software products

Contact Northway

9005 Overlook Blvd
Brentwood, TN 37027 USA
Phone: 866.611.8762
Web: www.northwaysolutions.com
Email: Info@northwaysolutions.com