412TW-PA-19145

A CONCEPTUAL FRAMEWORK FOR FLIGHT TEST MANAGEMENT AND EXECUTION UTILIZING AGILE DEVELOPMENT AND PROJECT MANAGEMENT CONCEPTS

4
1
2
T
W

AUTHOR: CRAIG A. HATCHER

AIR FORCE TEST CENTER
EDWARDS AFB, CA

March 2019

Approved for public release; distribution is unlimited.
412TW-PA-19145

412TH TEST WING
EDWARDS AIR FORCE BASE, CALIFORNIA
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

[DISTRIBUTION STATEMENT A. Approved for public release; Distribution is unlimited 412TW-PA-19145]

A Conceptual Framework for Flight Test Management and Execution Utilizing *Agile* Development and Project Management Concepts

By Craig A. Hatcher

3/26/19

Table of Contents

Table of Figures

Introduction

Over the years the 412th TW have been attempting to find ways to understand the capacity to accomplish work. Key questions that need to be answered are the following:

- How much work can be accomplished during a period of time?
- Can more work be taken on without impacting other work?
- Where are the bottlenecks that keep more work from being accomplished?

A key metric that must be in place to answer these questions is throughput. The question of how to measure throughput has been elusive. The work within an organization dedicated to aircraft system testing is very diverse and difficult to measure with a single metric. Even at the lower testing unit level, understanding throughput has not been fully achieved.

In addition, the 412th TW has been on a journey to improve project management practices in order to be more efficient and effective in the execution of test and delivery of products to the customer. The focus of improving project management has been to utilize project data to maintain the level of situational awareness required to effectively make decisions. The Test Management Group (TMG) adopted Theory of Constraints (TOC) Project Management as the primary methodology to provide the needed information in a consistent manner across the group. Although much has been gained, the current system is not functioning as well as it could in our environment.

The current system is based on the concept that a network of tasks can be defined with their associated resources to model the work in a project. This concept is the primary concept found in both TOC and traditional critical path project management. Early attempts to model flight test projects in this manner, such that the work defined provided insight into actual project progress, ran into difficulties during project execution. At that time, typical network models would include tasks such as the flight planning, the test, data processing, and data analysis. Resources would be assigned based on the planned test points to be monitored for a particular flight. However, during project execution test points would not be flown as planned and thus the resources required would not match what was in the project schedule. The schedule quickly failed to reflect reality and had to be constantly changed to remain relevant. In addition, the construction of project plans at this level was time consuming and required expertise in the project management tool to construct. To combat these problems, many projects have modeled flight test as a series of high-level tasks that last several weeks. In other words, projects have been modeled as simple blocks of work. Although this simplified the project modeling and eliminated the thrash of keeping the network up to date, the ability to understand project progress from reviewing the status of the project schedule was lost. In addition, such simplification eliminates the advantage of having a network-based schedule and tool.

In addition, tracking flight test project progress based on task completion can be deceiving. Just because flights have been flown and data has been analyzed does not necessarily mean progress has been achieved. Flight test by nature is exploratory and much is learned as the test program progresses. The work required to meet test objectives can change as more is learned. Tests have to be rerun due to

these changes and other issues that manifest themselves. Due to these dynamics, schedule progress based on task completion may not be accurate without continuously updating the schedule to reflect newly discovered work. Updating a project schedule to keep up with this dynamic is cumbersome.

The purpose of this paper is to outline how adopting an Agile methodology for project execution and management can provide a consistent methodology for understanding throughput at all levels of the test organization and a simpler, more natural way of measuring true project status based on progress toward product production. The methodology will also provide insight into where bottlenecks occur in the test process.

Introduction to Agile Concepts

The Agile approach is a throughput-oriented approach for delivering value to the customer as quickly as possible. It is really a management framework on which to hang familiar implementation practices. This approach was created by a group of software developers as a response to traditional development processes that were expensive, slow, and did not fit well with the software development environment. In the software development environment there is extreme business pressure to produce software quickly and cheaply in order to stay competitive and maintain/capture new market share. Traditional systems engineering approaches were too slow and expensive to be competitive in this market environment. In addition, the traditional approaches followed a paradigm that assumed scope could be defined up front, a plan could be put in place, and the plan could be executed with minimal changes. The reality experienced by software developers was that users could not articulate what they truly desired, programmers could not correctly interpret requirements based on documents alone, and the creative process of designing software was truly an iterative process. The result of utilizing traditional processes caused products to be slow to market and not provide the functionality that users really wanted. To combat these problems, software developers came together and developed the Agile Manifesto that provides a set of guiding values that served as a basis for the Agile methodology. The Agile Manifesto states the following:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

These values shape the way projects are structured, managed, and tracked.

Agile Principles

Twelve Agile Principles were developed to further define these overall values. They are the following:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in the development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for a shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them an environment and support they need and trust them to get the job done.
6. The most efficient and effective method of conveying information to and with a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the work not done – is essential
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Project Lifecycle

An Agile project lifecycle is very different from a traditional plan driven lifecycle. In a traditional plan driven lifecycle, requirements are defined up front. A master plan is developed to design, develop, and test the product. Changes to requirements and plan are strictly controlled through configuration management processes. Changes to requirements are resisted as a matter of policy. In contrast, an Agile program lifecycle embraces change and focuses on delivering value to the customer as early as possible. The Agile program lifecycle is structured as follows:

Project Planning

- Requirements are defined from a user perspective, usually in the form of user scenarios called user stories.
- User stories are prioritized and placed in a Backlog. A Backlog is simply a prioritized list of user stories.
- From the Backlog, user stories are assigned to a set of interim releases. Each release provides capability that will be useful to the end user and has enough functionality to be delivered and put into production. The goal of a release is to provide value to the customer as early as possible

and then to obtain feedback to incorporate improvements back into the product such that these are in place before final delivery.

    - Customers are allowed to change their mind from one release to the next in order to keep the value of the endeavor in line with business and market realities.

- Within each release, user stories are assigned to short duration time periods called Sprints.

    - Sprints are iterations that produce value to the customer.

        - User stories for the Sprint are assigned at the beginning of the Sprint per the current release plan and backlog.

        - Each Sprint produces a deliverable quality working product. It may take several Sprints to create enough value to provide a usable product to the customer for a release. Value accumulates incrementally as outcomes are committed to production.

        - Sprints are short (typically, 2 weeks to 1 month) timeframes where detailed design, development, and testing take place. In other words, a complete development cycle is contained in a single iteration. All Sprints haves the same defined length of time, (i.e. they are time-boxed).

- Since Sprint time periods are fixed then the time to complete a release can easily be calculated. If the number of Sprints is more that than two or three, then an additional sprint with unassigned user stories may be added to account for scope changes that may occur during the execution of the release. More on this topic will be discussed during the project execution section of this paper.

- Once release plans are defined an initial overall program plan and schedule can be constructed as the summation of all the releases. The overall program plan is a rough estimate of program length.

Project Execution

A project will execute one release at a time. To begin a release a project team will be assigned user stories for the first Sprint. They will work on developing the product for these user stories. At the end of the Sprint they will get credit for only those stories that were completed. Since progress is measured by completed functionality and the work associated with completing a user story includes all work required to design, construct, test, and document the functionality, no credit is given to partially completed stories. If product features are not completed in the allotted time they are not included in the product release of the Sprint, and are re-prioritized with new features for a future Sprint. Sprint outputs are normally reviewed by the customer to ensure the features being developed meet the need. User stories that are not completed are assigned to the next Sprint and other user stories from the

backlog are added to define the next Sprint. At this point other user stories can be added and prioritized within the backlog based on feedback from the user and development team. These scope changes are limited to changes to the work completed within the release. Other scope changes will be entertained at the end of each release.

Figure 1 illustrates this process:

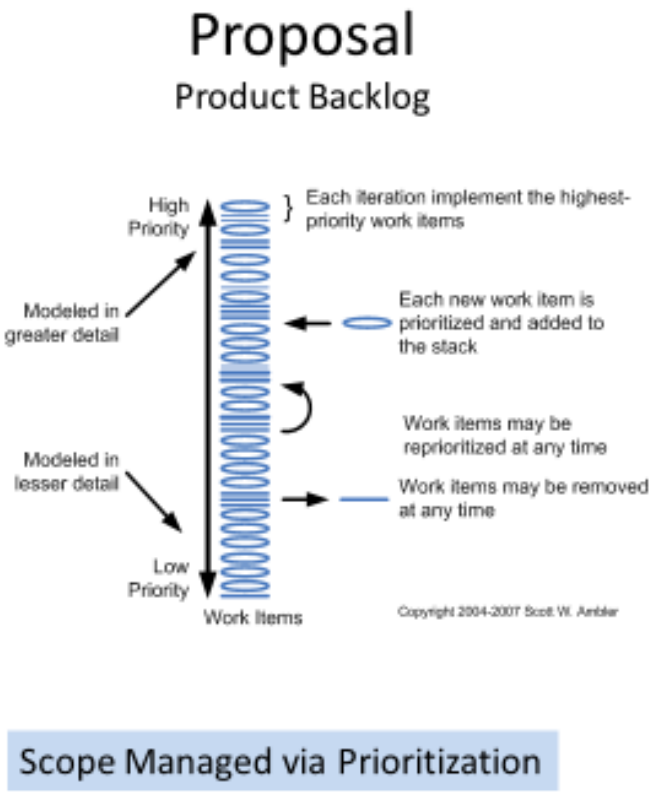


Figure 1 – Backlog Prioritization

After each Sprint, the project team evaluates their effectiveness and makes appropriate changes to both product and process. As a project progresses, the release plan is updated after each Sprint to reflect changes. This practice enables potential schedule slips and budget overruns to be predicted early and dealt with. The execution process is repeated until the release is complete. Once the release is complete the customer will review and provide feedback to the program team. At this point scope can be adjusted as needed. The backlog is adjusted appropriately and the next release begins. The overall project schedule is also adjusted based on scope changes and actual progress. This process continues until the product is deemed complete by the customer.

Agile Project Management Techniques

Estimating Techniques

Project Level Estimating

Agile estimating is done at a high level. Given the volatility of Agile projects, it does not make sense to spend an inordinate amount of time developing a detailed plan only to have it evolve significantly. Precise estimating is impossible. Therefore high-level estimating techniques are used as the basis for determining the overall project schedule. These techniques produce time estimate ranges. Ranges are provided to help communicate the true uncertainty of the project. For example, the final delivery of the software will be between 10 and 12 months from now.

Sprint and Release Estimating

Project volatility also makes detailed bottoms-up estimating for sprints and releases unproductive. Agile estimating, at this level, relies on comparative estimating. Comparative estimates provide a quick way to estimate sprints and therefore releases. Comparative estimates have been shown to be more accurate than bottoms up detailed estimates. A 1991 study by Vicinanza et al., "Software-Effort Estimation: An Exploratory Study of Expert Performance", showed that experts are more accurate in comparative estimation, vs. bottoms up.

Comparative estimating does not focus on activities (tasks), as with traditional project estimating, but rather focuses on outcomes instead (i.e. user story completion). In other words, estimates focus on throughput. Two key concepts underpin the basis for all comparative estimating. They are size and velocity.

Size

Size is a measure of the complexity of a user story. A key mechanism for determining size is the Story Point. A Story Point is a dimensionless unit of measure for expressing the overall size of a user story. To measure size, a baseline definition for a story point must be established. Typically, a typical user story is chosen that is easy for all to understand the amount of work required to complete it. This user story must be small enough to fit within the Sprint timeline. This baseline story is given a value, usually between 1 and 10. All other user stories are given a story point value based on their relative size to the baseline story point. All user stories must be small enough to be completed within a Sprint. If they are too large they must be broken down into smaller sub-stories. Since Story Points measure the relative size of a user story, they are a measure of all work required to produce a portion of product. In other words, a Story Point results in a unit of outcome.

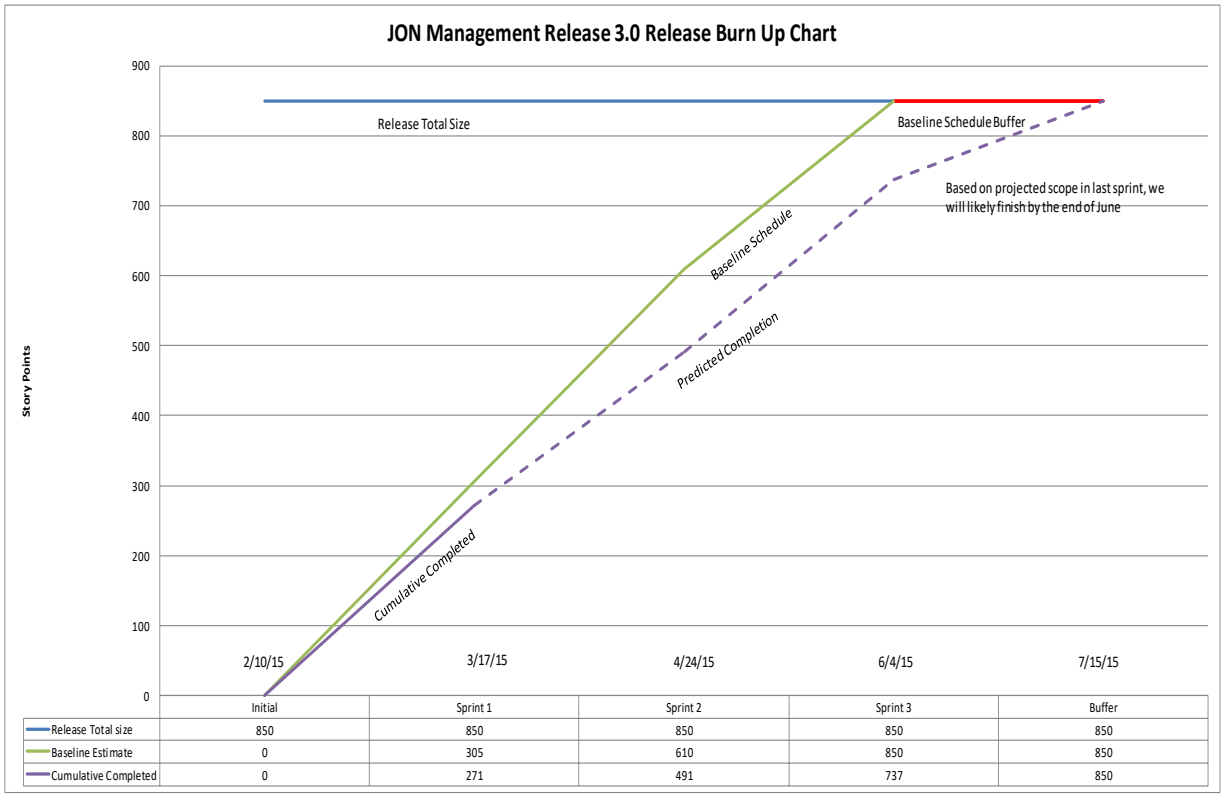
Velocity

Velocity is a measure of a team's rate of progress. It is the measure of throughput expressed in units per scheduled iteration. It is expressed in increments of product developed and completed during one

iteration (i.e. # of Story Points per iteration). The velocity of a team is determined based on experience. Sprints are planned around the number of story points that can be completed during the Sprint length. Over a period of time, a team can begin to understand their throughput by understanding how many story points, on average, they can complete per iteration. The velocity of a team is the metric for monitoring throughput. When there is not a track record of velocity, velocity can be obtained by planning out the number of user stories, with associated story points, that the team believes it can complete in the first iteration. The number of story points actually completed can be monitored until the velocity of the team can be fine-tuned (normally after three iterations). Once velocity is established, determining the number of Sprints required to complete the current scope of work can be easily estimated.

Release Scheduling and Status Tracking

All project progress is based on the number of story points completed. Based on Story Point Estimates and Velocity, a Release Burn Up chart can be constructed that tracks the total number of Story Points in a release and a time-phased plan of completing Story Points. Actual progress can also be plotted during execution such that it can be compared to the original estimate per the release plan. Scope increases can also be shown by adjusting the Release total size (total number of Story Points). The release end date can be predicted based on an average of past Sprint performance. Figure 2 illustrates the release schedule.



	Initial	Sprint 1	Sprint 2	Sprint 3	Buffer
Release Total size	850	850	850	850	850
Baseline Estimate	0	305	610	850	850
Cumulative Completed	0	271	491	737	850

Figure 2 – Release Burn Up Chart Example

Detailed Progress Tracking

During Sprint execution detailed progress can be tracked by monitoring user story completion. Software tools exist that enable user stories to be tracked within a Sprint with associated status. Figure 3 illustrates how the status of stories can be tracked.

Agile Product Backlog

Task Name	Story	Sprint Ready	Priority	Status	Story Points	Assigned to Sprint
Sprint 1	No	No	High	In Progress	24	No
Task 1	Yes	Yes	Medium	Complete	8	Yes
Task 2	Yes	Yes	Medium	Complete	16	Yes
Task 3	Yes	Yes	Medium	Complete	0	Yes
Sprint 2	Yes	Yes	Medium	In Progress	96	Yes
Task 4	Yes	Yes	Low	Complete	32	Yes
Task 5	Yes	Yes	Low	Complete	48	Yes
Task 6	No	No	Medium	Not Started	16	No
Sprint 3	Yes	No	Medium	In Progress	32	No
Task 7	Yes	No	Low	In Progress	8	No
Task 8	No	Yes	Medium	In Progress	8	No
Task 9	Yes	No	Medium	In Progress	16	No
Sprint 4	Yes	Yes	Medium	In Progress	64	Yes
Task 10	Yes	No	Low	In Progress	32	No
Task 11	Yes	Yes	Low	Complete	32	Yes
Task 12	Yes	Yes	Medium	Complete	0	Yes
Sprint 5	No	No	Low	Not Started	64	No
Task 13	No	No	Low	Not Started	48	No
Task 14	No	No	Low	Not Started	8	No
Task 15	No	No	Low	Not Started	8	No

Figure 3 – Backlog Tracking

To further understand the progress of a user story, Kanban Charts are used to track the progress within the development process. Figure 4 provides an illustration of a Kanban Chart.

Kanban Chart						
Sprint 2						
	Design	Code	Test	Integrate	Integration Test	Document
User Story 1	X	X	X	X		
User Story 2	X	X	X			
User Story 3	X	X				
User Story 4	X					

Figure 4 – Kanban Chart Example

Project Execution Governance

In an Agile paradigm, projects are controlled via daily standup meetings. During these meetings, all project personnel provide detailed status and work for the day is determined. Problems are raised and solved. Each member has the opportunity to speak. Project metrics are reviewed to provide an overall understanding of iteration progress. These meetings are time limited to last no more than 20 minutes.

Application to Flight Test Programs

The application of Agile principles and techniques can be quite easily used in a flight test environment. The following paragraphs will show what has to be done and how easily these concepts fit into the flight test paradigm.

Translated Agile Manifesto

To understand the potential fit of Agile in flight test I have translated the Agile Manifesto into the flight test paradigm.

"We are uncovering better ways of conducting flight test by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Delivered results over comprehensive documentation
- Customer collaboration over PID/SOC negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

The question is whether these values ring true in our culture? I believe they do.

Translated Agile Principles

The fundamental principles that underpin agile software development can be easily translated into the test environment. The following are the 12 Agile Principles translated into the flight test environment:

1. Our highest priority is to satisfy the customer through early and continuous delivery of test objectives.

2. Welcome changing requirements, even late in the in the test program. Agile processes harness change for the benefit of the weapon system/warfighter.

3. Deliver test objective results useful to the customer frequently, from a couple of weeks to a couple of months, with a preference for a shorter timescale.

4. System Program Offices (SPOs) and testers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them an environment and support they need and trust them to get the job done.

6. The most efficient and effective method of conveying information to and with a test team is face-to-face conversation.

7. Delivered test objectives are the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, testers, and support infrastructure should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence, good test discipline, and good safety discipline enhances agility.

10. Simplicity – the art of maximizing the work not done – is essential.

11. The best test planning emerges from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The question is whether we value these principles? Do these principles provide the foundation for a better way of doing business?

Agile Flight Test Program Lifecycle

The Agile program lifecycle can be adapted to flight test programs. However, moving to an agile project management paradigm can have significant impacts on how business is conducted, particularly in the test planning and test reporting phases of a test program. It is very possible that moving to a pure Agile model may not make sense in the test environment. Some hybrid approach may fit better or may be a palatable first step in the transition process.

Requirements Definition

Test objectives are currently defined for a program in the Program Introduction Document (PID). In a test program, test plans are written to meet objectives. Tests are then planned, executed, data processed, and results analyzed. Results are then compiled for each test and analyzed to provide the information required to meet test objectives. In the Agile paradigm, **test objectives** are equivalent to a user story in software development. Completing a test objective provides usable information to the customer. In other words, **completed test objectives represent throughput.**

Agile Flight Test Lifecycle Models

The following are several proposed lifecycle models that could be implemented in flight test that conform to the Agile model to varying degrees.

Pure Agile Flight Test Lifecycle Model

The first model we will investigate is the implementation of a pure Agile lifecycle model for flight test. This model will illustrate what it would mean to apply all the Agile Principles to flight test. The key questions to answer after this model is described are: Could we work this way? What would have to change? Are these changes possible? If not, why not? Making this assessment of the pure Agile lifecycle model will help determine if it truly fits and serves as a foundation for evaluating other hybrid approaches.

Project and Release Planning

Project planning begins by developing a test plan just as it is done today. Acquisition programs define test objectives from which a test plan is written that defines the tests required to meet those objectives. A prioritized Backlog is developed from test objectives. Test objectives are prioritized and a set of Interim Releases is defined. Interim Releases are divided into Sprints. Test objectives are then assigned to Sprints. If test objectives are too large to fit in a Sprint, they are broken down into sub-objectives, prioritized, and assigned to Sprints. Based on the Backlog a high-level schedule is created and Interim Release plans are developed. Burn Up charts are developed for releases. Each release is designed to provide a deliverable product (Preliminary Report of Results (PRR)) that provides business value to the customer. The goal is to provide value to the customer as early as possible and then to obtain feedback to incorporate required changes (in the form of new test objectives or required retesting of existing test objectives) back into the Backlog as increased scope. Customers are allowed to change scope in order to keep the value of the endeavor in line with current System Program realities. The overall program schedule will be updated as required.

Sprint Planning and Execution

Releases are made up of Sprints that produce value to the customer. Test objectives for Sprints are assigned at the beginning of the iteration per the current release plan. Each iteration produces a deliverable quality product. It may take several iterations to create enough value to provide a usable product to the customer for a release. Value accumulates incrementally as outcomes are delivered. In

the test environment the deliverable product for a Sprint is based on what the customer desires. It could be one of the following:

- Raw data
- Processed data
- Analyzed data summarized into a deliverable quality documented results

Sprints are short (typically, 2 weeks to 1 month) timeframes. In software programs the detailed design; development, testing, and documentation all take place within the iteration. In other words, a complete development cycle is contained in a single iteration. In flight test this approach is equivalent to detailed test planning/safety planning, test execution, data analysis, and documentation of results for the typical situation where the tester is required to produce a report as a deliverable. The Sprint is a defined length of time, i.e. time-boxed. Within a Sprint test objectives are tracked for completion and Kanban charts are developed to track detailed progress of test objective completion. If test objectives are not completed they are not included in the product release of the Sprint, and are re-prioritized with new objectives for a future Sprint.

After each Sprint, the project team evaluates their effectiveness and makes appropriate changes to both product and process. At this time customers, also, have the opportunity to change requirements. New test objectives, existing test objectives, and test objectives that were not completed in previous Sprints are all prioritized within the Backlog in order to define the next Sprint within the release. Requirements changes are managed via prioritization with the customer. It is the responsibility of the customer representative and program manager to ensure requirement growth enhances business value while staying within cost and schedule constraints. After all Sprints are completed within a release a PRR is delivered to the customer.

As a project progresses, the release plan is updated after each Sprint to reflect changes. This practice enables potential schedule slips to be predicted early and dealt with.

Ramifications

Test Execution

Since the measure of progress is based on completed test objectives, test programs would be incentivized to conduct testing such that test results are produced and analyzed as data is produced. The concept of flying all test points and then analyzing the data would be discouraged since no throughput could be derived by simply flying. This concept is currently considered best practice (although not always followed) because it helps avoid re-testing due to issues or problems being found in the data.

Interim Releases

To enable results to be provided the customer at the end of each release, PRRs would be required in some form. This concept is based on a key Agile principle of providing the customer a product early and often in order to ensure the product meets their true needs. In this pure Agile paradigm, PRRs would be delivered to the customer reporting results of completed test objectives. The question must be asked is: Are there any risks involved with providing PRRs of completed test objectives? Could our conclusions evolve even though dedicated testing of those test objectives is complete? If so, what could be done to mitigate the risk?

Test/Safety Planning

In a pure Agile paradigm, detailed test and safety planning would be right before each Sprint (or one or two weeks before a Sprint) instead of for the whole project up front. Pure Agile methodology attempts to spread planning throughout the project lifecycle in order to make detailed plans as accurate as possible (Why make up detailed plans if they are going to change radically as time passes?). Current processes would have to be re-designed to be agile enough to accommodate this new paradigm. Lengthy approval cycles could not support it. In this paradigm test plan/safety plan approval would more than likely have to be delegated down to the testing organization level for all but the most dangerous testing. Safety personnel may have to be imbedded within each testing organization. The key question that we have to ask is whether such an approach could be implemented while still allowing the appropriate technical rigor to be applied. Could our processes be redesigned to conduct an initial review of the test plan early and conduct more detailed reviews during Sprint planning? Agile methodologies must not compromise good engineering practices.

Test Reporting

In this pure Agile paradigm, report quality sections of a test report based on test objectives completed would be produced during each Sprint. At a release milestone, a PRR covering completed test objectives would be delivered to the customer. Part of the Sprint before a release would be dedicated to consolidating the interim report. Another approach would be to construct the outline of the report in the first iteration and add content during each iteration such that the release report would not require much consolidation before delivery. Similarly the final test report would be a consolidation of release reports. To accommodate this different approach, test reporting processes would have to be modified. Lengthy review processes do not fit well with this approach. Interim report approval would likely have to be delegated to the testing organization level for all but the most critical/politically charged reports. Minimal report editing during iterations would be required as well as minimal editing of the interim report. Delegating approval to the testing unit level would make the testing organization director responsible for the product without personally being involved in the production of it. The testing organization director would still be liable. This concept is a significant departure from the current paradigm. Following the pure Agile paradigm would require delegation to and trust in the testing unit leadership to ensure a quality product is produced. Agile thinking puts as much trust in the project team as possible and minimizes bureaucratic oversight.

Personnel Usage

Release planning by test objective would allow a test program to release resources (test engineers) once their discipline is no longer needed to fulfill a test objective. In addition, resources (test engineers) could be added for the particular release cycle that focused on their test objectives.

Hybrid Agile Flight Test Lifecycle Model

There are several variations to the Pure Agile Model that could be implemented to make the model better fit what we are willing or could actually implement.

Variation 1

Releases could only be delivered internally instead of to the SPO. Interim flight test reports could then be reviewed by technical experts and management in parallel with project execution. Feedback could be provided before the final test report is completed.

Variation 2

Instead of producing a deliverable quality piece of a report tied to the objectives of a Release, a less formal delivery of the data results could be substituted. After project execution or at each release a Sprint could be added to formally write the test report, or PRR. This variation would allow much of the existing reporting process to stay intact without major modification.

Agile Flight Test Program Management Techniques

Estimating Techniques

The definition of a Story Point can be determined by defining a baseline story point. A baseline story point would be a well understood test objective that could be executed during one Sprint. This baseline story point (standard) would be given a value between 1 and 10. All other test objectives would be compared to the standard and given a value based on their relative size to the standard. If a test objective is too large to fit in one Sprint then it would be broken into sub-objectives such that each sub-objective could be executed within a single iteration. There will be times that a test objective cannot be broken into small enough sub-objectives to fit in a Sprint. This happens in software projects also. Techniques exist to deal with this aberration, however the usage should be minimized so that progress is based primarily on throughput.

Pure Agile Approach

In a pure Agile approach, a story point would consist of all the effort to plan, execute, and deliver the product required to meet the test objective. For example, the standard definition for the baseline test objective could consist of the work required to plan, test, analyze, and document the results of the tests required to meet the objective. The definition would be different if we were not required to deliver a report.

Alternative Agile Approaches

If one of the Agile variants discussed above is chosen different story point definitions would be required depending on the phase of the program. For example, if test planning continued as it is today, a different story point definition would be required for the test planning phase versus test execution. The same concept would apply to the test reporting phase if it were left intact. Another option would be to conduct test planning and/or test reporting as traditional projects and run test execution as an Agile project.

Scheduling

Projects would utilize a master schedule for tracking the overall project. Release Burn Up Charts would be utilized to track each release. Within a Sprint, the test objective completion would be tracked and Kanban charts similar to Figure 4 would be constructed.

Kanban Chart						
Sprint 2						
	Detail Plan	Schedule	Test	Produce Data	Analyze Data	Document Results
Test Objective 1	**X**	**X**	**X**	**X**	**X**	**X**
Test Point 1	X	X	X	X	X	
Test Point 2	X	X	X	X	X	
Test Point 3	X	X	X	X	X	
Test Objective 2	**X**	**X**				
Test Point 1	X	X				
Test Point 2	X	X				
Test Point 3	X	X				

Figure 5 – Flight Test Kanban Example

Project Execution Governance

Projects would conduct daily stand up meetings with the test team to review status, resolve issues, and make task assignments for the day.

Throughput Metric

Within each project a throughput metric is based on the number of story points completed during each Sprint. To develop a CTF or TW throughput metric two things must be standardized.

Sprint Length and Start Date

The first is the length of a Sprint with a standardized start date. A Sprint length of 1 month is recommended with the start date set at the beginning of the month.

Baseline Story Standardization

A standard baseline test objective with associated story point value would have to be established to normalize a throughput metric across the organization. All projects would determine the size of their test objectives based on this standard baseline test objective. If non-test execution work is tracked with story points this work would also be estimated based on the standard baseline test objective.

Ramifications

A standard measure for throughput based on completed story points would provide testing units and the test organization a normalized way to understand throughput and work capacity. Projects will understand their capacity in terms of throughput. Testing units and the test organization will understand capacity in terms of throughput. Data will be present at the lower levels (i.e. Kanban charts) that will enable the identification of bottlenecks. Throughput can be increased if bottlenecks are eliminated.

Conclusion

Using an Agile paradigm to structure flight test programs is feasible, but requires decisions to be made as to what extent it can be implemented. A pure Agile model would require a re-design of how we accomplish test planning and test reporting. A hybrid approach may, however, be a better fit or, in any case, be a good first step along the path. We should not ignore the potential of going to a pure Agile model. We also should not fully implement a pure Agile model if it would compromise the rigor required to test weapon systems. A pure or modified Agile approach would simplify project planning and allow scope changes to be seamlessly integrated into our projects. Story Point (test objective) Burn Up metrics would provide a better measure of progress than current task completion (TOC schedule) metrics show. Testing units are already using burn down charts to better communicate progress during execution via test point completion. Burn Up charts based on Story Points (test objectives) provide a true throughput metric that can measure the impact of process improvements. Product Backlog Status and Kanban charts would provide the capability to drill down to get detailed status. Interim releases would ensure test reports are written as the project is executed (whether or not we decide to deliver interim reports to the customer). Analyzing data and writing the report as a project is executed has been shown to be a best practice.

Much can be gained by moving to an Agile paradigm if we are willing to incorporate the required changes. To this point, this paper simply presents a theoretical argument. The next step required to make a decision would be to prototype these concepts on a real project. Results from prototyping would validate the concept and provide direction on what, currently unforeseen, adjustments would have to be made.

Bibliography

Goodpasture, John, C. (2010). Project Management the Agile Way. Ft. Lauderdale, FL: J. Ross Publishing, Inc.

Cohn, Mike (2006). Agile Estimating and Planning. Upper Saddle River, NJ: Pearson Education, Inc.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
03/28/2019	Technical Paper	15-17 May 2019/ITEA Conference

4. TITLE AND SUBTITLE

A Conceptual Framework for Flight Test Management and Execution Utilizing Agile Development and Project Management Concepts

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Craig A. Hatcher

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)

812 TSS/ENTI
307 E. Popson Ave
Edwards AFB CA 93524

8. PERFORMING ORGANIZATION REPORT NUMBER

412TW-PA-19145

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

412th Test Wing
195 E Popson Ave
Edwards AFB CA 93524

10. SPONSOR/MONITOR'S ACRONYM(S)

N/A

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release A: distribution is unlimited.

13. SUPPLEMENTARY NOTES

ITEA/ Las Vegas NV / 15-17 May, 2019

14. ABSTRACT

Tracking schedules for flight test is difficult in a traditional network based scheduling paradigm. Traditional network based scheduling relies on being able to lay out a plan and follow it during project execution without many changes. This paradigm begins to struggle when the plan changes often. Flight test is a very dynamic endeavor. Scope changes frequently based on what is learned in test. Data is also collected at different times than planned due to the realities of test execution. As a result schedules are quickly out of date from both a time and resource usage standpoint. The software industry has moved away from network scheduling techniques toward Agile techniques and processes to manage projects. The reason for this paradigm shift is due to the volatile nature of software projects. User needs and scope changes often as users refine what they truly need. Schedules become obsolete quickly. Agile methods have been invented to minimize upfront planning and embrace scope changes as necessary during the project lifecycle. Flight test and software development projects share similar characteristics. They both are very volatile and require constant changes. This presentation will outline a conceptual framework that describes how Agile techniques, concepts, and processes can be used to monitor and execute flight test. In addition, this presentation will show how Agile techniques provide a throughput metric that can provide the basis for understanding the capacity of an organization.

15. SUBJECT TERMS

Agile, Throughput, Capacity, Story Point, Burn Up Chart, KANBAN chart

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					412 TENG/EN (Tech Pubs)
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	None	23	**19b. TELEPHONE NUMBER** *(include area code)*
Unclassified	Unclassified	Unclassified			661-277-8615