Software Engineering Institute

Agile Metrics: Progress Monitoring of Agile Contractors

Will Hayes
Suzanne Miller
Mary Ann Lapham
Eileen Wrubel
Timothy Chick

January 2014

TECHNICAL NOTE
CMU/SEI-2013-TN-029

Software Solutions Division

http://www.sei.cmu.edu

Carnegie Mellon University

Copyright 2014 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution except as restricted below.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0000811

Table of Contents

Acknowledgments **vii**

Executive Summary **ix**

Abstract **xiv**

1 How to Use this Technical Note **1**
- 1.1 Scope 1
- 1.2 Intended Audience 1
- 1.3 Key Contents 2

2 Foundations for Agile Metrics **4**
- 2.1 What Makes Agile Different 4
  - 2.1.1 Agile Manifesto 4
  - 2.1.2 Comparison to Traditional Approaches 5

3 Selected Measurement Considerations in DoD Acquisition **6**
- 3.1 Specific Governance Requirements 8
  - 3.1.1 Earned Value Management; Cost and Schedule Monitoring 9
  - 3.1.2 Delivery and Quality Monitoring 9
- 3.2 Tools and Automation in Wide Usage 10

4 Agile Metrics **11**
- 4.1 Basic Agile Metrics 12
  - 4.1.1 Velocity 12
  - 4.1.2 Sprint Burn-Down Chart 13
  - 4.1.3 Release Burn-Up Chart 14
- 4.2 Advanced Agile Metrics 15
  - 4.2.1 Velocity Metrics 15
  - 4.2.2 Flow Analysis 20

5 Progress Monitoring in Acquisitions Using Agile Methods **25**
- 5.1 Software Size 25
- 5.2 Effort and Staffing 27
- 5.3 Schedule 28
- 5.4 Quality and Customer Satisfaction 29
- 5.5 Cost and Funding 30
- 5.6 Requirements 31
- 5.7 Delivery and Progress 32
- 5.8 Agile Earned Value Management System 32

6 Conclusion **35**

Appendix A Past Publications in the SEI Agile Series **37**

References/Bibliography **39**

List of Figures

Figure 1: Audience 2
Figure 2: Agile Manifesto 4
Figure 3: The Defense Acquisition Management System [DoD 2007] 6
Figure 4: Sample Velocity Column Chart 12
Figure 5: Sample Sprint Burn-Down Chart 13
Figure 6: Sample Release Burn-Up Chart 14
Figure 7: Sample Defect Burn-Up Chart 17
Figure 8: Coefficient of Variation Example 20
Figure 9: Sample Stacked Column Chart 21
Figure 10: Sample Cumulative Flow Diagram 22
Figure 11: Sample Cumulative Flow Diagram – Fixed Pace of Work 23
Figure 12: Sample Cumulative Flow Diagram – Illustration of Workflow Issues 24
Figure 13: Estimating Workload for a Single Sprint 26
Figure 14: Notional Illustration of Agile Versus Waterfall Integration/Test Efforts 28
Figure 15: Many Quality Touch-Points in Agile Development 29
Figure 16: Defect Analysis Using a Cumulative Flow Diagram 30
Figure 17: Accumulation of Earned Business Value [Rawsthorne 2012] 34

List of Tables

Table 1: Sample Regulatory References 8

Acknowledgments

The authors wish to thank the many members of the Agile Collaboration Group who continue to offer their experiences, insights, and advocacy to the task of advancing the implementation of cost-effective methods for development and sustainment of software-reliant systems within the Department of Defense. Several of this group generously gave their time to participate in interviews about their experiences. Their insights and anecdotes added greatly to our understanding.

We wish to specifically acknowledge the contributions of:

Mr. John C. Cargill, MA, MBA, MS
Air Force Cost Analysis Agency

Mr. Russell G. Fletcher
Vice President, Federal Agile Practice Manager
Davisbase Consulting Inc.

Ms. Carmen S. Graver
USMC

Mr. Dan Ingold
University of Southern California
Center for Systems and Software Engineering

Larry Maccherone
Director of Analytics – Rally Software

Carol Woody
Technical Manager, Cyber Security Engineering
Software Engineering Institute
Carnegie Mellon University

Executive Summary

Agile methods are seen by some as an effective means to shorten delivery cycles and manage costs for the development and maintenance of major software-reliant systems in the Department of Defense. If these benefits are to be realized, the personnel who oversee major acquisitions must be conversant in the metrics used to monitor these programs. This technical note offers a reference for those working to oversee software development on the acquisition of major systems from developers using Agile methods. While the primary focus of the technical note involves acquisition of software-reliant systems in a highly regulated environment, such as the U.S. Department of Defense (DoD), the information and advice offered can be readily applied to other settings where a development organization that employs Agile methods is being monitored.

The reader is reminded of key characteristics of Agile methods that differentiate them from the so-called traditional approaches. Previous technical notes in this series[1] provide a broad discussion of these topics [Lapham 2010, Lapham 2011]. In this technical note, we focus specifically on important considerations for the definition and use of metrics. Important differentiators that affect metrics include the following:

- There is a reliance on a “time-box approach” in place of the traditional phase-gate approach to managing progress. This means that the development organization is working to maximize valuable work performed within strictly defined time boundaries. The schedule remains fixed, and the work to be performed is the variable. This is in contrast to many traditional approaches where the period of performance may be subject to negotiation, while attempting to fix the scope of the work product(s) to be delivered. Also, the customer (or customer representative) is involved in frequent evaluations of deliveries – rather than successive intermediate work products that lead up to a single delivery of the software.
- The staff utilization approach favors a more uniform loading across the life of a program—rather than an approach that involves “ramping up” and “ramping down” staffing for various specialized job categories. This means that establishing the needed staffing profile is a high priority from the start, and deferred decisions are potentially more costly. This also means that once a well-functioning development team is established, the pace at which it performs can be sustained over a longer period of time. Combined with a de-emphasis on phase-gate-management, this also means that greater continuity can be achieved. That is, there are fewer “handoffs” among specialized staff, where information may be lost or delays may be introduced as intermediate work products change hands.
- A focus on delivering usable software replaces the scheduled delivery of interim work products and the ceremonies that typically focus on milestones. This means that the customer (or customer representative) should expect to see working product, frequently. The planned interactions and demonstrations permit a disciplined process for changing the requirements and shaping the final form of the delivered product. Acceptance testing happens iteratively, rather than at the end.

---

[1] A complete listing of these publications is found in Appendix A

With such fundamental differences in the underlying operating model, there is great potential for miscommunication between an Agile development organization and the traditional acquisition organization. The experience of seasoned managers and many of the “rules of thumb” developed through hard experience may not apply with Agile methods. Metrics that help diagnose progress and status differ from established and familiar ways of working.

Acknowledging the regulatory and legal requirements that must be satisfied, we offer insights from professionals in the field who have successfully worked with Agile suppliers in DoD acquisitions. Strategies are summarized that fulfill the expectations of senior leadership in the acquisition process, with new approaches driven by this different philosophy of development. The reader is provided with examples of new metrics that meet the existing reporting requirements.

The hallmarks of measurement in Agile development include:

- **Velocity**: a measure of how much working software is delivered in each sprint (the time-boxed period of work)
- **Sprint Burn-Down**: a graphical depiction of the development team’s progress in completing their workload (shown day-by-day, for each sprint as it happens)
- **Release Burn-Up**: a graphical depiction of the accumulation of finished work (shown sprint-by-sprint)

Most of the measurement techniques employed in Agile development can be readily traced back to these three central tools. We offer specific advice for interpreting and using these tools, as well as elaborations we have seen and been told of by people who use them.

One of the most promising tools for metrics emerging from Agile development is the Cumulative Flow Diagram. This depiction of data shows layers of work over time, and the progression of work items across developmental states/phases in a powerful way. Once you learn to read these charts with the four-page introduction we offer at the end of Section 4, many intuitively logical uses should become apparent.

Based on our interviews with professionals managing Agile contracts, we see successful ways to monitor progress that account for the regulatory requirements governing contracts in the Department of Defense. The list below addresses ingredients for success:

- **Software Size** is typically represented in story points when Agile methods are used. This approach is supported by the decomposition of functionality from a user’s perspective—into *user stories*. Tracing these user stories to system capabilities and functions, a hierarchy within the work can be meaningfully communicated and progress monitoring based on delivered functionality will focus on utility and function—rather than proxies like lines of code or function points.
- **Effort and Staffing** must be tracked because they tend to be the primary cost drivers in knowledge-intensive work. Use of Agile methods will not change this fundamental fact, nor will it be necessary to make major changes to the mechanisms used to monitor progress. What does change, however, is the expected pattern of staff utilization. With the steady cadence of an integrated development team, the ebb and flow of labor in specialized staff categories is less prevalent when using Agile methods. In general, Agile teams are expected to have the

full complement of needed skills within the development team—though some specialized skills may be included as part time members on the team. Rules of thumb applied in monitoring this element of performance on a contract will need to be revised. The expectation of a slow ramp-up in staffing during the early phases of a development effort may be problematic, and plans for declining use of development staff during the last half of the program (when testing activities traditionally take over) will need to be recalibrated. Organizations may establish test teams to perform system testing or regression testing outside the context of the development team. We are planning for a focused paper on testing in the context of Agile development where this topic will be covered more fully—targeting FY14 as of this writing.

- **Schedule** is traditionally viewed as a consequence of the pace of work performed. In Agile development, the intent is to fix this variable, and work to maximize performance of the development team within well-defined time boxes. This places important requirements on stakeholders who must communicate the requirements and participate in prioritization of the work to be performed.
- **Quality and Customer Satisfaction** is an area where Agile methods provide greater opportunity for insight than traditional development approaches tend to allow. The focus on frequent delivery of working software engages the customer in looking at the product itself, rather than the intermediate work products like requirements specifications and design documents. A strong focus on verification criteria (frequently called “definition of done”) sharpens the understanding of needed functionality, and attributes of the product that are important to the customer.
- **Cost and Funding** structures can be tailored to leverage the iterative nature of Agile methods. Using optional contract funding lines or indefinite delivery indefinite quantity (IDIQ) contract structures can add flexibility in planning and managing the work of the development organization. A more detailed discussion of the considerations for contracting structures to handle this is the subject of an upcoming paper in the SEI series.
- **Requirements** are often expressed very differently in the context of Agile development—in contrast to traditional large-scale *waterfall* development approaches. A detailed and complete requirements specification document (as defined in DoD parlance) is not typically viewed as a prerequisite to the start of development activities when Agile methods are employed. However, the flexibility to clarify, elaborate and re-prioritize requirements, represented as *user stories*, may prove advantageous for many large programs. The cost of changing requirements is often seen in ripple effects across the series of intermediate work products that must be maintained in traditional approaches. The fast-paced incremental approach that typifies Agile development can help reduce the level of rework.
- **Delivery and Progress** monitoring is the area where perhaps the greatest difference is seen in Agile development, compared to traditional approaches. The frequent delivery of working (potentially shippable) software products renders a more direct view of progress than is typically apparent through examination of intermediate work products. Demonstrations of system capabilities allow early opportunities to refine the final product, and to assure that the development team is moving toward the desired technical performance—not just to ask whether they will complete on schedule and within budget.

Detailed discussions with graphical illustrations and examples provided in this technical note will lead you through lessons being learned by Agile implementers. New ways of demonstrating progress and diagnosing performance are offered, with a narrative driven by actual experience. Many of the explanations contain direct quotes from our interviews of practitioners who oversee, coach, or manage Agile development teams and contracts. Their field experience adds much depth to the information available from articles, books, and formal training available in the market.

Abstract

This technical note is one in a series of publications from the Software Engineering Institute intended to aid United States Department of Defense acquisition professionals in the use of Agile software development methods. As the prevalence of suppliers using Agile methods grows, these professionals supporting the acquisition and maintenance of software-reliant systems are witnessing large portions of the industry moving away from so-called “traditional waterfall” life cycle processes. The existing infrastructure supporting the work of acquisition professionals has been shaped by the experience of the industry—which up until recently has tended to follow a waterfall process. The industry is finding that the methods geared toward legacy life cycle processes need to be realigned with new ways of doing business. This technical note aids acquisition professionals who are affected by that realignment.

1 How to Use this Technical Note

This technical note is one in a series of publications from the Software Engineering Institute intended to aid United States Department of Defense (DoD) acquisition professionals. As the prevalence of suppliers using Agile methods grows, these professionals supporting the acquisition and maintenance of software-reliant systems are witnessing large portions of the industry moving away from so-called “traditional waterfall” lifecycle processes. The existing infrastructure supporting the work of acquisition professionals has been shaped by the experience of the industry – which up until recently has traditionally followed a waterfall process rooted in a hardware-centric approach to system development. The industry is finding that the traditional methods geared toward legacy lifecycle processes need to be realigned with new development methods that change the cadence of work, and place new demands on the customer. This technical note aids acquisition professionals who are affected by that realignment.

1.1 Scope

Our focus in this technical note is on metrics used and delivered by developers implementing Agile methods. In particular, we are concerned with Agile teams responding to traditional acquisition requirements and regulations. Explanations and examples provide focus on progress monitoring, and evaluation of status. We provide practical experience and recommendations based on lessons learned by knowledgeable professionals in the field, as well as authors of influential books and papers.

For the contract planning stages, we provide a means of evaluating the feasibility and relevance of proposed measurements. Illustrations of common implementations and the range of metrics available provide context to professionals unfamiliar with Agile methods. For those unfamiliar with Agile methods, we recommend earlier papers in this series as background [Lapham 2010, Lapham 2011].

During program execution, metrics support progress monitoring and risk management. Adequate scope and converging perspectives in the set of metrics enable timely insight and effective action. We provide insights into common gaps in the set of measures and effective ways to fill them.

Finally, evaluation of key progress indicators and diagnostic applications of metrics represent a common theme throughout the paper. Efficiently converting collected data into information that meets the needs of stakeholders is important to a measurement program. Novel implications of Agile methods and unique needs of the DoD environment set the stage for this focused discussion.

1.2 Intended Audience

Our intent is to support program and contract management personnel “working in the trenches.” A diverse set of interests are served by a paper that helps to span the (potential) gap between Agile developers and traditional acquirers.

Our primary audience consists of program management professionals involved in tracking progress, evaluating status and communicating to government stakeholders. Often the person charged with these responsibilities is a junior officer in one of the branches of the military or a civilian who holds a GS12 to GS14 rank within the acquisition organization. These people frequently must interact with representatives of the development organization to ascertain technical status, and then communicate that to leadership in the acquisition organization. These individuals often have training and experience in project management, and are well-versed in the rules and regulations that govern the acquisition process. However, many of these skilled professionals are not familiar with Agile development methods. As the graphic below depicts, these professionals sometimes find themselves in the middle – between an innovating supplier and an entrenched set of expectations that must be satisfied. The connections between the new ways of working and the old ways of doing business may not be readily obvious.

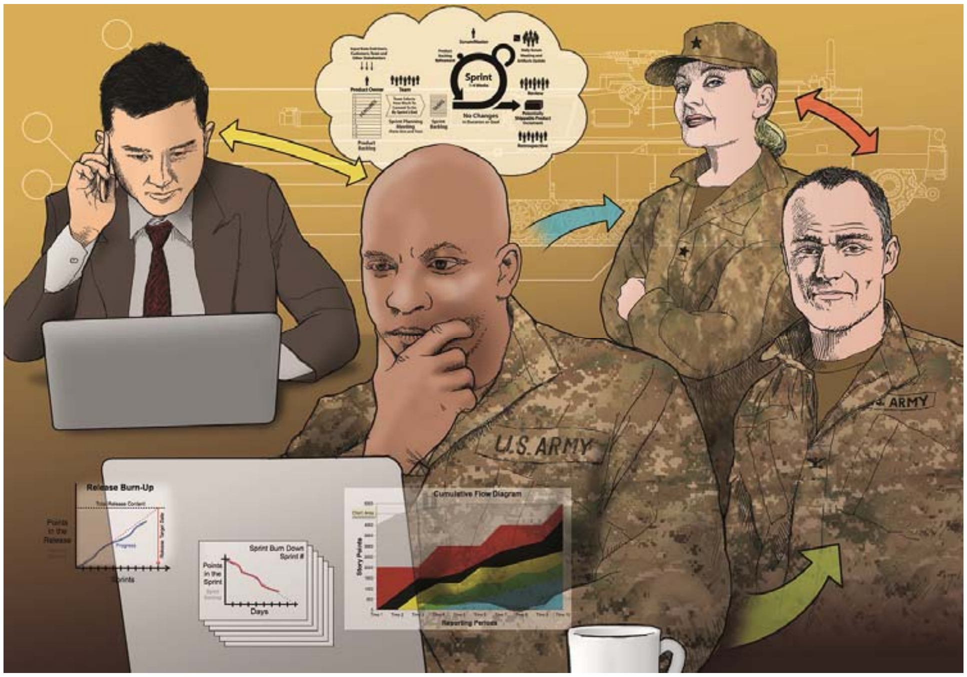

Sprint
No Changes
Product Backlog
Team
U.S. ARMY
ARMY
Release Burn-Up
Points in the Release
Sprints
Sprint Burn Down
Points in the Sprint
Days
Cumulative Flow Diagram
Story Points
Reporting Periods

Figure 1: Audience

1.3 Key Contents

In Section 2, *Foundations for Agile Metrics*, we provide a brief introduction to Agile methods, and comparisons to traditional methods that will aid the reader in understanding the remaining sections of the report.

In Section 3, *Selected Measurement Considerations in DoD Acquisition*, we describe the regulatory context in which Agile metrics must be implemented, along with a listing of categories of metrics that will need to be considered.

In Section 4, *Agile Metrics*, we begin the discussion of metrics that are specifically associated with the Agile methods, illustrating with examples, the metrics typically used by Agile development teams.

In Section 5, *Progress Monitoring in Agile Acquisitions Using Agile Methods*, we provide a detailed discussion of metrics used to monitor the ongoing work of Agile development teams.

Finally, Section 6 provides a summary of the paper.

2 Foundations for Agile Metrics

2.1 What Makes Agile Different

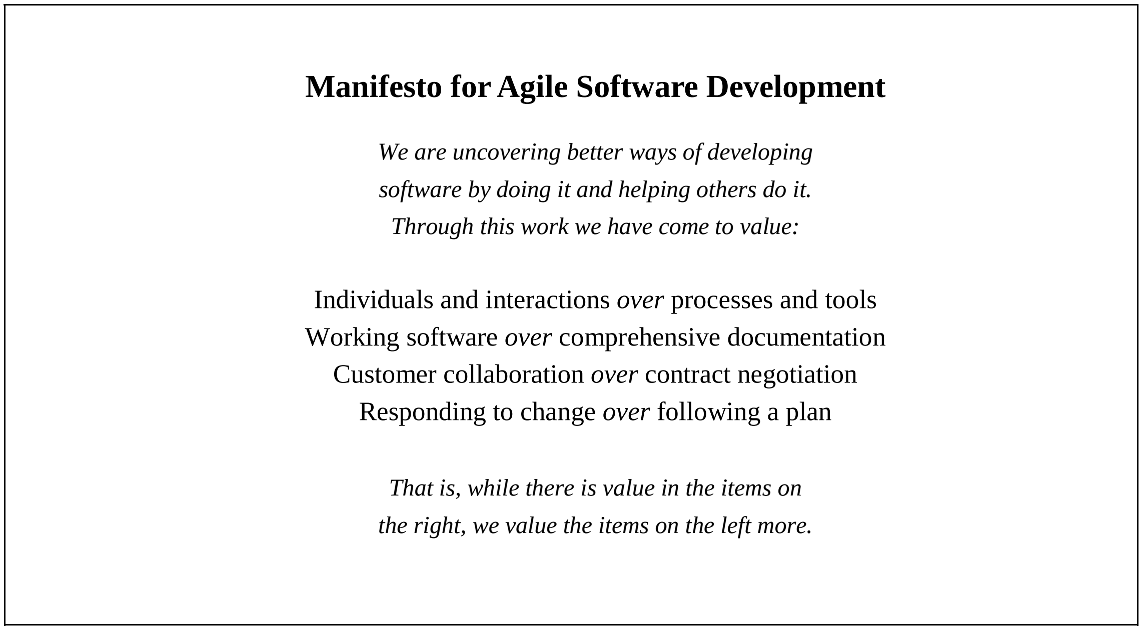
Interest in Agile methods for delivering software capability continues to grow. Much has been written about the application of methods with varying brand names. Lapham 2011 provides a frame of reference for discussions of metrics here:

> *In agile terms, an Agile team is a self-organizing cross-functional team that delivers working software, based on requirements expressed commonly as user stories, within a short timeframe (usually 2-4 weeks). The user stories often belong to a larger defined set of stories that may scope a release, often called an epic. The short timeframe is usually called an iteration or, in Scrum-based teams, a sprint; multiple iterations make up a release* [Lapham 2011].

The waterfall life cycle model is the most common reference when providing a contrast to Agile methods. The intermediate work products that help partition work into phases in a waterfall life cycle, such as comprehensive detailed plans and complete product specifications, do not typically serve as pre-requisites to delivering working code when employing Agile methods.

2.1.1 Agile Manifesto

The publication of the Agile Manifesto is widely used as a demarcation for the start of the "Agile movement" [Agile Alliance 2001].



Manifesto for Agile Software Development

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.

Figure 2: Agile Manifesto

The tradition of developers' irreverence surrounding the items on the right sometimes overshadows the importance of the items on the left. Failed attempts to adopt Agile methods have occurred through narrow emphasis on eliminating documentation, planning and the focus on process; without understanding ramifications for the items on the left. A great deal of discipline is required to consistently and successfully build high quality complex software-reliant systems.[1] Professional practices are required to demonstrate working software to the customer on a consistent basis, and to embrace technical changes that result from these frequent interactions.

2.1.2 Comparison to Traditional Approaches

Like many things, a strict interpretation of methods such as we find in textbooks will often bear only limited resemblance to the practice of people in the field. However, implementations of Agile methods generally differ from traditional approaches in the following ways:

1. Time is of the essence—and it's fixed. In place of strict phase-gates with deliverables at the phase boundaries, a fast-track *time boxed* approach to ongoing delivery of working software is typical. Rather than formally evaluating intermediate work products like comprehensive requirements or design documents, the customer is called upon to provide feedback on a potentially shippable increment of code after each iteration. Establishing this fixed cadence is considered essential.
2. Staff utilization is intended to be more uniform, rather than focused on phased use of specialty groups with formal transition of intermediate products from one state to another. (Consider: the "ramping up" of test resources traditionally seen toward the end of the coding cycle on a waterfall-based project.) The interdependence between different specialties (e.g., testing, information assurance, and development) is managed within each iteration, rather than using a milestone handoff to manage interfaces at a greater level of granularity. This self-contained team is empowered to make more of the key product-related decisions—rather than waiting for an approval authority to act during a milestone review.
3. An emphasis on "maximizing the work not done" leads to removing tasks that do not directly address the needs of the customer. While the Agile Manifesto clearly states a preference for working software over comprehensive documentation, this does not eliminate the need for documentation in many settings. Agile projects do not tend to be "document-focused," they are "implementation-focused."[2] Balancing the ethos reflected in the manifesto with the business needs of real customers may prove challenging at times – especially in a highly regulated environment.

These are the most prominent aspects of Agile methods that have an effect on the selection of metrics used in progress monitoring. The detailed explanations and illustrations provided in the remainder of this technical note can be traced back to one or more of these.

---

[1] A soon-to-be-published technical note in this series will focus on systems engineering implications for Agile methods.

[2] A soon-to-be-published technical note in this series will focus on requirements in the Agile and traditional waterfall worlds.

3 Selected Measurement Considerations in DoD Acquisition

In the SEI technical note *Considerations for Using Agile in DoD Acquisition* we discussed how Agile could be used within the DoD acquisition life cycle phases. We introduced the discussion as follows:

> *The DoDI 5000.02 describes a series of life cycle phases that "establishes a simplified and flexible management framework for translating capability needs and technology opportunities, based on approved capability needs, into stable, affordable, and well-managed acquisition programs that include weapon systems, services, and automated information systems (AISs)* [Lapham 2010]*."*

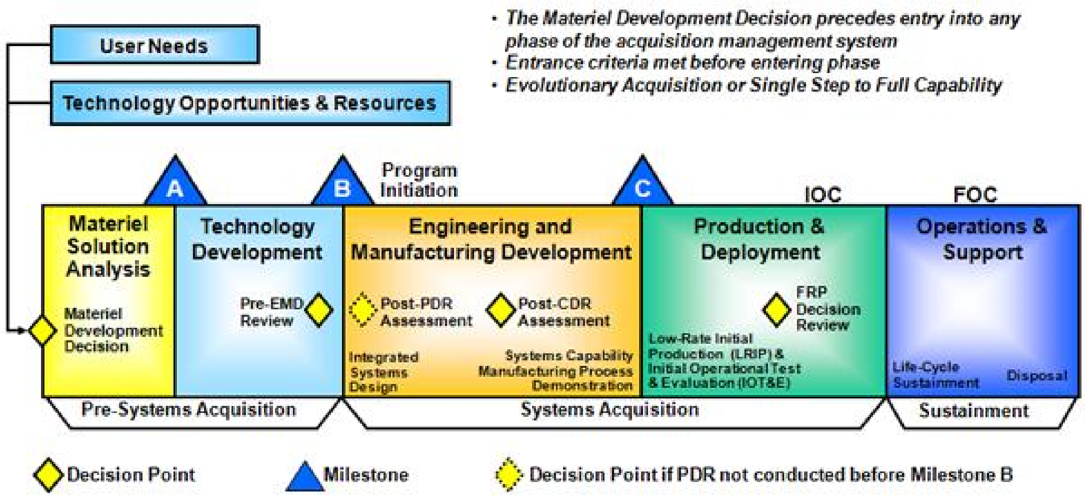

User Needs
Technology Opportunities & Resources
• The Materiel Development Decision precedes entry into any phase of the acquisition management system
• Entrance criteria met before entering phase
• Evolutionary Acquisition or Single Step to Full Capability
A
B
Program Initiation
C
IOC
FOC
Materiel Solution Analysis
Materiel Development Decision
Technology Development
Pre-EMD Review
Engineering and Manufacturing Development
Post-PDR Assessment
Post-CDR Assessment
Integrated Systems Design
Systems Capability Manufacturing Process Demonstration
Production & Deployment
FRP Decision Review
Low-Rate Initial Production (LRIP) & Initial Operational Test & Evaluation (IOT&E)
Operations & Support
Life-Cycle Sustainment
Disposal
Pre-Systems Acquisition
Systems Acquisition
Sustainment
Decision Point
Milestone
Decision Point if PDR not conducted before Milestone B

Figure 3: The Defense Acquisition Management System [DoD 2007]

Figure 3 shows how acquisition currently works[3]. In brief, there are five life cycle phases, three milestones and several decision points, used to determine if the work should proceed further. Within each life cycle phase, there are opportunities to develop and demonstrate the capability that is desired. As the acquisition progresses through the phases, options for system alternatives are systematically identified and evaluated. As a result of this analysis, some potential solutions survive, while others are eliminated. Eventually, a system configuration that meets stakeholder needs is specified, developed, built, evaluated, deployed, and maintained. The technical note goes on to say that there are opportunities to employ Agile methods within every lifecycle phase [Lapham 2011]. This raises a question that should be considered when employing an Agile method in any of the phases. What type of metrics should you be looking for or requiring from the contractor or organic staff in an Agile environment?

The simple answer is that the typical metrics that are always required by regulation (e.g. software size, schedule, etc.) should be provided. That's a simple answer, but what does this mean in an Agile environment? For instance, typical development efforts in the Technology Development and Engineering and Manufacturing Development phases may require the use of earned value management (EVM) [Fleming 2000]. This concept is foreign to a typical Agile implementation. The intent of the typical required DoD metrics needs to be met but Agile practitioners will want to favor the use of data that are naturally created during the execution of the Agile implementation. In the following paragraphs, we identify issues to consider in building an Agile program and its related metrics program. We go into more detail for some of these in subsequent sections.

---

[3] As of this writing, the publication process for an update to the 5000 series is underway, but not yet complete.

If the Project Management Office (PMO) is doing a request for proposal (RFP), no matter which phase, ensure that the RFP contains language that allows the use of Agile. In many instances, the traditional RFP language makes it difficult, if not impossible, to propose an Agile-based solution. In many instances, EVM metrics are required. These are typically based on a plan that is completed up front and rarely changes [Agile Alliance 2001]. If learning occurs that changes the plan, both revision of the plan and the EVM milestones require a rigorous change process. Agile implementations by definition allow and even embrace change. An Agile response to EVM requirements could provide considerable amounts of data taken from a typical Agile implementation, but that data would need to be interpreted and translated to compliance metrics. A proposal to the government should provide information on how that translation would be accomplished and if missing, the government should request that information.

In addition, we heard from several of our interviewees that the government personnel need to interpret the data provided to them. This in turn implies that the government personnel need to have an understanding of the Agile metric data and what it does or doesn’t mean. Some basic training in Agile concepts and definitions of Agile metrics would be in order if a government person was to attempt this type of interpretation. A starting point could be a basic overview of traditional software cost estimation followed by a description of story point estimation [Coelho 2012]. In addition, when shifting to using Agile methods, there is also a shift (from the Agile perspective) to focus on measuring business value delivered. This idea is critical in understanding the types of metric data typically collected when employing Agile methods [Hartman 2006].

Be prepared to mine and effectively use the metrics data that naturally occur in typical Agile teams. In the acquisition life cycle phases, metrics are requested to monitor the state of the program, that is, how much work is done, how much is left, what kinds of issues are there (number, severity, priority, etc.), and is the program on schedule. One interviewee pointed out that Agile programs are rich in metrics. Metric data is collected during each sprint or iteration. These occur frequently, usually every two to three weeks. Even if they are monthly, the resultant data is plentiful. Data collected includes burn down and burn up charts; estimates for each sprint; backlogs with priorities; continuous testing and integration metrics (percent passed, failed and related issues); number of defects both found during the sprint and after the sprint completed; and inspect and adapt data created during post sprint/iteration reviews. This abundance of data can provide transparency into the state of the program.

Take advantage of the transparency provided in Agile processes. Don’t wait for the metrics to come to you. Go look at what the Agile team is doing. Transparency is one of the key ideas behind Agile usage. All stakeholders can tell what is going on at all times. In some instances, the government is provided with access to the contractor’s tool set where all the metric data resides. In other instances, the government requests specific reports from the tool on a regular basis. More specific examples and discussion of progress metrics for programs using Agile methods will be provided in Sections 4 and 5.

Keep abreast of the changing policy environment that affects metrics collection and use. DoDI 5000.02 is currently undergoing an update. While this has yet to be published at the time of writing this paper, early indications are that the new 5000.02 contains multiple models for acquisition. The proposed Model 3, Incrementally Fielded Software Intensive Program, seems to be aimed for use with programs that use incremental and iterative methods such as Agile. A full review of this model needs to be accomplished once the updated 5000.02 document is signed and released.

3.1 Specific Governance Requirements

Contractual requirements for monitoring performance typically specify areas of focus for metrics. Government acquisition organizations operate within a framework of command media built to protect the integrity and consistency of their operations. The table below lists typical examples of requirements for metrics in contracts with two of the military services [USAF 2008, USA 2011].

Table 1: Sample Regulatory References[4]

USAF Software Core Metrics	Army Regulation (AR) 70-1 Army Acquisition Policy
Software size Software development effort Software development schedule Software defects Software requirements definition and stability Software development staffing Software progress (design, code and testing) Computer resource utilization	*Section 7-13 Software Metrics: PMs will negotiate a set of software metrics with the software developer to affect the necessary discipline in the software development process and to assess the maturity of the software product. At a minimum, the metrics should address—* • Schedule and progress regarding work completion. • Growth and stability regarding delivery of the required capability. • Funding and personnel resources regarding the work to be performed. • Product quality regarding delivered products to meet the user's need without failure, as reflected in associated requirements documents. • Software development performance regarding the capabilities to meet documented program requirements. • Technical adequacy regarding software reuse, programming languages, and use of standard data elements.

These requirements are written to allow flexibility in implementation—to fit the scope and nature of the contract at hand. For some of the metrics listed, Agile alternatives are available—such as software requirements definition and stability—and need to be specified in the contract if they are to be used. For other metrics—such as staffing levels—the traditional metrics used may be well suited, but the range of variation in the metrics over time may differ (successful Agile teams tend to have a more stable staffing profile over time than teams using traditional approaches). Finally, there may well be metrics needed in monitoring the progress of a contract that are beyond the focus of Agile methodologies. Managing different funding streams, specialized labor categories, and highly specialized technical performance or regulatory requirements are just a few such examples. The intersection of traditional monitoring approaches and Agile development approaches may be more challenging in these areas.

---

[4] These references are used for the purpose of illustration, not to imply that all regulations are encompassed by these examples. Such illustrative examples are also available from other services and agencies as well.

In the latter sections of this technical note, we describe specific examples of Agile metrics, and discuss important considerations for their use. Note that while Agile methods bring new ways of doing work, they are not intended to serve as a panacea for all concerns and priorities of the government acquisition environment (i.e., agile is not a silver bullet).

3.1.1 Earned Value Management; Cost and Schedule Monitoring

Nearly ubiquitous in the government acquisition realm, earned value management systems (EVMS) provide a means for monitoring cost and schedule performance, and support projections of future performance based on history of performance to date [EIA-748]. Alleman, in discussing Agile development in government acquisition, reminds us of a common reality in this realm:

> *Contract progress payments are based on "earned value" for the accounting period and therefore are considered our "rice bowl," something you simply do not mess with"* [Alleman 2003]

A formal treatment of the application of EVMS in the context of Agile methods is found in the 2006 publication on Agile EVM by Sulaiman, Barton, and Blackburn. These authors provide a mathematically rigorous basis for applying EVMS with Agile methods, then report on their experiences using the approach on two projects [Sulaiman 2006].

The use of EVMS is not required on all types of contracts; in particular, smaller efforts may tend to rely on alternate approaches. Under the 2008 DoD policy

- EVM compliance is required on cost or incentive contracts, subcontracts, intra-government work agreements, and other agreements valued at or greater than $20 million.
- An EVMS that has been formally validated by DCMA and accepted by the cognizant contracting officer is required on cost or incentive contracts, subcontracts, intra-government work agreements, and other agreements valued at or greater than $50 million.[5]

However, schedule and cost containment is clearly important in any acquisition setting. Metrics described in Section 5 will address this in more detail.

3.1.2 Delivery and Quality Monitoring

Traditionally, delivery is tied to major milestones on the timeline written into a plan. Work products are enumerated and specified in advance, and these deliverable items are evaluated at those milestones. The milestone often provides a gate that enables (or prevents) progress into the next phase—and, like the EVMS discussion above, payment is often tied to this progress. Therefore, metrics reported and analyzed during these "capstone events" play an important role in assessing quality and confirming acceptance of deliveries.

By contrast, Agile methods emphasize continual delivery and inspection. The full value of rapid feedback and reprioritization of requirements is enhanced by a continual approach, rather than a phased approach to delivery. Lapham 2011 describes two alternatives for resolving this:

> ***Progressive Technical Reviews****: is an iterative approach to implementing these capstone events. Only one event of record exists, but it is carried out in successive waves that build on*

---

[5] http://www.acq.osd.mil/evm/faqs.shtml

> *the previous reviews. In this fashion, the intent of the milestone review is realized, but in gradual fashion, rather than at a single point in time.*
>
> ***Multiple Mini Quality Reviews**: allow the conduct of quality reviews at a more detailed scope of the product (e.g., for feature teams developing part of a capability) and these more frequent, more detailed reviews are considered as input to the single capstone event held according to the preset schedule* [Lapham 2011].

These approaches strive to balance the needs of the acquisition process and the benefit of more frequent (and more detailed) interactions associated with Agile methods. Metrics described in Section 5 of this technical note play an important role in the process, regardless of the approach selected.

3.2 Tools and Automation in Wide Usage

Modern software tools available to support team collaboration during Agile development provide a wide range of customization and data capture options. These life cycle management tools model workflow and provide extensive communication features as well as configuration management functions. Some organizations manage detailed communications among developers using these tools—rather than relying on email to document important information.

Depending on the scale of the effort and the experience of the development teams with the tool suite, a variety of automated reports or custom queries may be available to support progress monitoring and analysis of critical events on a contract. Direct access to these tools for monitoring progress can greatly increase visibility for the acquisition professional willing to learn the interface and implementation choices made by the development organization. Whether requesting standard reports defined by the development organization, or performing ad hoc queries and analyses themselves, availability of the detailed “raw data” can be powerful. The graphs presented in Sections 4 and 5 of this technical note are easily generated from many of these tools.

One area of particular focus for automation, especially in larger software development programs, is testing. Automated testing tools support activities like unit testing and regression testing, to enhance the ability of the development team to retain confidence in the evolution of the capability without compromising the integrity of the code baseline. Agile methods include unit testing as part of development—rather than consider it a separate step in the process per se. As well, a rigorous focus on acceptance criteria—typically viewed as a component of the requirements—means developers can gain great leverage through use of effective tools for automating testing [Cohn 2004].

For defects discovered later in the process—during integration for example—the Agile team is able to better manage the potential impact on progress when automated test suites are built and maintained over time, rather than implemented “manually.” The ability to perform regression testing in a modern test environment enables a more thorough approach, while freeing the development team to focus on building a high quality product. This brings an emphasis on testing as a quality-enabling activity rather than a milestone to be achieved. Because of these and other implementation differences found in Agile methods, the metrics used to monitor progress may reveal a different emphasis when compared to traditional approaches. These are highlighted and discussed in Sections 4 and 5 of this technical note.

4 Agile Metrics

One of the experts interviewed for this technical note offered the following insight:

> *When the sponsor of a project gets a plan from the developer (whether they are an internal provider or an external contractor), they traditionally think of that as an exchange of risk. The plan is seen as a commitment to successfully build the product, and the developer—having planned the project—now assumes the risk. The sponsor mistakenly presumes that this commitment protects his/her interests. The reality is, if the project is late, or the product does not meet user expectations, the sponsor (not the developer) always owns the risk of market consequences* (Interviewee #6).

This explanation sets the stage for perhaps the most important emphasis on metrics for Agile software development—measuring delivered value. Merely tracking to a plan is not sufficient to manage risk. Thoughtless allegiance to the original plan, in fact, could assure failure if market conditions (mission parameters) or assumptions made while building the plan have changed. The iterative nature of Agile methods protects sponsors and developers from this trap.

Traditional implementations of earned value management systems attempt to protect against a cost and schedule only focus by ascribing value to the completion of tasks in the work breakdown structure. As these tasks are completed, the project sponsor gains a view of work accomplished. The amount of value attributed to each completed task is based on the estimated effort for that task.

To be fair, progress monitoring must always rely on proxies for value, as the true worth of the developed system is not realized until it is placed in operation and users gain the benefit of its operation. However, the choice of proxies has an effect on the amount of insight gained—as well as the potential for unintended consequences. In general, Agile methods place a great deal of emphasis on “delivering working software” over other outcomes. For this reason, the proxies used as metrics in progress monitoring tend to relate more to the outputs of the work than the process used to complete the work. Metrics driven by story points, direct customer feedback, and products or defects as they flow through the development process (such as velocity, or customer satisfaction) are more common than metrics reflecting parameters of the process (such as schedule performance index, or estimate to complete).

Because schedule is fixed based on the cadence established (and agreed upon with the customer), progress monitoring tends to focus more on the amount of product delivered and the amount remaining to be delivered. This is not to say other measures aren’t used. However, the traditional emphasis on projecting the end date and the likelihood of a cost-overrun tends to be de-emphasized in favor of understanding what the team is building and delivering to the customer when using Agile methods. In Agile, a question more common than “how late will we deliver?” is “which features have to be deferred if we run into unanticipated problems?”

In Section 4.1 we familiarize the reader with some basic, common metrics used in Agile projects. Then we will explore more advanced uses and interpretations of them in Section 4.2.

4.1 Basic Agile Metrics

Three key views of Agile team metrics are typical of most implementations of Agile methods: velocity, sprint burn-down, and release burn-up. We present each of these below in their most basic form for the audience not already familiar with these cornerstones of Agile measurement. It is worth noting that metrics used in Agile development contexts tend to focus primarily on the needs of development teams, rather than on the status of a project or program.

4.1.1 Velocity

Simply stated, velocity is the volume of work accomplished in a specified period of time, by a given team. Typically, this is measured as story points accomplished per sprint. This measure is sometimes called “yesterday’s weather” by Agile practitioners [Alleman 2003], as if to indicate its sensitivity to local conditions as well as seasonal trends. Indeed most experts explain that velocity is “team-unique” and thinking of this measure as a parameter in an estimating model is a mistake. The team must establish its own velocity for the work at hand [Sliger 2008]. In addition, there are metrics associated with *trends* and *stability* of velocity—over time and/or across varying conditions. Elaborations, extensions and cautions are discussed later, but first—the basics.

The height of each bar in Figure 4 is the velocity for that sprint. Looking at this information, expecting this team to deliver 50 story points on the next sprint would seem unrealistic, since the team’s previous highest velocity had been 35 story points in sprint 6. Conversely, a workload of just 15 story points would likely underutilize the talents of the team. For another team, or this same team working on a different product (or under different conditions), the expectations may be very different.

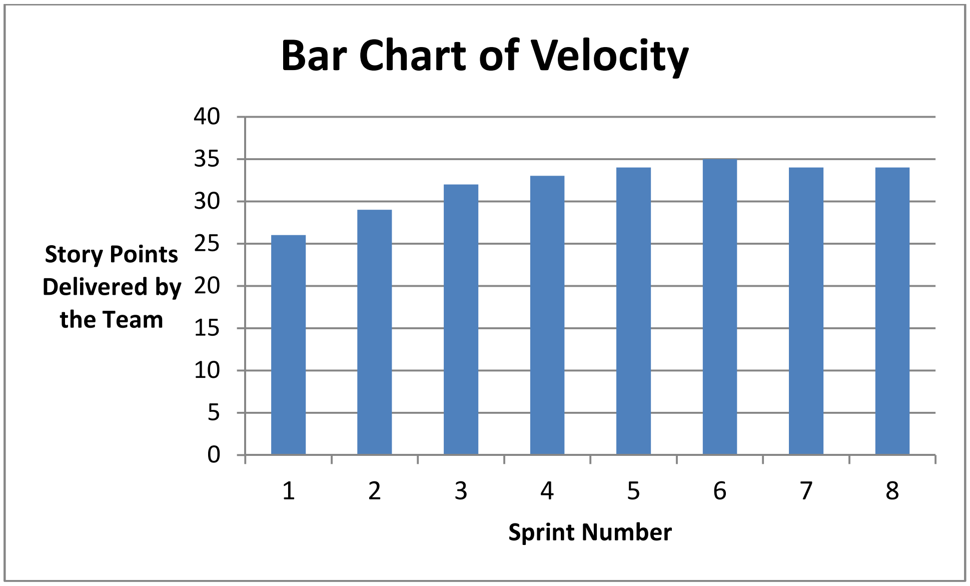


Figure 4: Sample Velocity Column Chart

Looking at the fluctuations in the height of the columns in Figure 4, we see a trend in gradual improvement across the first few sprints, then velocity fluctuates by one or two points for the rest of the sprints shown.

In this fictional case, the ‘back-story’ could be this is a new team, finding their way in a self-directed mode. During the first few sprints, some members were learning how to work with stories, and the process of testing against the “definition of done.” A bit of cross-training occurred—where those with more experience in testing coached novices in writing test cases. In addition, the use of the collaboration tools evolved as they experimented with preferences in the interface settings. The graph illustrates a concept called “velocity lag” which is discussed later in section 4.2.1.2.

The definition of velocity seems deceptively like a productivity measure, and many mistake it for that. However, the traditional uses of productivity measures are left wanting when velocity is considered—as the motivation to standardize the calculation (fixing the basis of estimates and standardizing across teams) runs counter to tenets of Agile methods. Velocity is a local measure, used by an individual development team to gauge the realism of commitments they make. The velocity metric is logically tied to the sprint burn-down chart, discussed next.

4.1.2 Sprint Burn-Down Chart

This graphical technique provides a means of displaying progress for the development team during a sprint. As items in the backlog of work are completed, the chart displays the rate and the amount of progress. This chart is typically provided for viewing on a team’s common wall, or electronic dashboard. Many elaborations and alternative implementations are seen in practice, but, again – first the basics.

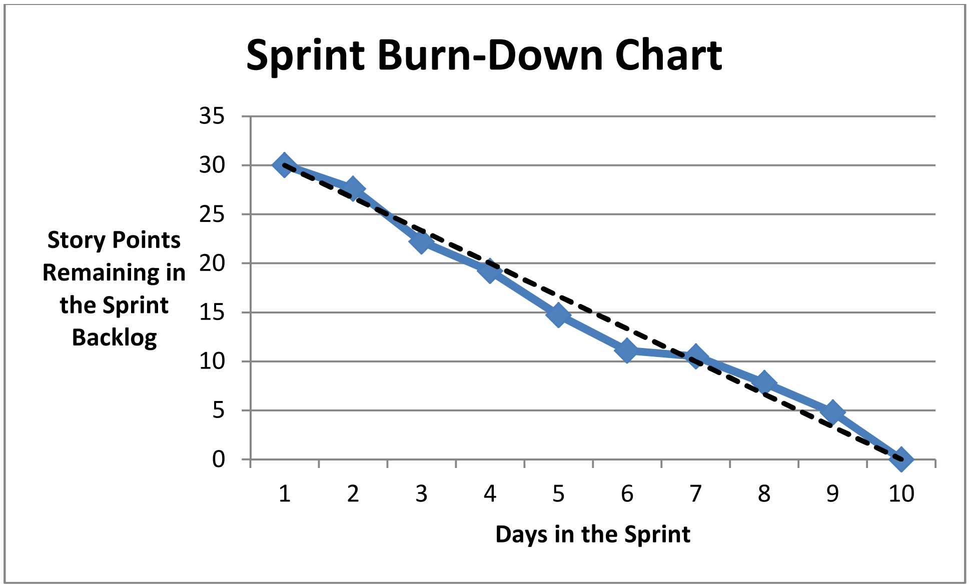


Figure 5: Sample Sprint Burn-Down Chart

Typically a line graph like the one in Figure 5, showing a decline in the remaining backlog items to be completed, is used. The workload chosen for that sprint (often called the sprint backlog) is reflected on the vertical axis—in story points for this example. A line sloping downward from left to right—with time depicted (in days) on the horizontal axis—shows the pace of work being completed. The dotted line shows the “ideal line” against which the thicker line can be compared. It is easy to relate this depiction to the completion of tasks remaining—or story points to be delivered—for the sprint. In this sample chart we see a noticeable change in rate for the accomplishments of day 6, because the story points remaining on day 7 do not appear to have gone down as much as in previous days—the "ideal line" also helps to highlight this. Such a data point may reveal that the team encountered an impediment to progress, which was addressed at that time. However, without more direct interaction with the team, that is just a guess. Like metrics in traditional environments, Agile metrics provide a starting point for a discussion, rather than providing a final answer by themselves.

4.1.3 Release Burn-Up Chart

A complementary graphical technique for the sprint burn-down, the release burn-up chart is also commonly used. Many cling to the convention that sprints burn down and releases burn up—though there is no mathematical principle that governs this choice.[6] With each completed sprint, the delivered functionality grows, and the release burn-up chart depicts this progress in an intuitively logical fashion. Here again, workflow management tools, and many extensions of the concept are in use—but first, the basics.

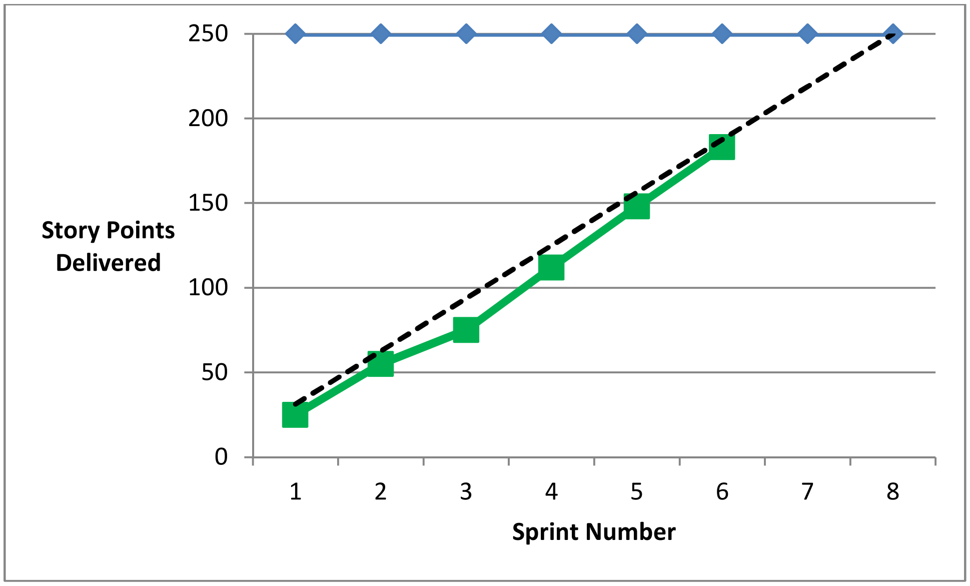


Figure 6: Sample Release Burn-Up Chart

This chart, too, is typically presented as a line graph showing accumulation of delivered value—in this case as story points—on the vertical axis. Time is depicted on the horizontal axis (by listing the sprint number), and a line sloping from the bottom left to the upper right of the graph depicts progress. As shown in Figure 6, a line at the top of the graph is typically used to represent the total number of story points planned for the release. This can also be used to depict changes in the planned content of the release (increases or decreases in scope). The viewer is led to think about the upward-sloping line *intersecting* with the top line, at or before the right edge of the graph (here again, the "ideal line" is shown as a dashed line in the graph). Many extensions of this representation are seen in practice—including the depiction of a “cone of uncertainty” drawn around the progress line, which reveals a widening level of uncertainty over time. [Cohn 2006].

---

[6] This is a paraphrase from one of our interviewees.

4.2 Advanced Agile Metrics

In the sections that follow, the basic metrics described in Section 4.1 are elaborated further, and extensions to the basic approach are described. The purpose of these discussions is to provide the reader with insights that aid in the interpretation of metrics presented by the development organization. We summarize tell-tale signals that can be gleaned from the metrics reports and provide some interpretive guidance. The examples in this section assume that story points are used for team estimation and tracking.

4.2.1 Velocity Metrics

Self-directed development teams commit to a sprint goal with associated user stories and success criteria. The story point estimates established by the team must be sufficient to provide a confident basis for committing to a series of sprint goals. The team sets a new goal at each sprint, in light of recent experience on previous sprints. The first few sprints may reflect learning by the team, as velocity may vary more while the team learns to work together.

Velocity (the number of story points delivered on a given sprint) is a metric primarily intended to guide the development team itself to understand the realism in delivery commitments. The feedback provided at the end of each sprint informs the planning for the next sprint. Adjustments in the level of commitment and a focused scrutiny of the workload promote deeper understanding of what is achievable. The point of monitoring velocity is not to promote on-going increases in productivity. Rather, the point is to assist the team in consistently operating with achievable commitments—and achieving those commitments. This mindset will aid the acquisition personnel in making beneficial use of this metric. The fundamental question to be answered by looking at velocity boils down to “is the team meeting its commitments?” The following discussions address typical uses of velocity and common patterns observed in practice.

4.2.1.1 Diagnosing Velocity Trends

During initial formation and startup, a new development team will observe fluctuations in velocity due to a variety of learning effects. Inexperience with relative size estimates, learning to account for new or different tasks, coping with defects discovered in later sprints, and planning for architecture or design work outside of a sprint are some contributors to common trends in velocity. While mundane, the team also needs to learn to include things like vacations and holidays. These can be planned for but are sometimes overlooked by inexperienced teams (inexperienced in planning that is). A skilled coach can help to navigate uncertain waters when the team is in the early learning stages. Measures involving story points and velocity serve as a basis for a number of different metrics used in Agile development. It is important to understand the maturation of the team, and the consequence of the learning process on those metrics.

Relative Size

The accepted approach to estimation in Agile methods, using story points (or ideal days[7]) as the basis, involves judging the size of the planned deliverables relative to each other. This is in contrast to other, absolute, estimation methods that rely on proxies like function points or lines of code—where an initial calibration is required for “correct” estimates to be produced. Cohn provides a clear and comprehensive discussion on the use of relative estimates in Agile development [Cohn 2006].

The value placed on limiting ceremony and documentation means the estimation process too is oriented to deliver needed value with the minimum of overhead. This means that teams early in the use of Agile methods may show more variation in velocity than a more seasoned team that has more experience in performing relative estimation within the context of the project at hand.

Unforeseen Tasks

A fully cross-functional Agile team enables all tasks—from requirements analysis, architecture, design, coding, to testing—to be performed within the development team, rather than relying on specialized external groups (e.g., separate groups of business analysis, coders, and testers). For some, this is a dramatic change from previous ways of doing work. This aspect of the learning curve for adopting Agile methods may contribute to false starts or real-time adjustments of tasks and roles after the team has begun to work.

Like other perturbations to the team’s pace of work, the adjustments to the work routine that come with adoption of Agile methods may be visible in the velocity of the team. The first two or three sprints may be unduly influenced by minor adjustments occurring in real time. The stable delivery performance of the team may not be seen until these transient causes of lower velocity are systematically eliminated. Here again, a skilled coach helps to navigate uncertain waters for the team early in its experience with Agile methods.

Defect Handling

Discovering and correcting defects is typically handled one of two ways in Agile development:

1. Testing and defect fixing entirely within sprints by the team, which does coding and testing
2. Testing conducted by a test team working in parallel with the development team(s)

Most who learn Agile methods are told that testing is part of the development done within a sprint. The team delivers “potentially shippable software” from each sprint, so testing is done before the sprint can be considered complete. We have seen teams who choose to write “test stories” or “integration stories” in order to account for this testing work. It is also common for acceptance criteria for a user story to serve as the requirements for test cases. Other teams we have seen budget a flat level of effort (e.g., 10 percent of the story points, and therefore 10 percent of the time) in each sprint to account for work that must be done to resolve defects from previous sprints. We caution the reader against accepting this as a rule of thumb—as each team needs to learn about its own velocity. Teams we have seen who manage defect discovery and resolution within a sprint do not tend to record defect counts or track time spent fixing defects discovered and resolved *within* a single sprint.

---

[7] *Ideal time* is the amount of time that something takes when stripped of all peripheral activities. When we estimate the number of ideal days that a user story will take to develop, test, and accept, it is not necessary to consider the impact of the overhead of the environment in which the team works [Cohn 2006].

For larger systems, developed over the course of years by a collection of Agile teams, a different strategy for handling testing and defects may be needed. In examples we have seen, unit testing remains the responsibility of developers within a sprint. Integration testing, however, was performed by a testing team working in parallel with the development teams. Successful organizations that take this approach have run the test team using Agile principles, and engage the testing team and the development teams collaboratively. Keeping the tested baseline and the developmental baseline from diverging too much can become a technical challenge in this case. New features built against outdated baselines may not be adequately tested. If the testing team falls behind, the development teams will be forced to either wait for the new baseline (stop work) or proceed at risk (developing against an old baseline). Neither choice is ideal, so the backlog being worked by the test team is monitored very closely. For some defects, it may be possible for members of the testing team to propose a fix that can then be more quickly implemented by the development team. However, for most major defects, one of the development teams will have personnel with familiarity and recent experience needed to address the defect—and fixing the defect will become an item in that team's backlog. If severe issues are encountered during testing, it is possible that the work performed to resolve defects displaces the work to continue developing the product. One particular team we saw struggling with this risk used weekly coordination meetings of all development teams and the test team to monitor progress. The status of the "next integrated baseline" was a standard agenda item in these meetings. Defects in the backlog as well as new stories in the backlog were sometimes deferred to a later sprint, if related functionality was not yet present in the baseline.

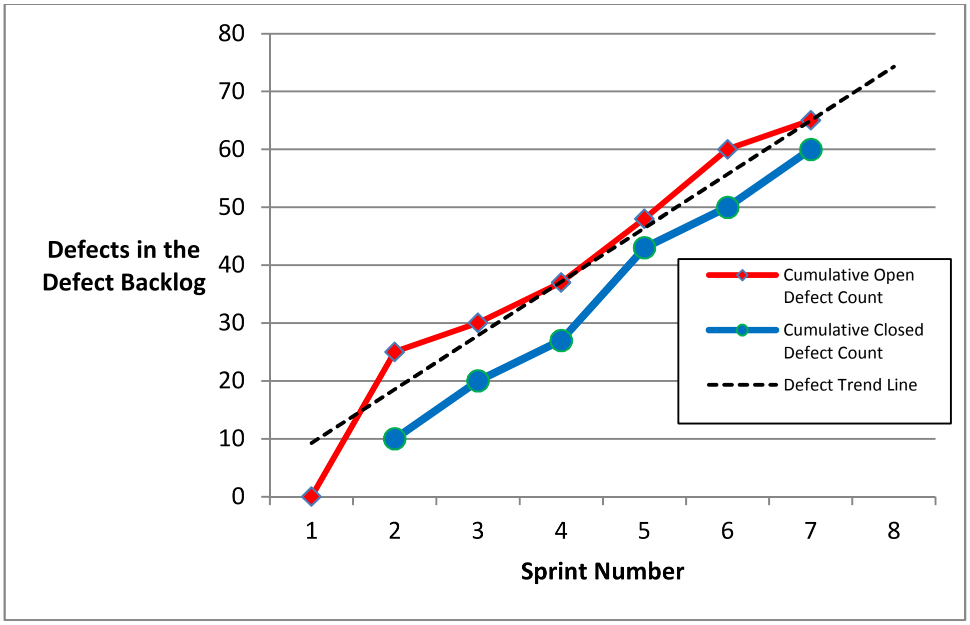


Figure 7: Sample Defect Burn-Up Chart

Monitoring metrics on the "defect backlog" is essential in this case—as early-warning signs enable corrective action before the team is overwhelmed with quality issues. Figure 7, above, shows a variant of the release burn-up chart that can be used to track the defect backlog. Note that the top line tracking the cumulative open defect count increases as the workload for defects "escaping into the next release" continues—while the bottom line tracking cumulative closed defect count reveals the progress in closing defects in each sprint. The dotted trend line supports a projection for the final sprint (yet to completed) in the figure. The discussion of cumulative flow diagrams in Section 4.2.2 describes another way to track work on the defect backlog. Many Agile coaches have ways of monitoring for testing-related dynamics, and triggering events and actions to prevent rework from overwhelming development.

Handling Spikes

The term spike is used to describe:

> *… a task included in an iteration plan that is being undertaken specifically to gain knowledge or answer a question* [Cohn 2006].

Others may use this term more loosely, to also include brief tasks focused on resolving a design issue, or elaborating the product architecture as needed. In any case, spikes are time-bounded activities performed to achieve a particular goal. A spike may be defined after a release plan is in place. They may displace some of the workload for an upcoming sprint. A skilled coach will have strategies for balancing competing needs.

In progress monitoring, the impact of adding spikes and the value they return to the development effort must be understood. The team must maintain the discipline of "time-boxing" when electing to use spikes to address issues. Examining metrics on spikes, such as planned and actual duration of each spike, as well as the number of spikes occurring in each reporting period will reveal how well the team is managing its use of spikes. Conducting too many spikes will block development progress—as revealed in the velocity metric, or the release burn-up chart. Doing no spikes may undermine the team's ability to embrace customer-driven change or improve the product based on feedback. This consequence may be seen in customer satisfaction metrics collected during sprint reviews or in release retrospectives.

4.2.1.2 Velocity Lag

Teams growing accustomed to working together, or those hampered by a common obstacle, may display a pattern in the velocity of sprints whereby a gradual rise in velocity is seen in the first few sprints (or those following a process improvement) that reflects the learning of individuals working with new conditions or people. This has been termed "velocity lag" by some, as the true performance of the team is realized after an initial lag in performance—while the team acclimates to the working conditions and rises to their potential as uncertainties are resolved. We've been told that teams typically take three sprints to get to "their velocity."

Organizations unfamiliar with Agile methods may flounder during this period, and attempt counter-productive corrective actions. One counter-productive pattern is seen when new staff are added to the team continually, in hopes of achieving a target velocity. Especially early in the effort, new team members create a potential ripple effect as they seek interaction with fellow team members to orient themselves in their new role. If the staffing on a team is fluid during the initial sprints in a release, then the natural pattern of velocity lag may be confounded by state of flux in the team. It is possible to prevent the team from realizing its potential velocity by continually changing staffing. Monitoring metrics for velocity and staffing levels—together with the burn down and burn up charts—will reveal clues that such an issue may need to be addressed. Such detailed staffing-related decisions are not typically the responsibility of the contract monitoring professional. However, the metrics serve to confirm (or call into question) the success of the development organization in managing issues encountered.

4.2.1.3 Coefficient of Variation

According to one of our interviewees, the coefficient of variation (CoV) provides insight about the stability of a team's velocity—an indication that they have found a steady state of performance. The coefficient of variation is computed as:

*Coefficient of Variation= (Standard Deviation / Average) * 100*

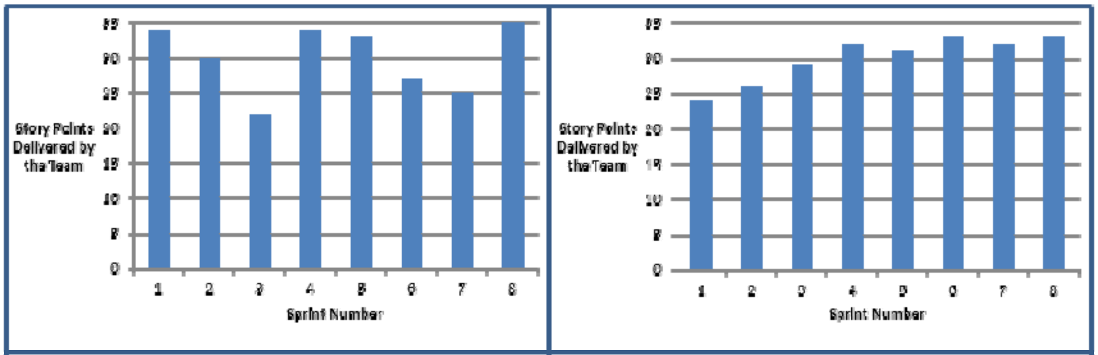
We heard about the use of this statistic as a method for analyzing variation in velocity while controlling for differences in average performance. The method requires positive numbers (which shouldn't be a problem for most analyses of velocity), and there are statistical tests available to compare teams or trends over time.

The CoV was used to analyze a large amount of performance data from Agile teams, as a means of differentiating teams that are struggling from the rest of the pack. For a team working with a good cadence, with good quality input (i.e., well-defined user stories) the velocity from sprint to sprint is expected to be stable. Computing the CoV for the three most recent sprints, and monitoring that statistic over time will provide insight about how well the team is performing with respect to their business rhythm.[8] Events that disturb this rhythm will be confirmed by their effect on the variability in this key team metric—velocity. Again, the responsibility to make focused management decisions to maintain this rhythm rests in the development organization, but monitoring this metric helps the contract monitoring professional build confidence in performance.

As we will see in the discussion of EVMS in Section 5.8, the average velocity across the sprints in a given release is a key element in computing AgileEVM metrics. The CoV can be used to understand how *reliable* the average velocity is—with higher values of the CoV indicating that the average velocity may be less reliable and low values of the CoV indicating that it is safer to use the average velocity as a measure that typifies team performance across the sprints.

---

[8] This is a specific recommendation from one of our interviewees.



Average = 30	Average = 30
Standard Deviation = 4.84	Standard Deviation = 3.38
Coefficient of Variation = 16.13	Coefficient of Variation = 11.27

Figure 8: Coefficient of Variation Example

The figure above illustrates a method for comparing the fluctuation of velocity for two different teams using the Coefficient of Variation (CoV). The performance depicted in the left panel shows a greater degree of fluctuation in velocity across the 8 sprints than seen in the panel on the right. Visual inspection of the column charts reveals that the velocity of the team on the left varies by 10 points or more from one sprint to the next. The velocity of the team on the right, by contrast, only fluctuates by 2 or 3 story points from sprint to sprint. As an acquisition professional monitoring the performance of these two teams, which team would you expect to provide a more reliable estimate for sprint 9? Both of these teams have an average velocity of 30, and they have both delivered 240 points thus far. However, the team depicted on the right appears to be more predictable. The smaller value of CoV for the team on the right helps to quantify this difference[9].

4.2.2 Flow Analysis

The cumulative flow diagram is a data visualization technique that comes from the Lean Product Development field [Reinertsen 2009]. This technique provides an effective means for monitoring performance from a variety of perspectives and is starting to see greater use among Agile developers. The technique brings a focus on work items that go through a set of state transitions and allows the user to count items in each state and evaluate the time items spend in each state, thereby identifying possible bottlenecks and risky logjams in the process.

To illustrate this technique, consider a very simple life cycle comprising four different states:

1. **Ready**: The customer stakeholder has written and/or reviewed the user story, the team has reviewed it to determine its size and to seek any needed clarifications or elaborations, and the story is ready to be selected for inclusion in an upcoming sprint.
2. **Coding**: Having been identified as one of the highest value stories in the backlog, the story has been selected for inclusion in a sprint, and the team is now working to implement it.

---

[9] In this example the value of the standard deviation would suffice as well, because the two teams have the same average velocity. The Coefficient of Variation provides a more robust basis for comparing fluctuation when the averages differ.

3. **Testing**: Having completed all development work, the code written to implement this story is currently being tested
4. **Done**: Having successfully passed all tests, this code is potentially ready to ship.

Given the above (simplistic) life cycle description, a backlog of user stories that makes up 30 story points can be depicted using the four color-coded zones on a chart. The graph in Figure 9 represents the allocation to the four different states at a given point in time. In this fictitious example, it can be seen that only one-third of the total story points remain in the "Ready" state.

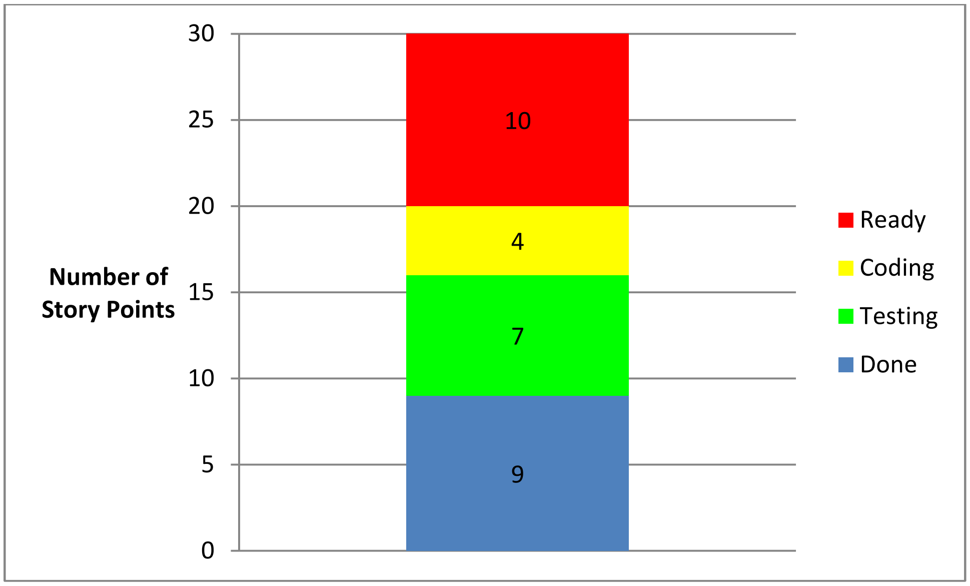


Figure 9: Sample Stacked Column Chart

In the next chart, the display is expanded to show all six weeks in the release cycle. For this fictitious example, the team is using two-week sprints, and status reports are compiled weekly. Therefore, this is a release with three sprints spanning a total of six weeks—to deliver 30 story points of working software.

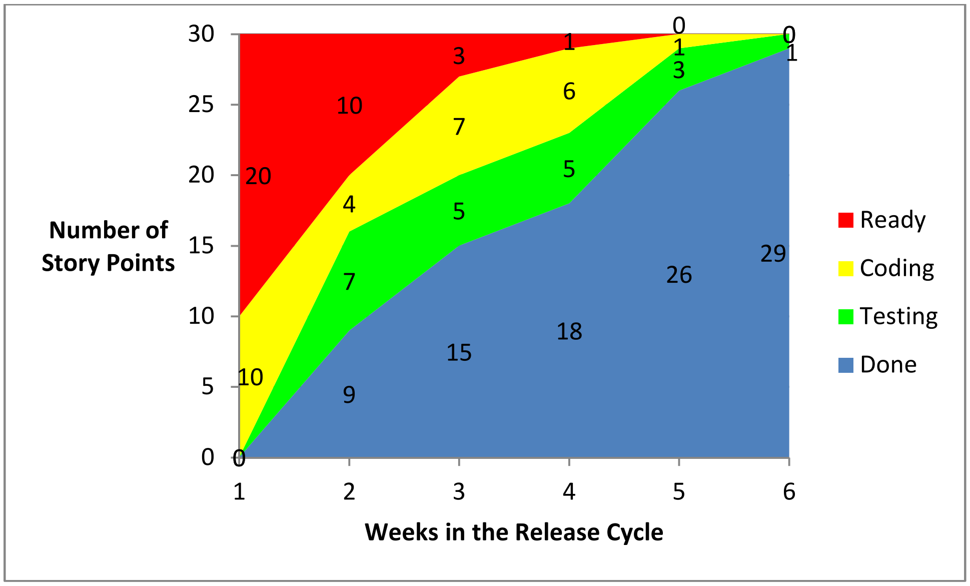


Figure 10: Sample Cumulative Flow Diagram

This display reveals a number of important performance metrics worth monitoring:

First, the number of story points in the coding and testing states—combined (often referred to as "work in process") appears to follow a fairly consistent pattern from week to week—each of the first four weeks show 10 to 12 points total, with the last two focused on a few remaining stories. One important principle in Agile methods is to maintain a small number of items in the work in process queue [Anderson 2010]. When the work in process grows substantially, it is a sign that the team may be stuck on some things, which remain unfinished and starting on others to keep moving. As the number of things set aside grows, so does the risk that the developers will never have time to get back to them. (A particularly risky example of this is when a backlog of deferred defects grows into a bow wave that swamps the team.)

Second, the steady increasing pattern in the number of story points that are "done" in this example shows a consistent rate of progress. However, the number of points done at the end of week four was only three more than the number done at the end of the previous week. The team would have examined their velocity—the number of story points delivered—to diagnose the pace of work and why it declined. Identified impediments might lead to process improvements intended to prevent that source of delay in the future.

4.2.2.1 Elements of the Cumulative Flow Diagram

With the above specific example in mind, the fundamental elements of this display technique are described here for more general understanding. A more complete elaboration of these (and related) principles can be found in Reinertsen.

The idealized picture in Figure 11 (below) illustrates a very steady pace of work entering the "In Process" state, and leaving the "In Process" state. The slope of the upper edge of the central (green) band depicts the rate at which work items enter the "In Process" state, while the slope of the lower edge of the central (green) band depicts the rate at which work items leave the "In Process" state. In this example, they appear as parallel lines because work items are completed at the same pace as new items are started.

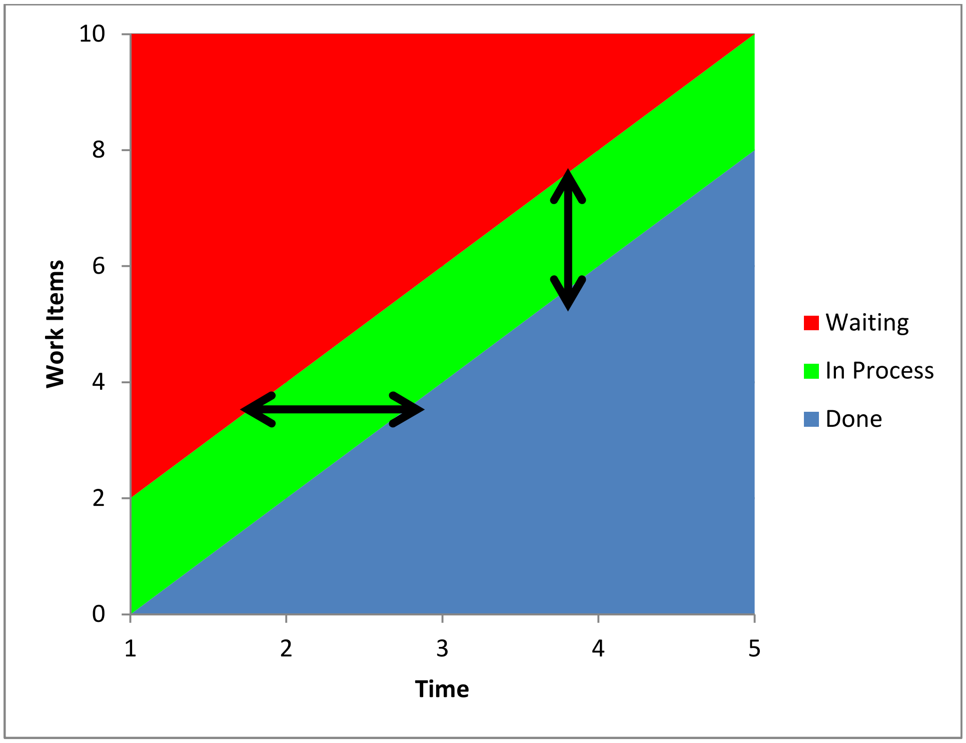


Figure 11: Sample Cumulative Flow Diagram – Fixed Pace of Work

Two more very useful pieces of information are available from this Cumulative Flow Diagram as illustrated in the arrows overlaid on the graph—the cycle time for items in process as well as the number of work items in process at any point in time. The horizontal double arrow depicts the average time required for an item to work its way through the in process step, and the vertical arrow depicts the number of items currently in process at the point where it is drawn. From this first simple graph, we can see that the cycle time is consistently one time unit (a distance on the horizontal axis) while the number of items in process is consistently two work items (a distance on the vertical axis). Such a steady pace of work and consistent workload is not typical.

The two graphs in Figure 12 illustrate graphical indicators that aid in the diagnosis of workflow issues.

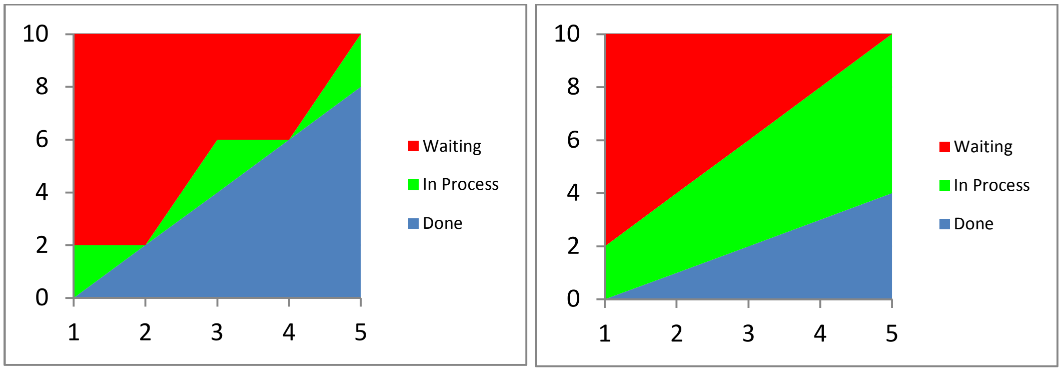


Figure 12: Sample Cumulative Flow Diagram – Illustration of Workflow Issues

In the graph on the left, the rate at which new work is taken into the in process state varies—and the work in process goes to zero at time 2 and time 4. This indicates that the personnel doing that work may be sitting idle while waiting for some prerequisite to complete before working on new items (e.g., waiting for customer sign off on a user story). In contrast, the graph on the right shows a growing backlog of work as the rate of items leaving the in process state is not keeping up with the rate at which work items enter that state. This may be indicative of a pattern in which new work items are accepted before current work items are completed (e.g., staff "get stuck" and can't proceed, so they take up new items so they don't sit idle).

4.2.2.2 Many Applications for Cumulative Flow Diagrams

As a tool for displaying and analyzing metrics, the cumulative flow diagram has numerous potential applications. For example, this display can be used to:

1. understand the flow of development work to implement user stories—serving as a complement to a series of sprint burn-down and release burn-up charts used by the team
2. track the progression of new user stories as they evolve during the life of the project—showing a parallel stream of work to the one mentioned above
3. monitor the execution and completion of test cases (including pass/fail and retesting) during major testing activities
4. assess the discovery and resolution of defects over the life of the project or within specific major test phases
5. monitor the handling of impediments reported by the Agile development team—which can relate to the team's ability to achieve its ideal cadence of work (as measured with velocity)

The influence of "Lean Thinking" deriving from manufacturing research has contributed greatly to analyzing flow in this way [Womack 1996]. A slide set provided by Yuval Yeret offers a tutorial on cumulative flow diagrams that extends the discussion provided here.[10]

---

[10] Please refer to http://www.slideshare.net/yyeret/explaining-cumulative-flow-diagrams-cfd. Another useful reference is available at http://brodzinski.com/2013/07/cumulative-flow-diagram.html

5 Progress Monitoring in Acquisitions Using Agile Methods

The subject of estimation for Agile development is discussed in Lapham 2011, including considerations for acquisition strategies and participation of government personnel as subject matter experts. Commonly used software tools for estimation and planning allow users to adjust parameters that account for differences in the methods used, as well as the capabilities and experience of development staff. Use of such tools (e.g., Price-S, Software Lifecycle-Management Estimate [SLIM], Constructive Cost Model [COCOMO], or Software Evaluation and Estimation or Resources [SEER]) will not be discussed here.

In the sections that follow, eight major categories of progress tracking metrics are discussed.

1. software size
2. effort and staffing
3. schedule
4. quality and customer satisfaction
5. cost and funding
6. requirements
7. delivery and progress
8. Agile earned value management system

5.1 Software Size

As a progress monitoring metric, the size of the product is often of interest in traditional development approaches because it is used as a leading indicator for the eventual cost and schedule. In Agile approaches, the preference is to fix cost and schedule, using a time-boxed approach [Anderson 2010]. When you establish a steady cadence of work—with predictable cost—your goal shifts to maximizing the amount of customer-valued functionality that is delivered. From this viewpoint, size becomes a variable attribute rather than a fixed starting point. Using the customer's prioritization, the team strives to load-balance the release by selecting the highest priority stories that will fit into their established capacity (the size of the funnel depicted in Figure 13). The association of stories with system capabilities is a useful way to provide context for interpreting accomplishments—when considering a broader perspective rather than the focus of the development team, as discussed here.

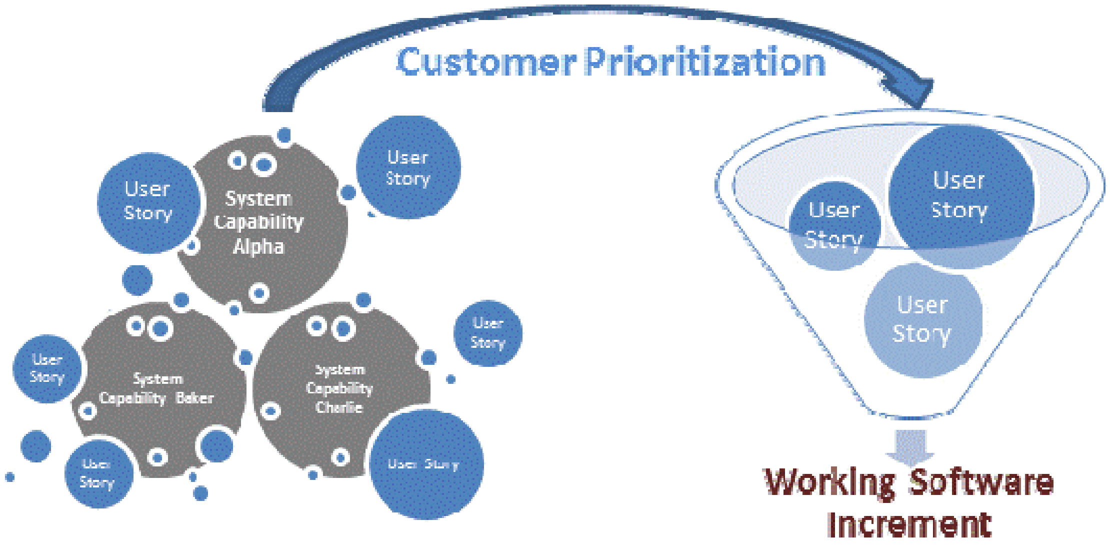


Figure 13: Estimating Workload for a Single Sprint

The relative size of each user story (all of the circles shown above) then becomes a source of information used by the development team to enable the customer to consider alternative sequences and groupings. Alternatives like:

- implement the most visible features in each capability, so users can see them as prototypes to be elaborated or refined
- choose the most important capability and build it first so that a partial system can be fielded to meet pressing needs, and the remainder can be added incrementally
- implement capabilities that represent the infrastructure needed to field other capabilities

In this process of considering alternatives, the size of the story is an enabler for judging feasibility of short-term options—not a basis for a larger forecast.

In tracking the progress of the team, the focus is on their ability to meet the commitment to deliver the working increment of software. Therefore, the correspondence between the number of story points planned and the number delivered becomes the *focus*, and not just a measure of what is delivered. Size is used here to help the team understand its capacity. Discussions of velocity presented earlier apply here. The size metric serves as a basis for diagnosing the performance of the team. When multiple development teams work to build a large system, it is important to understand that the story point is typically calibrated to *individual* teams—rather than being a proxy calibrated across teams. This may present a challenge to government acquisition staff who are unaccustomed to working with Agile development teams.

If a team consistently falls short of their targets for delivered story points, it is an indication that the scope of work may not be sufficiently understood, or that the capability of the team is not viewed realistically. A comparison of planned and actual velocity for each sprint, as well as the coefficient of variation (described earlier) for velocity, represent size-related metrics that contribute to diagnosing team performance. This type of diagnosis is typically performed by the team, working with its coach or leader. Acquisition professionals monitoring performance of the development organization view these metrics to identify the need for corrective actions, or to confirm the effectiveness of actions taken.

Use of story points—or any other measure of size—to understand the pace of work accomplished is fraught with risk. As discussed previously, a number of things can influence the count of story points. Individual differences in expertise and experience among team members will greatly influence the actual time spent to implement a given user story. When monitoring progress, it is important to distinguish between predictable performance and the rate of delivery—Agile methods emphasize *predictability* first.

5.2 Effort and Staffing

Staff effort is primarily of interest to acquisition personnel because labor is typically a predominant driver of cost for software development contracts. However, it is rare that a detailed analysis of effort hours expended will be the focus of progress tracking metrics—unless it is used to diagnose performance shortfalls, or the contractor is in danger of exceeding the budget. Typically, a so-called burn rate will be monitored as a matter of routine. Departures from expected rates or dramatic changes in rate will raise questions.

With a fully integrated team, Agile methods achieve greater velocity in part because the product does not transition in its intermediate state from one specialty group to another (e.g., requirements analysts handoff to designers, who handoff to coders, who then hand off to testers). In traditional models, each handoff increases the opportunity to introduce quality- and schedule-threatening errors. Furthermore, when the Agile team embodies all of the perspectives that other models treat as different roles in the process, the real-time interplay and synergy among these perspectives can be more effective.

The implication for staffing profiles, particularly as it relates to integration and testing activities, is described in Lapham 2010. The graphic in Figure 14 is copied from that report for emphasis here.

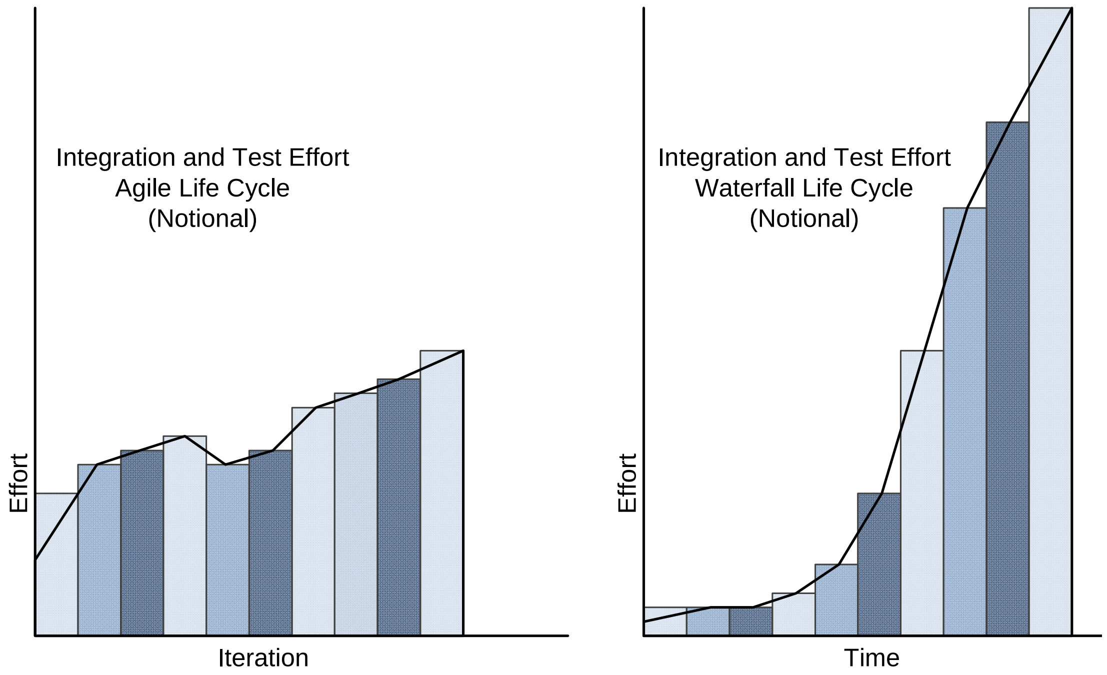


Figure 14: Notional Illustration of Agile Versus Waterfall Integration/Test Efforts

The continuity of staff implied by this discussion becomes an important ingredient to success. The sensitivity to Brooks' law[11] may be more apparent when looking at the performance of Agile teams [Brooks 1995]. Considering our discussion of velocity lag in Section 4.2.1.2, consider the proposition that a development team can be prevented from realizing its full capability by excessive changes in staffing. Because the workload is more effectively shared among team members, perturbations to the team may also impact their performance more uniformly.

The traditional approach to accounting for effort typically involves tracking a variety of effort categories (e.g. design, code, test)—essentially presuming a waterfall-like process. Organizations deploying fully integrated cross-functional teams will most likely find this expectation to be at odds with the way they operate. The burden to satisfy government stakeholders may fall on contract monitoring professionals who translate the contractor's data into reports addressing the traditional cost categories, unless a contractual agreement drives the detailed reporting format. In any case, it is important to understand this potential for miscommunication.

5.3 Schedule

As described earlier, the intent of Agile development methods is to treat cost and schedule as fixed parameters, and to manage progress in a way that maximizes the amount of high priority functionality delivered. This is in contrast to traditional approaches, where schedule and cost are a consequence of the choices made in the content and quality of what is delivered. Therefore, traditional metrics focusing on things like schedule performance, or projecting the end of a given activity are less meaningful in Agile development approaches.

---

[11] Brooks Law essentially states that adding staff to a late project makes it later.

Through tracking velocity over sprints and releases, program office staff can have a basis for understanding the pace of delivery, based (ideally) on a stable cadence of work activities. This can be used to examine the backlog of work remaining, and establish confidence (or identify risk) in meeting a set schedule. Taken a step further, an implementation of AgileEVM can also provide the same types of metrics available in traditional approaches [Sulaiman 2006]. We discuss AgileEVM in more detail in Section 5.8.

5.4 Quality and Customer Satisfaction

Agile methods place a great deal of emphasis on quality early in the project [Sliger 2008]. Acceptance criteria (sometimes called "Definition of Done") play an important role in documenting complete user stories [Cohn 2004]. The customer typically plays a prominent role in prioritizing user stories as well as providing timely feedback at the completion of each sprint. Therefore, the view of quality is not relegated to counting defects or shortfalls in delivered functionality. Metrics that quantify business value or "warfighter value" (as one interviewee put it) for each user story can be collected—based on a customer-assigned number for each story—though this is commonly skipped in favor of the customer participating as the arbiter of priorities for the content of each iteration and/or each release. Therefore, each box in the graphic (Figure 14) below has a connection to a potential quality metric. Metrics collected during events like sprint demos and release retrospectives may be unique to the use of Agile methods.

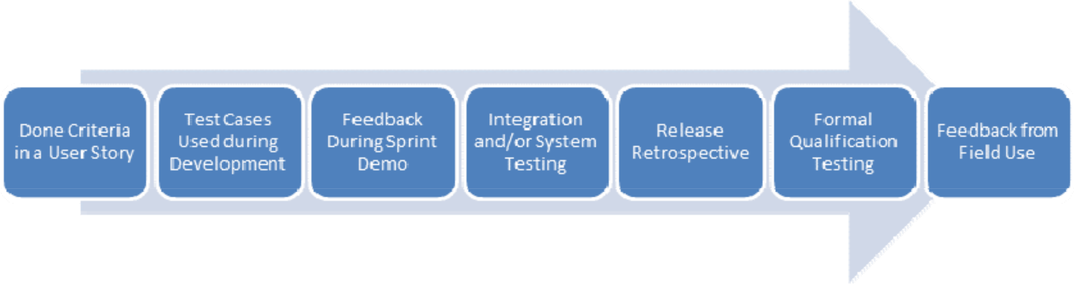
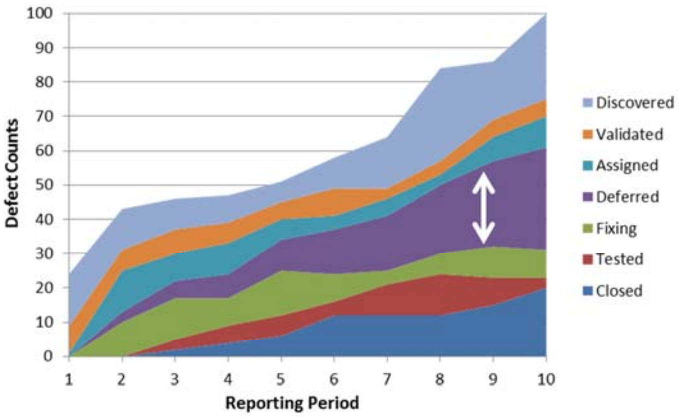

Done Criteria in a User Story
Test Cases Used during Development
Feedback During Sprint Demo
Integration and/or System Testing
Release Retrospective
Formal Qualification Testing
Feedback from Field Use

Figure 15: Many Quality Touch-Points in Agile Development

The choice to use Agile methods does not negate the traditional role of activities like System Testing, Formal Qualification Testing, or Customer Acceptance Testing. Defects discovered and fixed during these testing activities provide a basis for metrics in much the same way they do in more traditional development approaches. In addition, defects discovered in field use of the system warrant the same attention for metrics as we see in traditional approaches. With Agile methods, these traditional views of quality (i.e., measured defects) can be supplemented with a more direct measure of customer-perceived value—using customer satisfaction feedback or ratings collected in the various events depicted in Figure 15. Feedback loops specifically designed into the process serve to focus the development team on quality as seen by the customer.

In the earlier discussion of diagnosing velocity trends, three alternative approaches to defect handling are described. These shape the type of metrics available during development. In addition, the cumulative flow diagram is a powerful tool for analyzing defect metrics. Examining the flow of work to discover and correct defects, metrics that reveal cycle time and the backlog of defects in various states of work form a very powerful early warning indicator for the program.

Figure 16 illustrates a fictional analysis for workload relating to defects. The seven states listed on the right are depicted in colored bands in the cumulative flow diagram on the left.



Discovered: The defect has been logged but no other action has been taken

Validated: The logged defect has been accepted as an issue that warrants action

Assigned: The person who will work on the defect has been identified

Deferred: Work on the defect was deferred to a later (unspecified) date

Fixing: The person working on the defect has reported they have begun to fix it

Tested: The solution to the defect has been tested – and the tests have passed

Closed: The defect is fixed, and the fix confirmed

Figure 16: Defect Analysis Using a Cumulative Flow Diagram

The cumulative flow diagram illustrates the steady growth of deferred defects shown in the purple band at the center of the diagram (and highlighted by the double-headed arrow). This could be an indication of an unhealthy tradeoff resulting in an accumulation of deferred work across the 10 reporting periods. Alternatively this pattern could reflect a purposeful strategy for defining a later release with new feature requests submitted using the defect reporting system. In this case, the deferrals are needed to initiate the process of writing and reviewing new user stories—a flow of work we might want to monitor with a separate diagram.

The consistent and narrow bands for “validated” and “assigned” seem to indicate that work flows through these states more quickly and consistently (than other states). Finally, the number of defects discovered and waiting for further processing appears to be growing as well. Depending on the number of reporting periods remaining before the next (or final) release, this may represent a forecast for future trouble as well. Such patterns may lead to technical debt which threatens the successful realization of the product vision. A full treatment of this subject is beyond the scope of this technical note, but the subject is addressed well in Philippe Kruchten, Robert L. Nord, Ipek Ozkaya: Technical Debt: From Metaphor to Theory and Practice. *IEEE Software 29(6)*: 18-21 (2012). For a reference that looks into balancing development tempos from the perspective of architecture-related rework, consider: Robert L. Nord, Ipek Ozkaya, Philippe Kruchten, Marco Gonzalez-Rojas: In Search of a Metric for Managing Architectural Technical Debt. WICSA/ECSA 2012: 91-100. Finally, for a general reference on using architecture and Agile together, which includes treatment of technical debt: Stephany Bellomo, Robert L. Nord, Ipek Ozkaya: A study of enabling factors for rapid fielding: combined practices to balance speed and stability. ICSE 2013: 982-991

5.5 Cost and Funding

The use of Agile methods does not necessitate a change in the collection mechanisms associated with the cost of work performed—though such metrics are certainly not a focus of most Agile methods. However, there are some potentially advantageous structures to employ when setting up funding streams or cost accounts in an acquisition. Acquisition strategies that match the iterative nature of development with tiered funding—including optional extensions or add-on contract funding lines—can be used to great advantage when the developer employs Agile methods.

One of our interviewees explained that using an IDIQ contract was ideal in her situation, as this accommodated the needed flexibility to re-prioritize user stories between iterations. Each delivery of working software could then be planned and managed distinctly, while maintaining the continuity with a known provider and staff who have the experience of the previous iterations. The number of iterations could be increased or decreased without the costs that accompany revisions to other contract types. Using more than one contractor—to staff various teams in such a flexible manner—the pattern resembles acquisition of a service rather than acquisition of a whole system. However, as Lapham 2011 points out, it is important for the acquisition office to maintain cognizance across iterations. The risk of narrowly focusing on only the iteration at hand may cause local optimization for individual increments without due attention to the system evolution across increments.

5.6 Requirements

Traditional approaches to managing software development projects place a premium on stability of requirements—as an element of risk management. Various techniques for limiting "requirements volatility" have been lauded as best practices. In these contexts, the requirements (not unlike the size estimates) form a surrogate for many things that follow. With that mental model, it is easy to agree that keeping a requirements baseline "frozen" is helpful to the development team. However, the hard lessons learned in these contexts led one of our interviewees to observe:

> *If you've been on major contracts before, you know that the requirements are a fight from start to finish (bounded by cost, schedule, and the decision makers), and the delivered system is a compromise from the original vision.*

This experience reveals the challenge faced when the life cycle management model clashes with an evolving understanding of the product. Agile methods specifically address this natural tension through the way customer needs are understood and used to manage the work. At the level of the software development team (not the ACAT 1 Program level, for example) baselines of user stories are typically frozen during each short iteration, but changes to requirements and their relative importance are sought at regular intervals during development. Traditional metrics for requirements volatility do not fit the Agile development model. In fact, metrics that demonstrate orderly and timely acceptance of change—when correlated with customer satisfaction metrics—form a beneficial feedback loop.

Another consideration for examining requirements metrics derives from the variations in the way organizations adopt Agile methods. In some environments, requirements in the backlog are essentially feature lists—and do not contain all the necessary detail used by developers to drive implementation decisions. That information is acquired via direct conversation with users or their surrogates. In other environments, user stories as described by Agile authors are commonplace—with details including "done criteria" and test cases used to verify the implemented code. Metrics that reveal the process of maturing the requirements, or "grooming the stories" as it is often called, provide visibility into the state of the requirements that drive implementation. A cumulative flow diagram may be used to examine changes in the status of requirements, (e.g., how many of the user stories in the backlog are awaiting done criteria to be specified?) As well, the cycle time between a customer identifying a new story as a top priority, and the inclusion of that story in a subsequent sprint can be tracked—to gauge responsiveness of the team.[12]

5.7 Delivery and Progress

Agile methods place a greater emphasis on products delivered, rather than attributes of the process used to create those products. Measuring durations, costs, and schedule performance—while not unimportant—tend to receive less attention in Agile approaches. As we explained earlier, counts of delivered story points (embodied in working software) are the most prominent building block in Agile metrics.

One popular approach to Agile development is test-driven development [Beck 2003]. In this approach, rather than developing tests to verify an already built code base, test cases are developed first (and often automated) and the code is written to pass these tests. In this scenario, technical progress can be understood through metrics that show the pass/fail outcomes of the tests.

One of the experts interviewed for this paper described a novel implementation of this approach in a context where the unprecedented engineering challenges being tackled in the program formed the basis for tracking progress. The engineers were pushing beyond historical performance envelopes in the domain, and carefully crafted tests were used to demonstrate these ambitious performance levels. With each passed test, the stakeholders of the program could observe a reduction in risk to the vision of the program. This ruthless focus on performance by the engineers garnered much more interest than the quality of documents or status against future milestones. Our interviewee termed this a “risk burn-down” approach.

In all of our interviews of acquisition personnel involved in Agile programs, we heard a consistent theme of greater involvement in technical decision making. As described in Lapham 2011, acquisition personnel serve as subject matter experts in the programs. These professionals play an important role in prioritizing work, and providing feedback on the delivered products. This role differs from what we see in capstone events like preliminary design reviews (PDR) and critical design reviews (CDR). As described in Section 3.1.2, a leaner (more informal) approach is employed—one which yields deeper insight about the product.

5.8 Agile Earned Value Management System

The use of earned value management systems (EVMS) in monitoring major DoD development efforts is well established and required by regulations. Many acquisition professionals have time-tested approaches for diagnosing the health of the programs they oversee that rely on EVMS. Sulaiman, Barton, and Blackburn explain how EVMS can be employed using Agile metrics—chiefly velocity—in the context of monitoring a release in Agile development [Sulaiman 2006]. The authors present equations and use two sample projects to illustrate the equivalence of their approach to AgileEVM and what they call the burn-down approach.

As discussed in Section 4, an Agile development team’s use of velocity is a cornerstone for planning and monitoring work. The local nature of this metric warrants caution in its application.

---

[12] A soon-to-be-published technical note in this series will focus on requirements in the Agile and traditional waterfall worlds.

Generalizing across different teams or different project contexts is ill-advised. Sulaiman et.al. are cautious in their derivation of AgileEVM by scoping its application to an individual team, and an individual release. This is in contrast to most large-scale implementations of traditional EVMS for major programs comprised of multiple Integrated Product Teams (IPTs), often working in a variety of engineering disciplines. AgileEVM, like the velocity metric on which it is based, is a local metric system. The use of well-established concepts like schedule performance index (to name only one of many) and the ability to project a release date in a manner similar to traditional EVMS is advantageous. In settings where a single release is the focus, and the story point estimation is not done by separate IPTs operating independently, AgileEVM can be used.

Critics of EVMS take issue with how it is implemented in some settings, and these criticisms are instructive in this discussion. First, reliance on an initial requirements baseline to plan a large effort spanning a long timeline presents a challenge. Requirements and our understanding of them evolve during the life of the project. AgileEVM addresses this concern well by considering the timeline of a single release (comprised of multiple sprints). Story point estimates, and the relative priority of the stories are not likely to change much during the time span of a single release.

Another, perhaps more common criticism of EVMS, centers on the method by which “percent complete” is calculated at the work item level. The “inch-pebble” assessments of progress on each of the work items are aggregated to form metrics reflecting status against milestones. Without an objective basis for counting this progress, projections at higher levels are called into question. Here again, AgileEVM relies on velocity of a single team—rather than aggregations across different teams. This is important because story points (which are the basis for computing velocity) are the result of relative estimation performed by a single team [Lapham 2011]. However, the reliance on story points may not be adequate to represent “value” as emphasized in the Agile Manifesto.

Rawsthorne builds on the work of Sulaiman et. al. in suggesting an added perspective on measuring Earned Business Value (EBV). In explaining why this is important, Rawsthorne states:

> *This is the primary difference between traditional projects and agile ones. In traditional projects, we are expecting to deliver everything in our requirements document, so all we need to measure is efficiency and effectiveness of delivery. In other words, EVM metrics are enough for traditional projects. However, in agile projects, we don’t know exactly what we’re going to deliver. We’re constantly evaluating to see if we have enough, if we’ve generated sufficient ROI, etc., so we need some way of determining the Business Value of what we’ve got so far in order to make that determination* [Rawsthorne 2012].

The expectation in Agile development is that the development team will reach a point of diminishing returns before all of the stories have been implemented. As well, the team will need to spend time on architecture and other important work that may not be visible as important outcomes from the user’s perspective. Rawsthorne models this with an S-curve, as illustrated below.

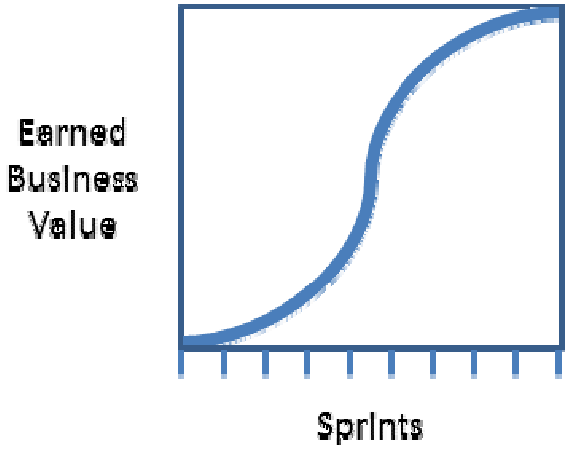


Figure 17: Accumulation of Earned Business Value [Rawsthorne 2012]

The figure above depicts a slower accumulation of EBV during early sprints, as the team works on architecturally significant stories that provide a foundation. At the center of the graph above, a much steeper rate of accumulation is apparent, as the team works to implement the highest value stories. Then, as the number of high business value stories in the backlog decreases over time, a point of diminishing returns is reached. It is at this point (at perhaps the eighth or ninth hash mark on the horizontal axis), before all of the originally projected funding and schedule have been consumed, that the project can be successfully closed and the product delivered. Traditional EVM metrics might show a projected cost- or schedule-overrun at this point. However, if the project can be closed before the originally projected end date—because the business value accumulated is sufficient to meet the operational need for the system—then the actual schedule and cost performance may not exceed the original budget.

6 Conclusion

It is fitting that the final detailed section of this report (above) ends with a call to add a perspective on “Business Value” to the traditional earned value management system used in many acquisition settings. If you are an acquisition professional working to monitor the progress of an Agile contractor, your top priority is to maximize the business value achieved from the contract. You must bring an understanding of the mission of those who will be served by the software-reliant system.

The foundation for Agile measurement, described in Section 2, derives from the set of values and principles tied to the Agile Manifesto. What many readers miss, when they first encounter Agile, is that the values espoused are not intended to condemn or dismiss the traditional hallmarks of software development. That is, things like *processes and tools*, *comprehensive documentation*, *contract negotiation*, and *following a plan*, are not inherently bad. The point of the Agile mindset is to maximize focus on *individuals and interactions*, *working software*, *customer collaboration*, and *responding to change*. Methods and processes that enable these things have been demonstrated to result in greater value to the customer. It is not feasible to operate without a process, provide no documentation, operate without a contract, or forego planning.

In DoD acquisition settings, in particular, there are important enablers to implementing Agile methods that warrant attention. Advice offered in Section 3 includes:

- If the PMO is doing a request for proposal, no matter which phase, ensure that the RFP contains language that allows the use of Agile.
- Be prepared to mine and effectively use the metrics data that naturally occur in typical Agile teams.
- Take advantage of the transparency provided in Agile processes. Don’t wait for the metrics to come to you. Go look at what the Agile team is doing.
- Keep abreast of the changing policy environment that affects metrics collection and use.

There are productive ways to meet typical regulatory requirements levied in the government contracting arena. Agile development organizations are expected to be responsive to customer’s needs—this is reflected in the values and principles that accompany Agile methods. Advantageous use of tools and automation often provide lean approaches to accomplishing what is needed.

Basic metrics commonly implemented on teams using Agile methods provide a foundation for a variety of uses. As illustrated in Section 4, there is a well-established knowledge base surrounding these common metrics. New rules of thumb are needed to use them well, but *velocity*, *sprint burndown charts*, *release burn-up charts*, and *cumulative flow diagrams* provide intuitively appealing displays of useful progress monitoring metrics.

Finally, putting the “Agile lens” on the traditional categories of progress monitoring metrics (like size, cost and schedule—among others) requires consideration of the value system underlying Agile development methods. The development organization’s focus on delivering business value, and the need for active customer participation, can bring clarity that is sometimes hard to find when using traditional phase-gate-oriented metrics.

Appendix A Past Publications in the SEI Agile Series

Below is a list of previous SEI publications in this series.

Considerations for Using Agile in DoD Acquisition
http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm

Agile Methods: Selected DoD Management and Acquisition Concerns
http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm

A Closer Look at 804: A Summary of Considerations for DoD Program Managers
http://www.sei.cmu.edu/library/abstracts/reports/11sr015.cfm

DoD Information Assurance and Agile: Challenges and Recommendations Gathered Through Interviews with Agile Program Managers and DoD Accreditation Reviewers
http://www.sei.cmu.edu/library/abstracts/reports/12tn024.cfm

Parallel Worlds: Agile and Waterfall Differences and Similarities
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=62901

References/Bibliography

URLs are valid as of the publication date of this document.

[Agile Alliance 2001]
Agile Alliance. *History: The Agile Manifesto*.2001. http://agilemanifesto.org/history.html

[Alleman 2003]
Alleman, G. B., Henderson, M., & Seggelke, R. "Making Agile Development Work in a Government Contracting Environment: Measuring Velocity with Earned Value," in *Agile Development Conference*, June 25-28, 2003.
http://www.informatik.uni-trier.de/~ley/db/conf/agiledc/agiledc2003.html

[Anderson 2010]
Anderson, David J. & Reinertsen, Donald G. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[Brooks 1995]
Brooks, Frederick P. Jr. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition* (2nd Edition). 1995.

[Coelho 2012]
Coelho, Evita & Basu, Anirban. "Effort Estimation in Agile Software Development using Story Points." *International Journal of Applied Information Systems (IJAIS) 3* (7): August 2012.

[Cohn 2004]
Cohn, Mike. *User Stories Applied: For Agile Software Development*. Pearson Education, 2004.

[Cohn 2006]
Cohn, Mike. *Agile Estimating and Planning*. Pearson Education, 2006.

[DoD 2007]
Department of Defense (DoD). *Department of Defense Directive (DODD) 5000.01*. November 2007. https://acc.dau.mil/CommunityBrowser.aspx?id=37343

[EIA-748]
ANSI EIA 748 *Intent Guide: A Standard for Earned Value Management.* 2005.
http://www.srs.gov/general/EFCOG/02GovtReferences/03NDIAANSI/NDIAIntentGuide.pdf

[Fleming 2000]
Fleming, C.D. & Koppelman, Joel M. *Earned Value Project Management – 2nd Edition*. Project Management Institute, 2000.

[Hartman 2006]
Hartman, Deborah & Dymond, Robin. “Appropriate Agile Measurement: Using Metrics and Diagnostics to Deliver Business Value.” *Proceedings of the Agile Conference*. Minneapolis, MN, July 2006. IEEE Computer Society Press, 2006.

[Lapham 2010]
Lapham, Mary Ann; & Williams, Ray; & Hammons, Charles (Bud); & Burton, Daniel; & Schenker, Alfred. *Considerations for Using Agile in DoD Acquisition* (CMU/SEI-2010-TN-002). Software Engineering Institute, Carnegie Mellon University, 2010.
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9273

[Lapham 2011]
Lapham, Mary Ann; & Garcia-Miller, Suzanne; & Adams, Lorraine; & Brown, Nanette; & Hackemack, Bart; & Hammons, Charles (Bud); & Levine, Linda; & Schenker, Alfred. *Agile Methods: Selected DoD Management and Acquisition Concerns* (CMU/SEI-2011-TN-002). Software Engineering Institute, Carnegie Mellon University, 2011.
http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=9769

[Rawsthorne 2012]
Rawsthorne, Dan. *Monitoring Scrum Projects with AgileEVM and Earned Business Value Metrics (EBV)*. 2012.
http://www.collab.net/resources/whitepaper_MonitoringScrumProjectsAgileEVM

[Reinertsen 2009]
Reinertsen, Donald. “The Principles of Product Development Flow: Second Generation Lean Product Development.” Celeritas Publishing, pp. 71–72. 2009. ISBN 978-1-935401-00-1

[Sliger 2008]
Sliger, Michele & Broderick, Stacia. *The Software Project Manager's Bridge to Agility*. Pearson Education, 2008.

[Sulaiman 2006]
Sulaiman, Tamara, Barton, Brent, & Blackburn, Thomas. “AgileEVM – Earned Value Management in Scrum Projects.” Presented at *Agile2006*, 23-28 July 2006.

[USAF 2008]
United States Air Force. *United States Air Force Weapon Systems Software Management Guidebook, Version 1 (Abridged)*. 2008

[USA 2011]
United States Army. *Army Regulation (AR) 70-1 Army Acquisition Policy*, Sections 7-12 and 7-13. 2011.

[Womack 1996]
Womack, James P. & Jones, Daniel T. *Lean Thinking*. Simon and Schuster, 1996.

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 the 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 Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY
(Leave Blank)

2. REPORT DATE
January 2014

3. REPORT TYPE AND DATES COVERED
Final

4. TITLE AND SUBTITLE
Agile Metrics:
Progress Monitoring of Agile Contractors

5. FUNDING NUMBERS
FA8721-05-C-0003

6. AUTHOR(S)
Will Hayes, Suzanne Miller, Mary Ann Lapham, Eileen Wrubel, Timothy Chick

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

8. PERFORMING ORGANIZATION REPORT NUMBER
CMU/SEI-2013-TN-029

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
HQ ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2116

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified/Unlimited, DTIC, NTIS

12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note is one in a series of publications from the Software Engineering Institute intended to aid United States Department of Defense acquisition professionals in the use of Agile software development methods. As the prevalence of suppliers using Agile methods grows, these professionals supporting the acquisition and maintenance of software-reliant systems are witnessing large portions of the industry moving away from so-called “traditional waterfall” life cycle processes. The existing infrastructure supporting the work of acquisition professionals has been shaped by the experience of the industry—which up until recently has tended to follow a waterfall process. The industry is finding that the methods geared toward legacy life cycle processes need to be realigned with new ways of doing business. This technical note aids acquisition professionals who are affected by that realignment.

14. SUBJECT TERMS
Agile, waterfall, metrics

15. NUMBER OF PAGES
58

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102