Estimating Agile Software Development Programs

NASA CAS 2016

Bob Hunt

Galorath Federal Incorporated
Agenda

• Background

• Agile/Hybrid Agile software development

• Establishing a process for developing the technical, cost and schedule baseline

• Identifying critical software management metrics, and

• Applying Earned Value Management to Agile Programs

• Summary
IT Project Success

20% to 30% Failures

Why they fail

Source: Gartner (June 2012)
Software Development

• While there are many approaches to Software Development, they can generally be placed into 2 categories:
  • **Plan Driven** – following a version of the Waterfall Development Process
  • **Iterative Driven** – following a “version” of the Agile Development Process

• Plan Drive programs have an assumption of some reliable/realistic size metric, for example:
  • Source Lines of Code (SLOC)
  • Function Points
  • Use Cases, User Stories, Web Pages
What is Agile Software Development?

- In the late 1990s, several methodologies received increasing public attention
- Each had a different combination of old, new, and transmuted old ideas, but they all emphasized:
  - Close collaboration between the programmer and business experts
  - Face-to-face communication (as more efficient than written documentation)
  - Frequent delivery of new deployable business value
  - Tight, self-organizing teams
  - And ways to craft the code and the team such that the inevitable requirements churn was not a crisis
How Formal Is Agile?

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.”

Agile is NOT a Method – it’s a mindset!
Individual Methods are Formal – sort-of
Agile 101 – How It Works

- **Agile is a set of software development methods in which solutions evolve through collaboration**
  - Integrated teams include PeopleSoft SMEs to configure the product, PeopleSoft Developers, Data Integration Developers, and Testers

- **Development is iterative**
  - The traditional software development phases are seen as continuous activities
  - Work is broken into smaller tasks
  - Multiple iterations may be required to release a product
  - Documentation is created as-built

- **For each iteration, a working product is demonstrated to stakeholders**

- **Emphasizes value-driven approach**
  - The usual project constraints still apply
  - Focusing on value allows most important functionality to be delivered first

- **Technology agnostic**
The waterfall approach is akin to painting by numbers, as it calls for a fully formed idea at the start, which is built incrementally, piece by piece without flexibility.

With Agile, we start with a concept: for IPPS-A the COTS product is the starting point.

Then, we iteratively build a rough version and validate it, slowly improving the definition and quality.
# Agile 101 – Key Agile Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scrum</td>
<td>A framework for team collaboration on complex software projects. 1-10 people (have seen up to 20)</td>
</tr>
<tr>
<td>Sprint</td>
<td>A short multiple-week period where a team completely builds working, tested software. All phases of the SDLC are executed iteratively during a sprint – Analysis, Design, Code, Test. 1-6 weeks (have seen up to 13 weeks) <em>(13 conveniently give 4 sprints per year)</em></td>
</tr>
<tr>
<td>Feature</td>
<td>A set of specifications that can be shown in a user demonstration and oriented on system capabilities.</td>
</tr>
<tr>
<td>Epic</td>
<td>A description of how work gets done using the new software (To-Be business process).</td>
</tr>
<tr>
<td>Spike</td>
<td>A special type of story used for research and prototyping activities, which can be functional or technical.</td>
</tr>
<tr>
<td>Backlog</td>
<td>A single definitive repository for all upcoming work. It consists primarily of future features intended to address user needs and deliver business benefits, as well as architectural features required to build the product.</td>
</tr>
</tbody>
</table>
Agile Implementation

To create working software product often and to demonstrate to the customer

- Sprint Backlog
- Sprint Burndown
- Impediment List

Ensures holistic activities and iterative processes
Agile 101 – Sprint Breakdown

- **Sprint Planning Meeting**
  - Product owner and scrum team meet at the start of the sprint to review the sprint goal, including a review of the requirements/features to be accomplished and corresponding tasks.

- **Daily Scrum/Daily Work**
  - Scrum team meets daily to review yesterday’s accomplishments, plan for today, and any blockers.
  - Scrum team performs design, build, and test tasks.

- **Sprint Review Demonstration**
  - Scrum team demonstrates sprint accomplishments (requirements and/or features implemented during that sprint) to the product owner.

- **Product Increment**
  - The sum of all the product backlog items completed across all the scrum teams through the last sprint.

- **Sprint Retrospective**
  - The scrum team meets to discuss what to keep doing, stop doing, and start doing. Focus on what is actionable for the next sprint.

- **Update Product Backlog**
  - The product owner continuously updates (adds to, reprioritizes, etc.) the product backlog.
Agile 101 – Recap

- Agile is a disciplined methodology.
- Agile is not...
  - ...unlimited or uncontrolled scope.
  - ...unplanned.
  - ...undocumented.
  - ...unverified.
  - ...mini waterfall.
  - ...trial and error.
  - ...a synonym for flexible.
  - ...a synonym for fast.
Agile 101 - Summary

- **Waterfall faces challenges in a large-scale ERP implementation.**
  - Significant effort to build and maintain momentum
  - High risk of a resulting product that does not meet needs

- **Looking for an alternative:**
  - Best candidate is Scaled Agile Framework (SAFe)
  - Lower risk implementation
  - Produces better results, meets IPPS-A Vision
  - Allows us to confirm that we are building the right thing
  - Gets early buy-in
  - Reduces the risk of rework when it is too late and more costly to change
  - Improved understanding of progress and cost

- **Better software development process, better user experience:**
  - Break work into smaller, more manageable segments
  - **Measure progress based on working software product**
  - **Drive functionality working on through to completion**
  - Emphasize showing working versions of product early and often to validate
  - Use product to review whether it meets requirements – shift to As-Built documentation
  - Use government time and resources better by reviewing software, not paper
  - Use cumulative assembly testing for earlier validation of product
  - Have automated regression test bed available on go-live
Release 1 (made up of multiple Themes/Increments)

Theme/Increment 1

Epic 1
Feature 1
User Story 1

Epic 2
Feature 2
User Story 2

Epic 2
Feature 3
User Story 3

EVM work Packages identified at Epic or Theme level

Sprints
Cost Estimating done at the Sprint Level

* These “building blocks” are program specific and may be called by different names
Hybrid Agile Development/Acquisition

INTEGRITY  INNOVATION  EXCELLENCE

Testing and Sustainment?
Agile Release and Sprint Backlog

Product Backlog

Release Backlog

Feature Backlog

Preliminary Design Planning

Conduct Fit/Gap Build

Conduct Framework Build

Conduct High/Medium Priority Build

Sprint Backlogs

Feature 1

Feature 2

Feature n...

Feature 1, Sprint 1

Feature 3, Sprint 1

Feature 3, Sprint 2

Feature 4, Sprint 1

Feature 4, Sprint 2

Feature 2, Sprint 1

Feature 2, Sprint 2

Feature 2, Sprint 1

IBR

F/G

Increasing use of Agile Methods

Agile methods can start earlier in the process because we are starting with a COTS product baseline.

User Feedback, Defects, and New Features
How to Scale

- Agile methods are generally used for projects of smaller scope.

- Scaled Agile Framework (SAFe) builds from Agile methods to provide a construct for large-scale implementations.

- A playbook tailored to IPPS-A will inform the development of the IMS and form the basis to guide teams.

  - Introduces features to bridge the gap between epics (business processes) and sprints.

  - Introduces spikes to explore various approaches to address key foundational decisions.
How to Define Work

1 - Requirements are defined based on the Business Process.
2 - Requirements are grouped into features aligned with the delivered software product.
3 - Features are decomposed into tasks to be executed in sprints.

- Features will be classified as one of the following types:
  - Framework: Foundational capabilities that need to be completed early as they will be leveraged throughout the solution.
  - Core: Capabilities support transactional processing. These include setup tables that maintain valid values and capabilities to manage employee records.
  - Self-Service: Any feature that has a self-service component is identified.
  - Information: Reports, queries, dashboards.
  - Data: Conversions and interfaces.
How to Apply Agile in a Non-Agile World

Plan → SRR → SFR → IBR

Analyze:
- Requirements Analysis
- Fit/Gap

Deploy:
- Operational Testing
- Integration Testing

Build:
- Detailed Design
- Unit Testing
- Develop

PDR → CDR → PLT

TRR → DIT → GAT → LUT → OPT → DEP

Waterline
Galarth: Driving The State of the Art (10 Step Estimation Process)

When performance is measured performance improves

Estimation processes are independent of tools

1. Establish Scope
2. Establish Technical Baseline, Ground Rules, Assumptions
3. Collect Data
4. Estimate and Validate Software Size
5. Prepare Baseline Estimates
6. Review, Verify and Validate Estimate
7. Quantify Risks and Risk Analysis
8. Generate a Project Plan
9. Document Estimate and Lessons Learned
10. Track Project Throughout Development

We will use these 10 steps in our Estimation Process Improvement Program (EPIP)
Fundamental Model Assumptions

Fundamental assumptions of most Software Estimating Models

- There is a fixed relationship between size and effort, e.g.
  \[(\text{Effort}^n) \times \text{Time} = \frac{\text{Size/Technology}}{}\]
- Results are then modified by current trends and analyses
- Total effort can be distributed by a mathematical model; e.g. Weibull, Rayleigh
Five Estimating Methodologies

• **Methodology 1:** Since many Agile programs are fixed price, it is often just a matter of labor rates times quantity

• **Methodology 2:** Simple Build-up approach based on averages can be defined as:
  - Sprint Team Size (SS) x Sprint length (Sp time) x Number of Sprints (# Sprints)

• **Methodology 3:** Structured approach based on established “velocity” – most often used internally by the developer since detailed/sensitive data are available to them

• **Methodology 4:** Automated Models approach based on a size metric – which may be difficult to quantify

• **Methodology 5:** Factor/Complexity approach based on data generated in early iterations
Methodology 5: Factor/Complexity Approach

<table>
<thead>
<tr>
<th>Scale</th>
<th>Functional Description</th>
<th>Effort Multipliers</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Significantly less functionality to be delivered</td>
<td>0.5</td>
</tr>
<tr>
<td>-</td>
<td>Moderately less functionality to be delivered</td>
<td>0.7</td>
</tr>
<tr>
<td>-</td>
<td>Slightly less functionality to be delivered</td>
<td>0.9</td>
</tr>
<tr>
<td>=</td>
<td>Functionality equivalent to Increment X</td>
<td>1.0</td>
</tr>
<tr>
<td>+</td>
<td>Slightly more functionality to be delivered</td>
<td>1.3</td>
</tr>
<tr>
<td>++</td>
<td>Moderately more functionality to be delivered</td>
<td>1.7</td>
</tr>
<tr>
<td>+++</td>
<td>Significantly more functionality to be delivered</td>
<td>2.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scale</th>
<th>Complexity Description</th>
<th>Effort Multipliers</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Significantly less complex</td>
<td>0.7</td>
</tr>
<tr>
<td>-</td>
<td>Slightly less complex</td>
<td>0.9</td>
</tr>
<tr>
<td>=</td>
<td>Complexity equivalent to Increment X</td>
<td>1.0</td>
</tr>
<tr>
<td>+</td>
<td>Slightly more complex</td>
<td>1.3</td>
</tr>
<tr>
<td>++</td>
<td>Significantly more complex</td>
<td>1.7</td>
</tr>
</tbody>
</table>

- These initial set of factors came from the environmental factor from traditional software cost models
- Because each Increment is a mini project, use a Rayleigh or simple Beta Curve (such as a 60/50 Beta curve) to phase costs
## What to measure

### WHAT TO MEASURE

<table>
<thead>
<tr>
<th>Information Category</th>
<th>Measurable Concepts</th>
<th>Prospective Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule and Progress</td>
<td>Milestone completion</td>
<td>Milestone Dates</td>
</tr>
<tr>
<td></td>
<td>Critical Path Performance</td>
<td>Slack Time</td>
</tr>
<tr>
<td></td>
<td>Work Unit Progress</td>
<td>Requirements Traced, Requirements Tested, Problem Reports Opened, Problem Reports Closed, Reviews Completed, Change Requests Opened, Change Requests Resolved, Units Designed, Units Coded, Units Integrated, Test Cases Attempted, Test Cases Passed, Action Items Opened, Action Items Completed</td>
</tr>
<tr>
<td></td>
<td>Incremental Capacity</td>
<td>Components Integrated, Functionality Integrated</td>
</tr>
<tr>
<td>Resources and Cost</td>
<td>Personnel Effort</td>
<td>Staff Level, Development Effort, Experience Level, Staff Turnover</td>
</tr>
<tr>
<td></td>
<td>Financial</td>
<td>BCWS, BCWP, ACWP, Budget, Cost</td>
</tr>
<tr>
<td></td>
<td>Environmental/Support</td>
<td>Quality Needed, Quality Available, Time Available, Time Used</td>
</tr>
<tr>
<td>Product Size &amp; Stability</td>
<td>Physical Size/Stability</td>
<td>Database Size, Components, Interfaces, LOC</td>
</tr>
<tr>
<td></td>
<td>Functional Size</td>
<td>Requirements, Function Changes, Function Points</td>
</tr>
<tr>
<td>Product Quality</td>
<td>Functional Correctness</td>
<td>Defects, Age of Defects, Technical Performance</td>
</tr>
<tr>
<td></td>
<td>Maintainability</td>
<td>Time to Release, Cyclomatic Complexity</td>
</tr>
<tr>
<td></td>
<td>Efficiency</td>
<td>Utilization, Throughput, Response Time</td>
</tr>
<tr>
<td></td>
<td>Portability</td>
<td>Stand Comp-Oliance</td>
</tr>
<tr>
<td></td>
<td>Usability</td>
<td>Operator Errors</td>
</tr>
<tr>
<td></td>
<td>Reliability</td>
<td>MTTF</td>
</tr>
<tr>
<td>Process Performance</td>
<td>Process Compliance</td>
<td>Reference Maturity Rating, Process Audit Findings</td>
</tr>
<tr>
<td></td>
<td>Process Efficiency</td>
<td>Productivity, Cycle Time</td>
</tr>
<tr>
<td></td>
<td>Process Effectiveness</td>
<td>Defects Contained, Defects Escaping, Rework Effort, Rework Components</td>
</tr>
<tr>
<td>Technology Effectiveness</td>
<td>Technology Suitability</td>
<td>Requirements Coverage</td>
</tr>
<tr>
<td></td>
<td>Technology Volatility</td>
<td>Baseline Changes</td>
</tr>
<tr>
<td>Customer Satisfaction</td>
<td>Customer Feedback</td>
<td>Satisfaction Rating, Award Fee</td>
</tr>
<tr>
<td></td>
<td>Customer Support</td>
<td>Request for Support, Support Time</td>
</tr>
</tbody>
</table>

* Practical Software Measurement; McGarry, Card, Jones; Addison-Wesley 2002
Earned Value Management Process

Define the Work
- SOW - WBS - WBS dictionary

Plan the Work
- IMP/IMS, Control Accounts/Work Packages, EV Methods

Work the Plan
- Execute the tasks and activities in the plan

Collect Results
- Actual labor and other direct costs

Measure Performance Against the Plan
- Status EV and schedule

Analyze Deviations
- Cost and Schedule Variances

Institute Appropriate Corrective Actions
- Baseline Change Requests

Change Control
- Engineering Change Proposals
Plan for Delivery of Features Versus Actual Delivery
(This chart will provide a good “Top Level” assessment)

**Summary:**
- Use a chart like this and the following two charts for estimating feature velocity/work performance.
Feature Point Delivery

Feature Point Plan & Performance
(This Chart provided an assessment of “Technical” accomplishment)

Summary: Overall the teams are currently operating at a 5.2 Feature Point velocity per Increment which equates to slightly less than planned.
- Performance Based Estimate (Average) – This projection is based on a 5.2 Feature Point average.
- Performance Based Estimate (%) – This projection is based on an 80% efficiency.
- Program Based Estimate – This projection is based on team expectations.
Feature Velocity

A Rate at which Features Will be Accomplished (Velocity) will be established and Progress against that goal will be tracked

Summary: Overall the teams are currently operating at a 5.2 Feature Point velocity per Increment which equates to slightly less than planned.

- Performance Based Estimate (Average) - This projection is based on a 5.2 Feature Point average.
- Performance Based Estimate (%) - This projection is based on an 80% efficiency.
Summary

• Fixed Price and/or LOE contracts in the early phases should be written so that key “value-added” metrics are collected and reported during each increment

• Estimators may have to employ a variety of software estimating methodologies within a single estimate to model the blended development approaches being utilized in today’s development environments
  • An agile estimating process can be applied to each iteration/sprint
  • Future Increments can be estimated based on most recentsuccessful IID performance

• Cost estimators will have to scrutinize these programs like a schedule analyst might to determine the most likely IOC capabilities and associated date
  • The number of increments are an important cost driver as well as an influential factor in uncertainty/risk modeling
Summary

• All of the estimation methods are susceptible to error, and require accurate historical data to be useful within the context of the organization.

• When developers and estimators use the same “proxy” for effort, there is more confidence in the estimate.
Recommended Reading

- “The Death of Agile” blog
- “Agile Hippies and The Death of the Iteration” blog
- Story Point Inflation
Endnotes

Additional References

- George, J., & Rodger, J. (2010). *Smart Data (Enterprise Performance Optimization Strategy)*.
Contact Information

- Bob Hunt
  - Email: BHunt@Galorath.com
  - Phone: 703.201.0651