Agile Estimation for Space and Safety Critical Software

Arlene Minkiewicz
Chief Scientist
PRICE Systems, LLC
Agenda

• Introduction
• Agile Software Development
• Agile for Space and Safety Critical
• Size and Cost Estimation for Agile Software Development
• Wrap Up
Introduction

• Agile development practices have enabled organizations to deliver quality software that optimizes customer satisfaction
  • But is agile for every type of project

• Space and other mission critical software have high reliability, fault tolerance requirements with strict safety and performance criteria

• Organizations developing space based and safety critical software are looking for ways to do development faster, better and cheaper

• Can agile development practices facilitate this requirement
Introduction

• Back in the day … Complexity of applications was overshadowed by the logistics of implementation
• Technology improved … today software solves increasingly complexity problems
• The so called ‘software crises’ (mid 60’s to 80’s) resulted in many ‘silver bullet’ type solutions
• Lots of smart software development professionals began looking for more lightweight methods to address complexity in achievable chunks
Agile Software Development
Agile Manifesto

• We are discovering better ways of developing software by doing it and helping others do it
  • Individuals and interactions over processes and tools
  • Working software over comprehensive documentation
  • Customer collaboration over contract negotiation
  • Responding to change over following a plan

• All agile projects adhere to this manifest
• All agile projects share a common set of principles
• Each agile project uses a unique set of agile practices to implement these principles
• Successful estimation for an agile project is like software estimation for any project – you need to understand the project properties and the practices employed
Traditional Software Development

- Requirements are analyzed
- Architecture and design are created
- Requirements are implemented, tested and delivered
- Months (or longer) occur before there is usable software for the customer to evaluate
Agile Software Development

- Usable chunks of software are developed in short periods of time (sprints, iterations, etc.)
- Requirements are translated into user stories and become the project backlog
- User stories deliver business value and are small enough to complete in an iteration
- Customer works with team and reviews software regularly
- Each iteration focuses on the user stories that are currently the highest priority of the customer
- Priorities may shift from iteration to iteration
- Agile teams expect and embrace change
12 Guiding Principles for Agile Development
Common Agile Practices

- Pair programming
- Continuous integration with automated testing
- Test driven development
- Daily stand up meetings
- Co-located teams
- Code refactoring
- Small releases
- Customer on team
- Simple design
### Agile in Space

- **“NASA was agile before agile was a common term”**
  - Jim Highsmith – one of the 17 original authors of the manifesto – worked for NASA at one point

- **In 1962 – John Paup was a senior NASA manager planning part of the Apollo program**
  - First thing every morning all key people reported to his office for a stand up meeting

- **NASA Ames – Mission Control Technologies**
  - Adopted a hybrid agile solution – segregating activities constrained by mission criticality from those more standard development activities

- **Cassini Mission**
  - 2015 (more than 10 years after the mission started) – the maintenance team has adopted a hybrid agile process for software changes

- **Software Probe Plus – built by JHU/APL**
  - Several of the ground system software module teams are adopting agile practices
Challenges to Being Agile in Space

- **Requirements for Compliance to industry standards and sponsor requirements**
  - CMMI
  - AS9100
  - NASA Software Engineering Requirements (NPR 7150.2B)
  - European Cooperation for Space Standardization (ECSS)

- **Requirements for detailed documentation**
- **Requirements flexibility (or lack there of)**
- **Detailed up front planning**
- **Requirements for specialized capability (as opposed to agile teams composed of generalists)**
- **Formalized customer interfaces**
But agile is a philosophy not a development process

- Hybrid applications make the most sense for space systems
- Agile practices that make sense
  - Small teams evolving product in small visible steps
  - Daily stand up meetings
  - Pair programming
  - Continuous automated testing
  - Test driven development
  - Collaborative planning (including the customer)
- Agile practices less likely to make sense
  - Evolving requirements
  - No formal up front planning
  - Little to no documentation
  - Refactoring
Agile Cost Estimation
Agile estimation

• **Frequently asked questions**
  • How to estimate size for an agile project when the team is working with Story Points?
  • What other cost drivers are indicated for an agile development project?
Agile Size Estimation

- Agile teams do a lot of their own estimation
  - High level estimation as the backlog is created in the beginning of a project (Sprint 0)

- Estimates are notional and only make sense to the team
  - Story points
  - T-Shirt size

- Estimators challenge is to translate the teams knowledge into a size measurement that relates to their Cost Estimating Relationships (CERs)

- In the context of a parametric model – agile size measures actually combine two typical cost drivers
  - Size
  - Complexity
Fortunately agile teams collect lots of metrics
Agile Size Estimation

- Study of PRICE’s agile data found no correlation between story points and software size or effort.
- Did find a significant relationship between software size and complexity (Functional Complexity in the PRICE model) pairs and effort.
Agile Cost Drivers

- The fact that your project is agile is not a cost driver
- There are potential cost implications to adopting agile practice
- Estimation team needs to determine which agile practices apply
Agile Cost Drivers

- **Agile teams tend to be highly skilled**
- Hard to be a slacker in an agile environment
- Working closely with high skilled team members, learning curve for new members is quick
- Input parameters to your model indicating team experience would be affected

- **Agile teams tend to have tool sets that are quite sophisticated**
- This would be especially true on teams working with space systems as it would greatly facilitate compliance to standards
- Input parameters around tools or automation would be affected
Agile Cost Drivers

- **Co-location of teams should improve team productivity**
  - Culture of interruption
  - Questions answered in real time
  - Team cohesion increases
  - Co-locating stakeholders and SMEs with development team creates a real time IPT
  - Well run stand-up meetings increase productivity and quality
  - Cost drivers indicating distribution of team and communication practices would be affected
Agile Cost Drivers

- Continuous integration with automated testing should increase delivery productivity
  - Important in space systems to maintain safety critical compliance requirements.
  - Code is checked in frequently and builds are run and test regularly before developers forget what they changed
  - Red tests raise red flags – team fixes them right away
  - Since little code is changed, errors are east to track down
  - Fixes occur quickly
  - Cost drivers focused on integration test complexity would be affected
Conclusion

• While not all agile practices make sense for space systems development, there are many that can (and have) improved the ability to deliver high quality space system software
• A hybrid version of agile is most appropriate for safety critical software
• NASA has been successfully employing agile on many programs for many years
• Estimating an agile program is no different that estimating any other software
  • Understand the program and the process being employed
  • Study data from previous similar programs
  • Discuss project particulars with the delivery team