Agile Acquisition

Peter Modigliani

10 Dec 12

Presented to:

Mr. Koen Gijsbers

General Manager NATO Communications and Information Agency

MITRE

Purpose / Outline

To highlight how Agile development can turn around a broken IT Acquisition environment to rapidly deliver capabilities

- **Current IT Acquisition Challenges**
- **Agile Development Overview**
- **How IT Acquisition Can Be More Agile**

MITRE

DoD IT Acquisition Challenges

- **The Defense Acquisition Framework is too large, complex, and slow to effectively acquire IT capabilities**
  - Framework built for major weapon systems (aircraft, ships, tanks)
- **Major IT systems average 91 months* to deliver capabilities**
  - An IT system today would deliver capabilities conceived in 2005
- **DoD Enterprise Resource Planning (ERP) programs are $6B over budget and 31 years behind schedule****
  - Cannot replace dozens of legacy systems with a new $1B system

* Defense Science Board Report on IT Acquisition
** Sen McCaskill at Congressional Hearing

MITRE

Large Software Projects Rarely Succeed

Software Project Labor Costs	Probability of Success
<$750K	71%
$750K - $3M	19%
>$3M	2%

Source: Standish Group Chaos Report 2009

MITRE

Agile Software Development

- **Focused around small, frequent releases of capabilities**
- **Working software valued over comprehensive documentation**
- **Responsive to changes in operations, technology, budgets, etc**
- **Active collaboration of users, developers, other stakeholders**

Agile Terms and Timelines

Release
6 Months

Capability to deliver to users
Comprised of multiple sprints

Sprint
1 Month

Priority capabilities developed, integrated, and tested
Demonstrated to users with potential to deliver

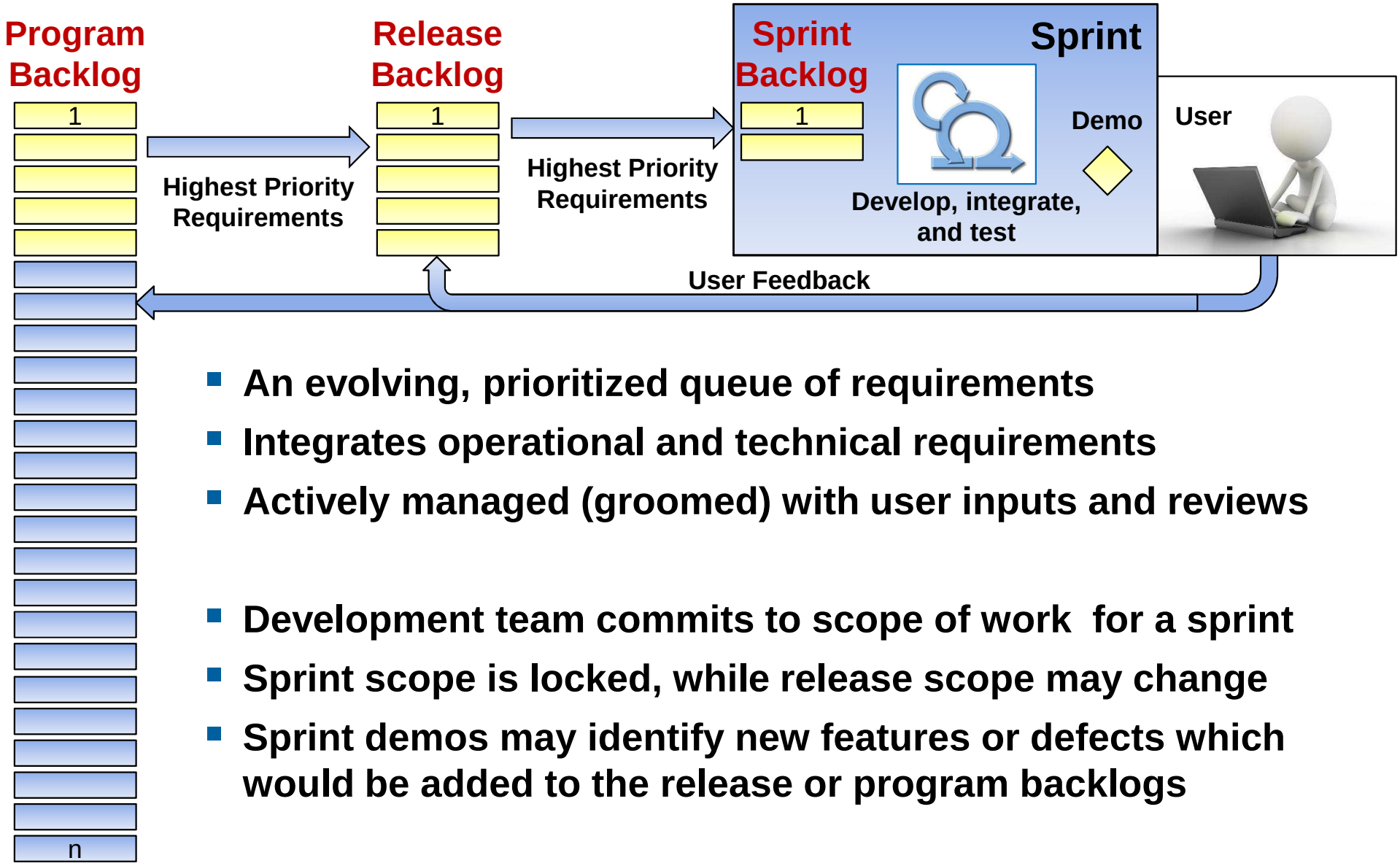
Scrum
Daily

Small, self-organizing teams plan development activities
Review progress and identify impediments

Agile terms and timelines will vary based on the dozens of development approaches

MITRE

Agile Requirements Are Managed Via Backlogs



- An evolving, prioritized queue of requirements
- Integrates operational and technical requirements
- Actively managed (groomed) with user inputs and reviews

- Development team commits to scope of work for a sprint
- Sprint scope is locked, while release scope may change
- Sprint demos may identify new features or defects which would be added to the release or program backlogs

MITRE

Agile Requirements Can Be in User Stories

As a [user role], I want to [goal] so I can [reason]

As a registered user, I want to log in so I can access subscriber-only content

- **Concise, written descriptions of a capability valuable to a user**
- **High-level description of features**
- **Written in user language, not technical jargon**
- **Provides information to estimate level of effort**
- **Small and succinct**
- **Worded to provide a testable result**
- **Traceable to overarching mission threads**

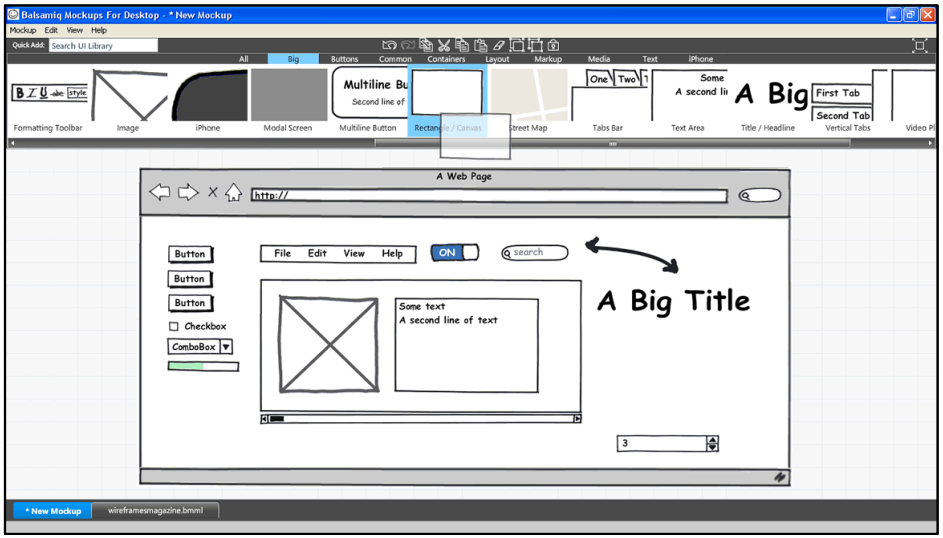
MITRE

Storyboards and Mockups Help Team Visualize the System and Features

Storyboards

Narrative visual depictions set in time to describe system use

Mockups

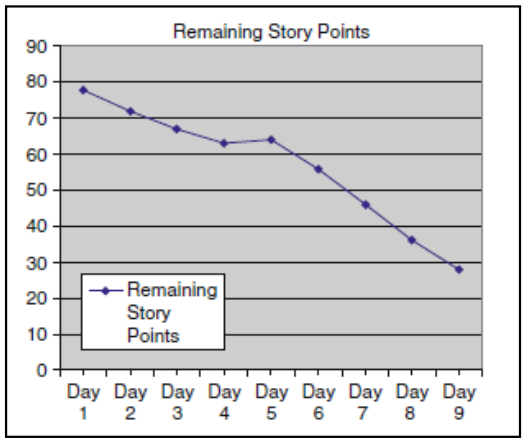


Visual depictions of the feature of the system

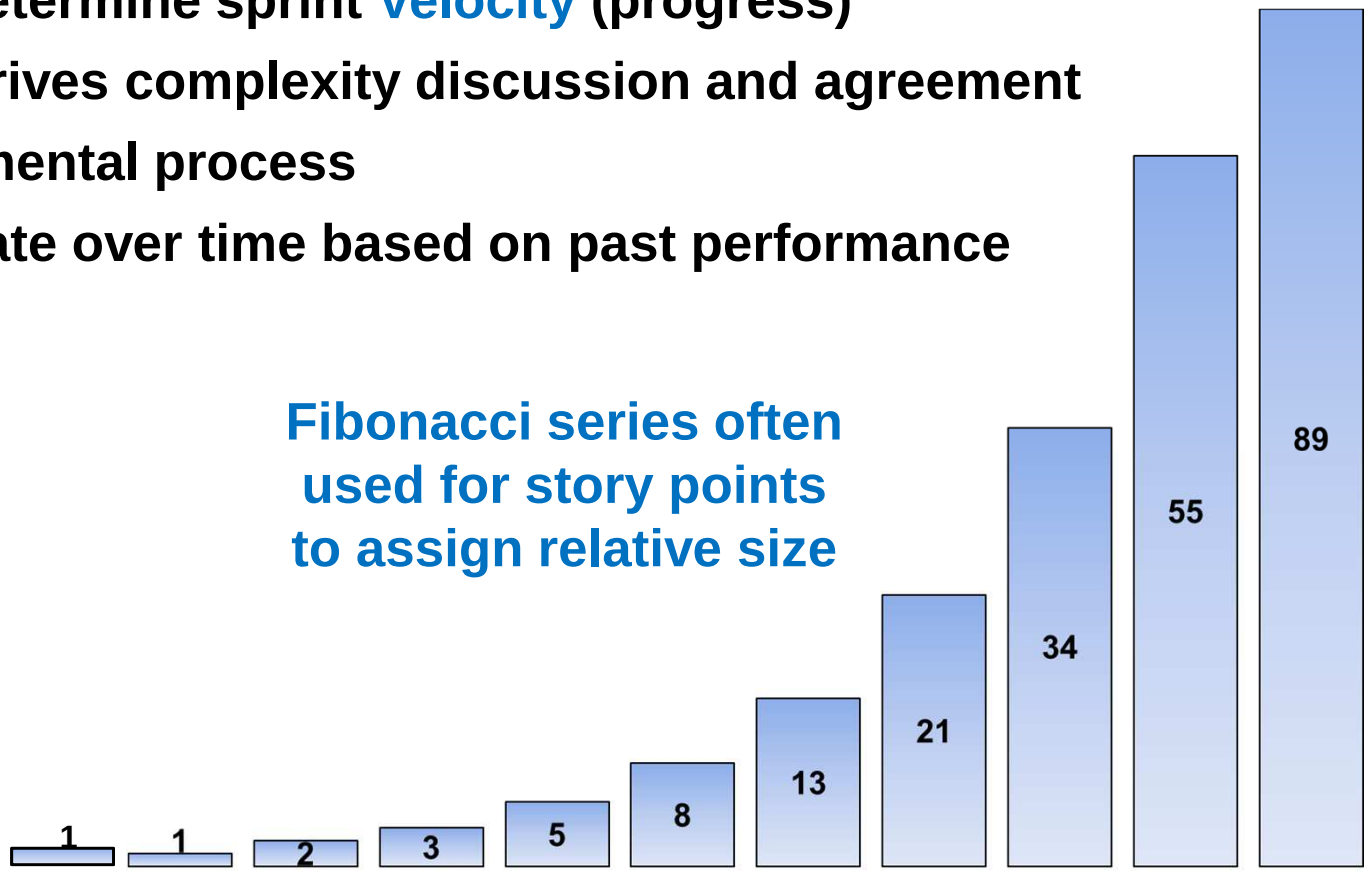
MITRE

Agile Estimation

- User stories can be estimated using story points, staff hours, etc.
- Performed by the development team to gauge complexity
- Sizing is used to determine sprint Velocity (progress)
- Determining size drives complexity discussion and agreement
- Iterative and incremental process
- Increasingly accurate over time based on past performance



Fibonacci series often used for story points to assign relative size



MITRE

Planning

Release

- Strategic – High level
- Considers user stories to develop for the major releases
- Conducted after initial product backlog developed
- Outlines intent, not a commitment
- Revised after every sprint

Sprint

- Tactical level details
- Commit to a set of user stories
- Development team and product owner

Daily

- Development Team stand-up
  - What have you done?
  - What are you going to do?
  - Any obstacles?

MITRE

Sprint Execution

Design	Develop	Integrate	Test	Review	Demo
How to go from user story to code	Develop code and track tasks	Continuously, at least daily	Automated and integrated testing	How team can improve for next sprint	Demo functionality to users and stakeholders

Run through the full process monthly

MITRE

Core Agile Roles

- **Product Owner**
  - Manages the backlog(s) and requirements prioritization
  - Responsible for understanding and communicating users and stakeholders operational concepts to development team
- **Scrum Master**
  - Facilitates the process, shields the team from distractions
  - Enforces rules and keeps team focused on tasks
- **Development Team**
  - Self organizing team comprised of 5-9 members
  - Developers, software and security engineers, data specialists, testers, etc

Traditional vs Agile Practices

Traditional Practices		Agile Practices
Completely defined in detail up-front	**Requirements**	Iteratively refined during development
Detailed cost estimates and full funding	**Risk Reduction**	Incremental releases and sprints
Early, large, and document-intensive	**Reviews**	Small, frequent, and often informal
Process and documentation	**Emphasis**	Knowledgeable, empowered teams
Detailed plans freeze solution early	**Baselines**	Adapted to new info in development
At end of an increment (years)	**Delivery**	**At end of a release (months)**
Earned value measures against plan	**Measurement**	Frequent capability deliveries
Independent, following development	**Testing**	Daily development, integration, test
Acceptance at end of increment	**Users**	Active for continual review and feedback

12 Best Practices for Agile Programs

1. Integrate users throughout the development lifecycle
2. Hold user conferences
3. Structure program to deliver capabilities every 12 months
4. Require developers to deliver "regular" iterations
5. Use Agile IT contracting
6. Perform ongoing, incremental systems engineering reviews
7. Conduct continuous, competitive prototyping
8. Integrate testing, certification, and accreditation activities
9. Integrate Agile IT expertise
10. Leverage common IT develop, test, and production platforms
11. Leverage capstone documents
12. Use collaboration software

MITRE

Scaling Agile for Large Programs

- Integration requires sound system engineering discipline
- Robust enterprise architecture guides individual developments
- System performance design and testing on system and components
- Strategies, backlogs, and roadmaps define clear program structure
- Cross-Team integration requires frequent collaboration on issues

MITRE

Agile Is More Responsive to Changes

Agile Program
$10M

Project 1: $1M
Project 2: $1M
Project 3: $1M
Project 4: $1M
Project 5: $1M
Project 6: $1M
Project 7: $1M
Project 8: $1M
Project 9: $1M
Project 10: $1M
Project New: $1M

How would each program respond to:

An under-performing contractor?

Schedule slips?

A $2M budget cut?

A new short-term user requirement priority?

Traditional Program
$10M

Program
$10M

MITRE

Risks for Agile Adoption

- **Resistance to adopt Agile from leadership, program office, or users**
  - Education and demonstration of success is critical early/often
- **Managing the many piece parts**
  - Increased emphasis on SE, Integration, Collaboration of activities
- **Imposing large system bureaucracy constraints**
  - Oversight, documentation, and reviews must be streamlined
- **Lengthy contracting, testing, and certification timelines**
  - Get stakeholders involved from start to operate within timelines
- **Expecting Success From the Start**
  - Agile is a new paradigm that will take time to effectively integrate

Summary

- **Small, frequent deliveries reduce cost, schedule, and risk**
- **Active user and stakeholder collaboration responsive to changes**
- **Tighter integration of users, developers, testers, and engineers**
- **Agile requires radically different policies, culture, and mindset**

MITRE