Agile Systems Engineering: What is it and What Have We Learned?

March 2012

Dr. Suzette S. Johnson
Agile Engineering
Northrop Grumman
Suzette.Johnson@ngc.com
Getting To Know You!

Dr. Suzette Johnson

- Northrop Grumman Agile Community of Practice Chair
- Champion of Agile practices across NGC
- ADAPT Executive Committee (Industry working group)
- As a systems engineer, project manager, and certified Scrum Professional, with an interest and passion for promoting and implementing Agile practices in large-scale systems environments.
- Provides coaching, consulting, mentoring, training, etc. for Northrop Grumman programs
- Lean and Agile experience started in 1999
- Dissertation focused on investigating the impact of leadership styles on software project outcomes in traditional and agile engineering environments
Discussion Outline

- Defining an Agile Environment and Agile Systems Engineering (ASE)
- Requirements, Use Cases, User Stories
- Levels Planning
- User Story Verification and Validation
- Summary
- References
ASE: What Have We Learned About...?

- Systems Engineering and Team Collaboration
- Evolving Architecture
- Requirements and Delivering Value
- Verification and Validation within the Iteration
- Collaboration and Transparency with Stakeholders
INCOSE Definition of Systems Engineering

2.2 Definition of Systems Engineering
Systems engineering is a perspective, a process, and a profession, as illustrated by these three representative definitions.

Systems engineering is a discipline that concentrates on the design and application of the whole (system) as distinct from the parts. It involves looking at a problem in its entirety, taking into account all the facets and all the variables and relating the social to the technical aspect. (Ramo1)

Systems engineering is an iterative process of top-down synthesis, development, and operation of a real-world system that satisfies, in a near optimal manner, the full range of requirements for the system. (Eisner2)

Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs. (INCOSE3)

Concentrates on the design and application of the whole system

It focuses on defining customer needs and required functionality early in the development cycle....

An Agile Environment

- **Adaptive, Responsive, Evolving, Continuous Improvement**
- Improved transparency of progress
- End-to-end accountability and ownership
- Reduces time-to-deploy operational capability
- Ability to adapt to changing requirements and new technological advancements

*Agile is an empirical process*
Agile Principles

Early and Continuous Delivery of Value

A Working System is the Primary Measure of Progress

Welcome Changing Requirements

Deliver a Working System Frequently

Business People and Developers Must Work Together Daily

Motivated and Empowered Individuals

Face-to-face Conversation

Promote Sustainable Development

Continuous Attention to Technical Excellence

Simplicity

The Best Architectures, Requirements and Designs Emerge from Self-Organizing Teams

Regular Team Reflection on How to Become More Effective

http://agilemanifesto.org/
3.4.4 Agile Development

The preceding discussions have emphasized the benefits of orderly, hierarchical baseline progression followed by a corresponding verification sequence. Recognizing that the development process may require more flexibility in some circumstances, the Agile Development approach provides a tailoring framework, based on opportunity to simplify control methods and to assess the risks in so doing. The extent of tailoring is determined by whether the opportunity to shorten the project cycle is worth the risk of doing development steps out of sequence or in parallel.

The Agile Alliance (www.agilealliance.com) is dedicated to: developing iterative and agile methods, seeking a faster and better approach to software and system development, and challenging more traditional models. There are many articles describing the agile concepts. The key objective is flexibility, and allowing selected events to be taken out of sequence (see Figure 3-10 and Figure 3-11) when the risk is acceptable.

Our User Story for this Presentation

• **User Story**
  – As a Systems Engineer I want to have a better understanding as to how Systems Engineering practices align with the Agile framework so I have insight as to how this might be implemented in an Agile environment.

• **Acceptance Criteria**
  – Communicate how requirements analysis and design occur when using an Agile approach
  – Discuss considerations regarding architecture
  – Understanding of the relationship of requirements to user stories and why user stories
  – Insight into the Agile framework and how it helps manage changing requirements and priorities
  – Identification of verification and validation practices within the context of the Agile framework
The Agile Scrum Framework

- **Capabilities and User Stories**
  Prioritized by Product Owners
  Creates the release plan

- **The Daily Tasks**
  Managed by the Cross Functional Team

- **Design, Code, Integrate, Test**

- **Commitment**
  Identification of Impediments
  Communication

---

**Release Plan**

- **Product Backlog**

- **Iteration Backlog**

- **Daily Scrum Meeting**
  24 Hours
  2-4 Weeks Iteration
  Feature Demonstration and Retrospective

- **Potentially Shippable Product Increment**

---

**System Architecture**

**System Design Requirements**

**Inspect and Adapt**

**Visibility and Transparency**
Lesson Learned 1: Team Collaboration

• Systems Engineers are part of the Cross-Functional Team with regular collaboration at the team level and across the teams to provide an end-to-end systems view

• Chief Systems Engineer/Architect becomes more critical as you scale
Project Team Structure

- PM and Technical Lead
- Chief Engineer Chief Architect Quality
- End-to-End System Capabilities
  - Cross Functional Team 1
  - Cross Functional Team 2
  - Cross Functional Team 3
  - Cross Functional Team n
- Services
  - Network/ Systems Administration
  - Configuration Mgt.

Supports Cross Functional Teams

- Push accountability and ownership to the team level
- Everyone trained

An Example

Progress against end-to-end capabilities

Copyright 2012 Northrop Grumman Corporation
Project Team Structure

- PM and Technical Lead
- Chief Engineer
  - Chief Architect
  - Quality
- End-to-End System Capabilities
- Product Owner
- Scrum Master
- Developer
- Developer
- Integrator
- Configuration Management
- Tester
- Systems Engineer
- Developer

Supports Cross Functional Teams

- Network/ Systems Administration
- Configuration Mgt.

Progress against end-to-end capabilities

• Push accountability and ownership to the team level
• Everyone trained

An Example
## Project Team Structure

### Cross Team Collaboration

*Understanding the bigger picture*

A Cross-functional team is responsible for capability/thread from planning to final acceptance. Systems Engineers from the teams collaborate to maintain an end-to-end systems view.

<table>
<thead>
<tr>
<th>Role</th>
<th>Scrum Master</th>
<th>Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Owner</td>
<td>Developer</td>
<td>Systems Engineer</td>
</tr>
<tr>
<td>Developer</td>
<td>Systems Engineer</td>
<td>Integrator</td>
</tr>
<tr>
<td>Tester</td>
<td>Integrator</td>
<td>Configuration Management</td>
</tr>
<tr>
<td>Product Owner</td>
<td>Developer</td>
<td>Systems Engineer</td>
</tr>
<tr>
<td>Developer</td>
<td>Systems Engineer</td>
<td>Integrator</td>
</tr>
<tr>
<td>Tester</td>
<td>Integrator</td>
<td>Configuration Management</td>
</tr>
<tr>
<td>Product Owner</td>
<td>Developer</td>
<td>Systems Engineer</td>
</tr>
<tr>
<td>Developer</td>
<td>Systems Engineer</td>
<td>Integrator</td>
</tr>
<tr>
<td>Tester</td>
<td>Integrator</td>
<td>Configuration Management</td>
</tr>
</tbody>
</table>
Lesson Learned 2: Architecture

- Systems Engineering activities provide the context for the development activities (i.e. user stories as part of a systems capability)

- Architecture development is ongoing (emerging) and focuses on small increments/releases of value

- Emphasis on a modular architecture with well defined interfaces

- Iteration 0 (defined timeframe) to define enough of the architecture to get started; model in smaller increments (release/iteration level)

- Emerging versus Intentional Architecture

- Regular Architecture Meetings
Agile Systems Engineering: Delivering Value

- **Product Vision**
  - Captured Capabilities
    - Use Cases/Requirements
      - Architecture
      - Sequence Diagrams
      - Activity Flow Diagrams
    - System Level Validations
  - User Stories and Acceptance Tests
    - Functional
    - Non-Functional

**System Architecture**
Requirements mapped to stories
Revisit architecture and design each release and iteration

**Product Backlog**

**Product Roadmap**
- 6-9 months
- 1 – 6 months
- 1- 4 weeks
- Daily

**Release Planning**
- User Story 1
- User Story 2
- User Story 3
- User Story n...

**Sprint Planning**
- Iteration 1
- Iteration 2
- Iteration 3
- Iteration n...

**Daily Plans and Commitment**
Daily Stand-Up
Lesson Learned 3: Requirements

- Requirement analysis is done during release planning for that release
- Requirement development and deployment of highest priority capabilities first
- Every iteration learning from the previous iteration is fed into the next iteration planning and prioritization cycle
- Uses cases are a tool and provide the greater context for user stories
- High level requirements are decomposed into a set of user stories
- User stories convey understanding (user, need, why) and includes acceptance criteria
- User stories are more than functional requirements
The Product Backlog
The list of ranked or prioritized stories to be worked
Use Case to Scenario to User Story

1. Use Case

2. One Scenario within the use case

3. A user story is a segment of a scenario

As a [user/system] I want [what] so that [why]....
Example

**Requirement:** The hotel reservation system shall enable the user to make room reservations with confirmation of payment.

- **Use Case:** Vacation Planner makes hotel room reservation

- **Actor:** Vacation Planner

- **Success Guarantee:** Room reserved; credit card charged and confirmation received

- **Main Scenario:**
  - Search for discounted rooms.
  - Room identified.
  - Save selection.
  - Vacation Planner selects payment type
  - Vacation Planner submits credit card number, expiration date, name.
  - System validates card
  - System charges credit card
  - Confirmation number provided
The hotel reservation system shall enable the user to make reservations with confirmation of payment.

- As a premiere member, I want to search for available discounted rooms.
- As a vacation planner, I want to pay for my reservation with a credit card.
- As a vacationer, I want to search for available rooms.
- As a vacationer, I want to save my selections.
The hotel reservation system shall enable the user to make reservations with confirmation of payment.

User Story

As a vacation planner I want to pay for my reservation with a credit card so I can confirm my reservation.

Acceptance Criteria and Verification

Verify:
• Demonstrate with American Express
• Demonstrate with Master Card
• Demonstrate with Visa
What About Other Types of Requirements?

Non-functional requirements
Performance requirements
Constraints

As a vacationer and user of the hotel website, I want the system to be available 99.99% of the time...

As a vacationer, I want web pages to download in <4 seconds...

As the hotel website owner, I want 10,000 concurrent users to be able to access the site at the same time with no impact to performance...

Describes system behavior or characteristics

Reference: Mike Cohn, mountaingoatsoftware.com
Product Vision and Roadmap

Release 1  
March 31, 2012  
- Room reservations and payment  
- User profiles for future visits  
- Hotel amenities

Release 2  
June 30, 2012  
- Conference offerings  
- Online chat support  
- Local information

Release 3  
Sept. 30, 2012  
- Special discounts for room reservations  
- Improve usability  
- Google maps

Release 4  
Dec. 20, 2012  
- Air and hotel package deal  
- Meeting and Business plans and reservations

 Longer term planning  
High level capabilities (sometimes written as goals)
**Requirement 1:** The hotel reservation system shall allow the user to make reservations.

### Release Plan (Set of User Stories)

<table>
<thead>
<tr>
<th>User Stories</th>
<th>Points</th>
<th>Iteration Plan (User Stories with Tasks)</th>
<th>The Daily Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>As a vacationer, I want to search room availability...</td>
<td></td>
<td><strong>Design Review</strong> 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Install Baseline</strong> 4</td>
<td></td>
</tr>
<tr>
<td>Demonstrate with search on 1 room</td>
<td>12</td>
<td><strong>Documentation</strong> 8</td>
<td></td>
</tr>
<tr>
<td>Demonstrate with search on executive suite...</td>
<td></td>
<td><strong>Acquire Test Data</strong> 8</td>
<td></td>
</tr>
<tr>
<td>As a vacationer, I want to save my request...</td>
<td>8</td>
<td><strong>Code</strong> 24</td>
<td></td>
</tr>
<tr>
<td>Demonstrate...</td>
<td></td>
<td><strong>Develop Tests</strong> 8</td>
<td></td>
</tr>
<tr>
<td>As a vacationer, I want to pay with a credit card...</td>
<td>21</td>
<td><strong>Run Tests</strong> 8</td>
<td></td>
</tr>
</tbody>
</table>

**Yesterday I started on the interface.... Today I plan to... The one thing standing in my way...**

**Detailed planning saved to the last responsible moment**

- 1. User Stories
- 2. Tasks under a User Story
- 3. Preparing for the next iteration

---

Copyright 2012 Northrop Grumman Corporation
Lesson Learned 4: Validation and Verification

- Validation is an important function of the iteration
  - Occurring at the beginning and the end
- Testing is defined before development for the release/iteration begins
- Testers collaborate daily as part of the Cross Functional Team
The Agile Framework

Capabilities and User Stories
Prioritized by Product Owners
Creates the release plan

The Daily Tasks managed by the Cross Functional Team

Design, Code, Integrate, Test

Commitment
Identification of Impediments

Verification
Some independent testing

Validation

Daily Scrum Meeting

2-4 Weeks Iteration

24 Hours

Feature Demonstration and Retrospective

How do you define "release"?

Potentially Shippable Product Increment

Validate

Release Plan

Product Backlog

Iteration Backlog

Capability
User Story

Prioritized by
Product Owner

Creates the release plan

The Daily Tasks managed by the Cross Functional Team

Design, Code, Integrate, Test

Commitment
Identification of Impediments

Validation

Daily Scrum Meeting

2-4 Weeks Iteration

24 Hours

Feature Demonstration and Retrospective

How do you define "release"?

Potentially Shippable Product Increment

Validate
Iteration Demonstration and Acceptance

- Transparency and information sharing
- Team presents what it accomplished during the iteration
- Typically takes the form of a demo of new features or underlying architecture
- Time-boxed
- Whole team participates
- Feedback from stakeholders and users
- User Stories validated and accepted
- Metrics updated

User Story Validation
Requirements Mapping

- Requirement to story mapping
- Requirement to Story to Test to Verification
- Updated each iteration

### Requirement to Story to Test to Verification

<table>
<thead>
<tr>
<th>ID</th>
<th>Test</th>
<th>Spec Paragraph</th>
<th>Test</th>
<th>Verification Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS-19707</td>
<td>(U) The SYSTEM shall collect and maintain metrics on the number of users logged in on the system: Total number, average daily; max/min simultaneously logged on, daily totals of users logged in by organization.</td>
<td>3.1.1.6, (U) Infrastructure Test Objectives</td>
<td>Inspection</td>
<td></td>
</tr>
<tr>
<td>SS-19709</td>
<td>(U) The SYSTEM shall collect and maintain metrics on the number, size and type of queries successfully and non-successfully executed by the system: Number, size and type of queries successfully and non-successfully executed; totals by day/month; average daily numbers; max/min number, max number simultaneously run.</td>
<td>3.1.1.6, (U) Infrastructure Test Objectives</td>
<td>Inspection</td>
<td></td>
</tr>
</tbody>
</table>
Lesson Learned 5: Transparency

- Regular engagement with the users and customer is imperative
- Ongoing collaboration and transparency are critical for mission success
Final Notes

• Requirements Analysis and Design
  – Upfront requirement analysis is done during release planning for that release
  – Uses cases are a tool and provide the greater context for user stories
  – Architecture and initial design are first developed during iteration/release 0
  – Every iteration the team determines if a design review is needed

• Requirements and User Stories
  – High level requirements are decomposed into a set of user stories
  – User stories convey understanding (user, need, why)
  – User stories create the Product Backlog

• Validation and Verification
  – High level requirements have tests and each user story has tests.
  – Validation begins with the release planning phase and occurs again during the iteration boundaries
  – Each story has acceptance criteria that is defined before the release/iteration begins
  – Requirements Traceability/Verification Matrix is updated each iteration
Checkpoint: Our User Story

• User Story
  – As a Systems Engineer I want to have a better understanding as to how Systems Engineering practices align with the Agile framework so I have insight as to how this might be implemented in an Agile environment.

• Acceptance Criteria
  – Communicate how requirements analysis and design occur when using an Agile approach
  – Discuss considerations regarding architecture
  – Understanding of the relationship of requirements to user stories and why user stories
  – Insight into the Agile framework and how it helps manage changing requirements and priorities
  – Identification of verification and validation practices within the context of the Agile framework
# References and Recommended Readings

## Agile Requirements and Collaboration

<table>
<thead>
<tr>
<th>Requirements by Collaboration</th>
<th>Ellen Gottesdiener, EBG Consulting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaboration Explained</td>
<td>Jean Tabaka, Rally Software</td>
</tr>
<tr>
<td>User Stories Applied</td>
<td>Mike Cohn</td>
</tr>
</tbody>
</table>

## Agile Development Practices

<table>
<thead>
<tr>
<th>Agile Software Requirements</th>
<th>Dean Leffingwell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile Software Development with Scrum</td>
<td>Ken Schwaber and Mike Beedle</td>
</tr>
<tr>
<td>Agile Testing</td>
<td>Lisa Crispin and Janet Gregory</td>
</tr>
<tr>
<td>Agile Estimating and Planning</td>
<td>Mike Cohn</td>
</tr>
<tr>
<td>Agile Modeling</td>
<td>Scott Ambler</td>
</tr>
</tbody>
</table>
The INCOSE SE Handbook V3.2.1 states: “INCOSE use: Permission to reproduce and use this document or parts thereof by members of INCOSE and to prepare derivative works from this document for INCOSE use is granted, with attribution to INCOSE and the original author(s) where practical, provided this copyright notice is included with all reproductions and derivative works. Content from ISO/IEC 15288:2002 and 15288:2008 is used by permission, and is not to be reproduced other than as part of this total document.”

THE VALUE OF PERFORMANCE.

NORTHROP GRUMMAN