Agile Systems Engineering
A Case Study

Author:
Warren B. Smith
Systems Engineering Partner
(480) 560-2655
wsmith@gatech.edu
wsmith@wrayn.com

What, When, Why

- Upgrade a major Army vehicle system-of-record
  - Multiple Variants
- Mission: Get to System Requirements Review (SRR)
  - Over 50 System Analyses, less than 5 Months
- Author’s company brought in as an “Engineering SWAT Team”
- Structured intensive analysis approach
  - MBSE-based development
  - Multiple Teams

How to accomplish this???

- Effectively derive and communicate requirements
- Quick buy-in of requirements
- Diverse stakeholders
  - Customer
  - End-users
  - IPTs
  - Logistics

Project Objectives

- Starting point: List of new system capabilities
- Develop: A comprehensive set of System and Subsystem Requirements
- Ensure buy-in from all stakeholders
- Rhapsody-based analysis of capabilities
  - Use graphical modeling techniques

...do the impossible...
...time-critical work...

Logistical Planning

- The approach was organized and formalized
  - SCHEDULING!
- Multiple teams stood up
  - Facilitation/Leaders trained
- IPT Leads and Customer read-into the plan
- End-Users identified from the workforce

What is “All at Once”?

- Team-based analyses of Operational Scenarios
  - All stakeholders in the same room
  - Simultaneous concurrence
  - Live development of work products
- Short-turnaround time for each analysis
  - Daily meetings
  - Formalize work products, post-meeting

...do the impossible...
...time-critical work...

Repeat for all analyses
Getting Started...

Phase 1: Develop User Needs

Develop current & desired
use of the system

User Needs

- Developed list of scenarios (Backlog)
- Assigned each to a Team Leader (Scrum Master)
- Bring users with expertise into the meetings

Daily Meetings (Scrum Process)

- Scrum Master Lead:
  - Facilitates meetings
  - Cleans up model, integrates with other work
  - Ensures interface consistency
  - Publish results
- Development Team
  - Write validation methods
- At next meeting:
  - Review cleaned up work
  - Press ahead

Daily Meetings - Systems analysis

1. Prioritize capability list (Backlog)
2. Post source requirements
3. Model primary thread using Activity Diagrams to satisfy mission requirements
   - Expect a lot of discussion!
4. Notes placed directly on the diagram.
   - Constraints / scenarios / etc.
   - System-level requirements written for each activity

Between Daily Meetings...

- Systems Engineer cleans up models
  - Formulate good requirements
  - Clarify diagrams

Work Tempo
CONCURRENCE
End Users  Customers  IPTs

Work Tempo

Work Products

- System Level Requirements...
  ...based on Analysis

- Buy-in from all stakeholders

- Key factors:
  - Management backing effort
  - Even handed, calm facilitation
  - Thorough preparation for the next day’s meeting

Phase 2: Develop Subsystem Requirements
Derive and Allocate subsystem requirements

Scrum Process - Daily Sprint Meetings

- Subsystem Analysis

1. Decompose system behavior to subsystems
2. Use allocation (Sequence Diagrams)
3. Define / negotiate interfaces
4. Write functional descriptions/derived rqmts for each operation
5. Allocate behavior to subsystems
  - Operations, Functions or States

Work Tempo

- Different set of participants
- Significant IPT negotiation
- Include Test Engineering
  - Identify Verification Methods

Requirements derived as part of analysis
Scrum Process Between Subsystem meetings

- Scrum Master Lead:
  - Cleans up model, integrates layer
  - Simulates thread, documents performance
    - Recommends performance requirements at next Scrum meeting
    - Ensures interface consistency
  - Publish results
- Development Team
  - Write their subsystem requirements
  - Write verification methods
- At next meeting:
  - Review cleaned up work
  - Repeat with next capability

Upon completion of each capability

- Integrate thread with previous work
  - Ensure self-consistent, complete model
- Backfill System analysis and requirements

At end of sprint

- Publish integrated Model
- Publish subsystem specifications
  - Functional, Performance, Interface, Constraints
- Publish views
  - Threads analyses
  - Requirements
  - Sequence diagrams

Why is this Agile?

Defining Agile Development

Agile Software Development

Agile Manifesto
- Agile Manifesto http://agilemanifesto.org/
- Many methods and techniques
  - Scrum
  - Feature Driven Development
  - Extreme Programming
- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

Agile Software Development

- Scrum Development
  - Prioritized Requirements Backlog
  - Sprints – 3-6 weeks
    - Daily Standups to assess status
  - Update system during sprint
    - Design software
    - Implement changes
    - Test
    - Deploy / release
Agile Systems Engineering

- **A scrum** is organized into 2-6 week sprints
  - Iterative and Incremental Development
- Characterized by an incremental development against a list of capabilities (**Backlog**)
- **Secret sauce**: Agile SE develops a set of systems engineering work products during each sprint

SE Agile Scrum Structure

- **Daily meetings**
  - 1 hour maximum
  - Same time each day. Start on time
  - MBSE tool / Projector
- **Interdisciplinary team**
  - Scrum Master (Systems lead and/or facilitator)
  - Tool Jockey
  - Facilitator
  - Product Owner (User)
  - Development Team (Subsystem and Test personnel)
  - Other Stakeholders
  - Users, Customers, Vendors
  - Managers

Key elements relating to Systems

- **Customer focus**
  - Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
  - Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.
- **People**
  - Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
  - The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- **Teams**
  - The best architectures, requirements, and designs emerge from self-organizing teams.
  - At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly

Benefits of Agile SE

**Management Benefits**

- Greatly reduced development times
- Reduced re-work

**Technical Benefits**

- Assures Completeness and Consistency of Analysis
- Quicker, higher quality development
- Fewer technical mistakes
- Traces and Documents Engineered Product Artifacts
- Specifications reflect collective knowledge
- Technique can be applied to other SE areas

Faster, Higher Quality, Less Expensive

Agile Systems Engineering

*A case study*

Warren B. Smith
wsmith@wrayn.com
wsmith@gatech.edu
(480) 560-2655

Agile Systems Engineering

A case study