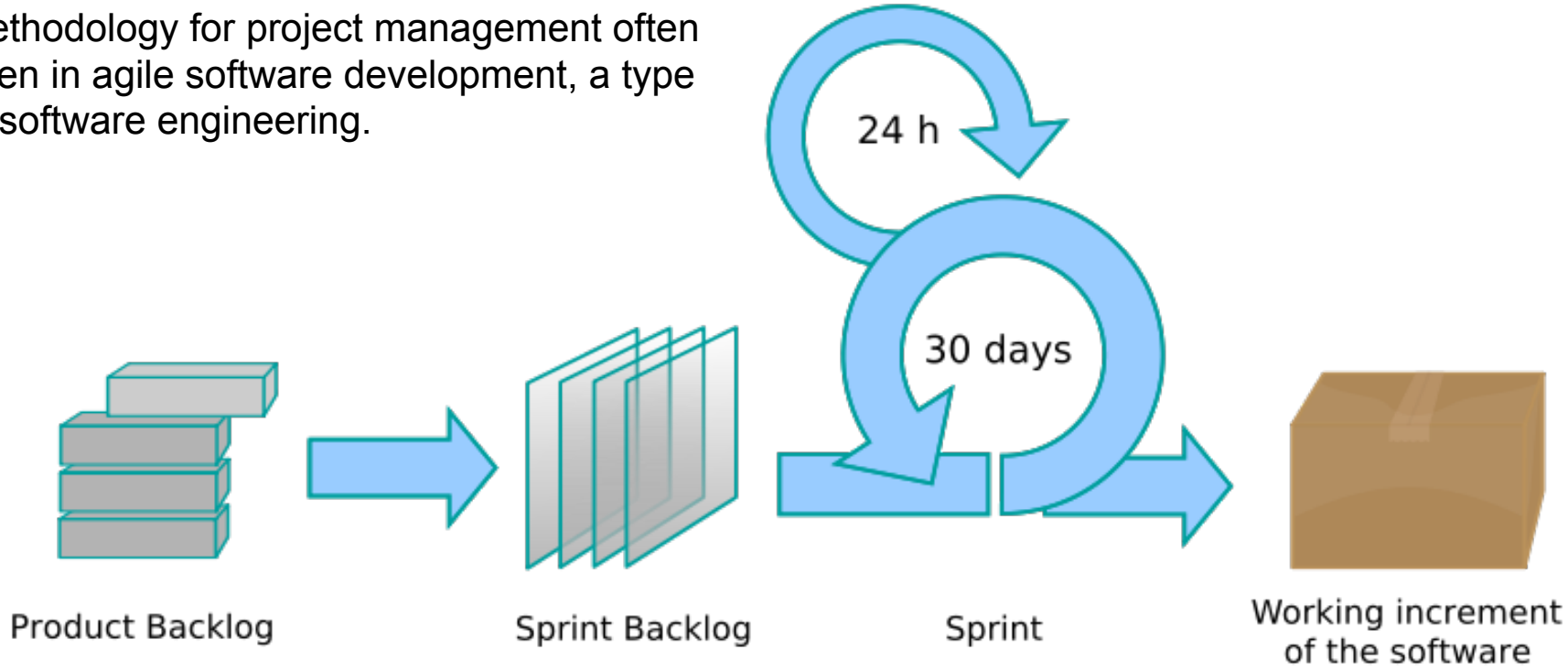
INCOSE International Symposium
Las Vegas, NV
June 30 - July 3, 2014

Systems Engineering for Software Intensive Projects Using Agile Methods

David Lempia, Rockwell Collins
Phyllis Marbach, Boeing
Gundars Oswalds,
Larri Rosser, Raytheon

Introduction to Agile (Scrum)

Scrum is an iterative, incremental methodology for project management often seen in agile software development, a type of software engineering.

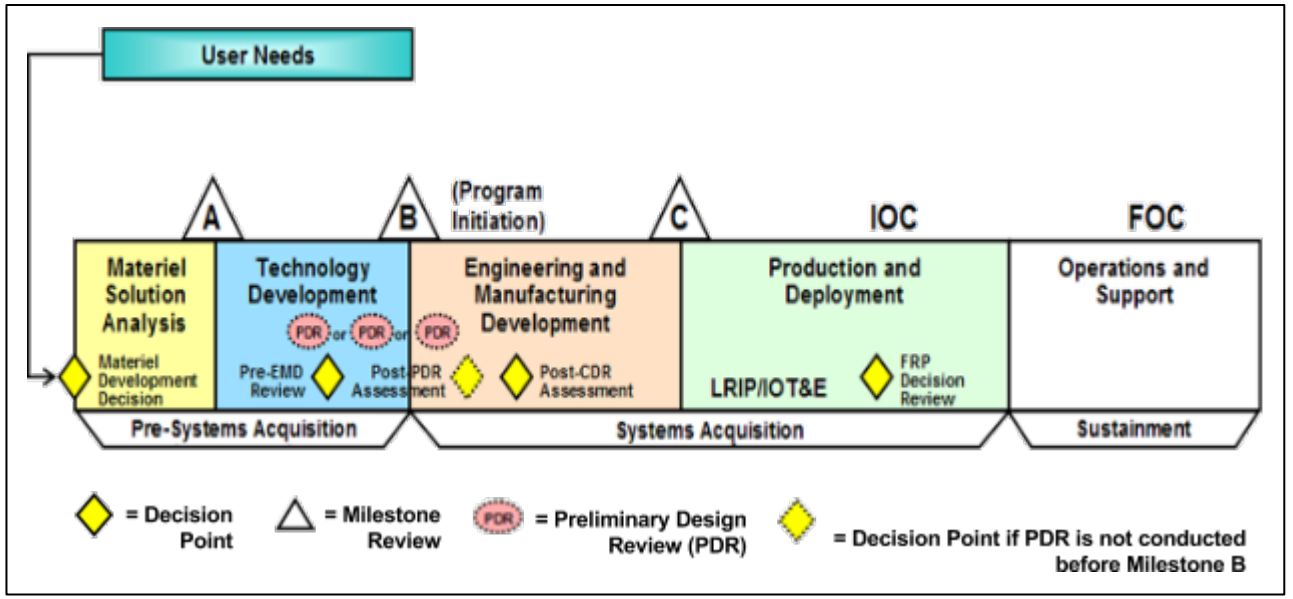


Systems Engineering

- An interdisciplinary approach and means to enable the realization of successful systems (INCOSE handbook)
- Many SE activities:
  - Technical Management
  - Mission and needs analysis
  - Requirements articulation and management
  - System architecture and design
  - Technical analysis and trades
- SE works with the customers and program office
- This paper's focus is on the role of SE in supporting implementation
- Technical processes addressed:
  - Stakeholder requirements definition
  - Requirements analysis
  - Architectural design
  - Implementation
  - Integration
  - Verification

System Acquisition Framework

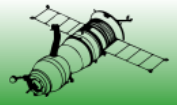
- Defense Acquisition Guidebook
- Focus on Engineering Manufacturing Development (EMD) Phase
- Requirements are defined at Milestone B
- Traditionally:
  - First engineers define and interpret stakeholder needs
  - Second SE develops the system design or architecture framework
  - Third software engineers develop detailed designs
  - Fourth SWE implements the capabilities



Used with permission from DAU

Agile SE Framework

- Changes to the architecture – modular and evolving
- Changes to the process – iterative, incremental
- Changes to the roles
  - SE become members of the implementation teams;
  - SE staffing remains more level throughout the development to support and maintain the architecture, requirements, testing, verification, artifact development, etc.

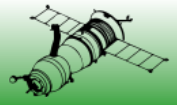




SE Architect Role

- SE identify and analyze architecture dependencies
- Create and continuously update an architecture description
- Participate with the SE Team (SEIT, Architecture, etc.)
- Participate on one or more Implementation Teams
- Work one iteration ahead of the developers

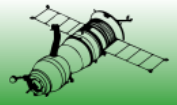
SE Architect Role Differences

- Flexible, modular architecture framework; rather than having Big Design Up Front (BDUF)
- Detailed design/architecture is implemented in each iteration providing technical and user evaluations often manages technical risk and enables user validation that the solution meets their expectations; rather than waiting until the end of a long development period for this verification of the technical solution and validation from the user
- Quality attributes of the architecture are built in from the beginning and shown to be met each iteration for that part of the development that was just completed; rather than showing traceability between tests and quality attributes at the end of a long development period
- The architecture is adjusted and modified as needed; rather than assuming the architecture and design is fixed and never changing because that phase of development is in the past.


SE Process

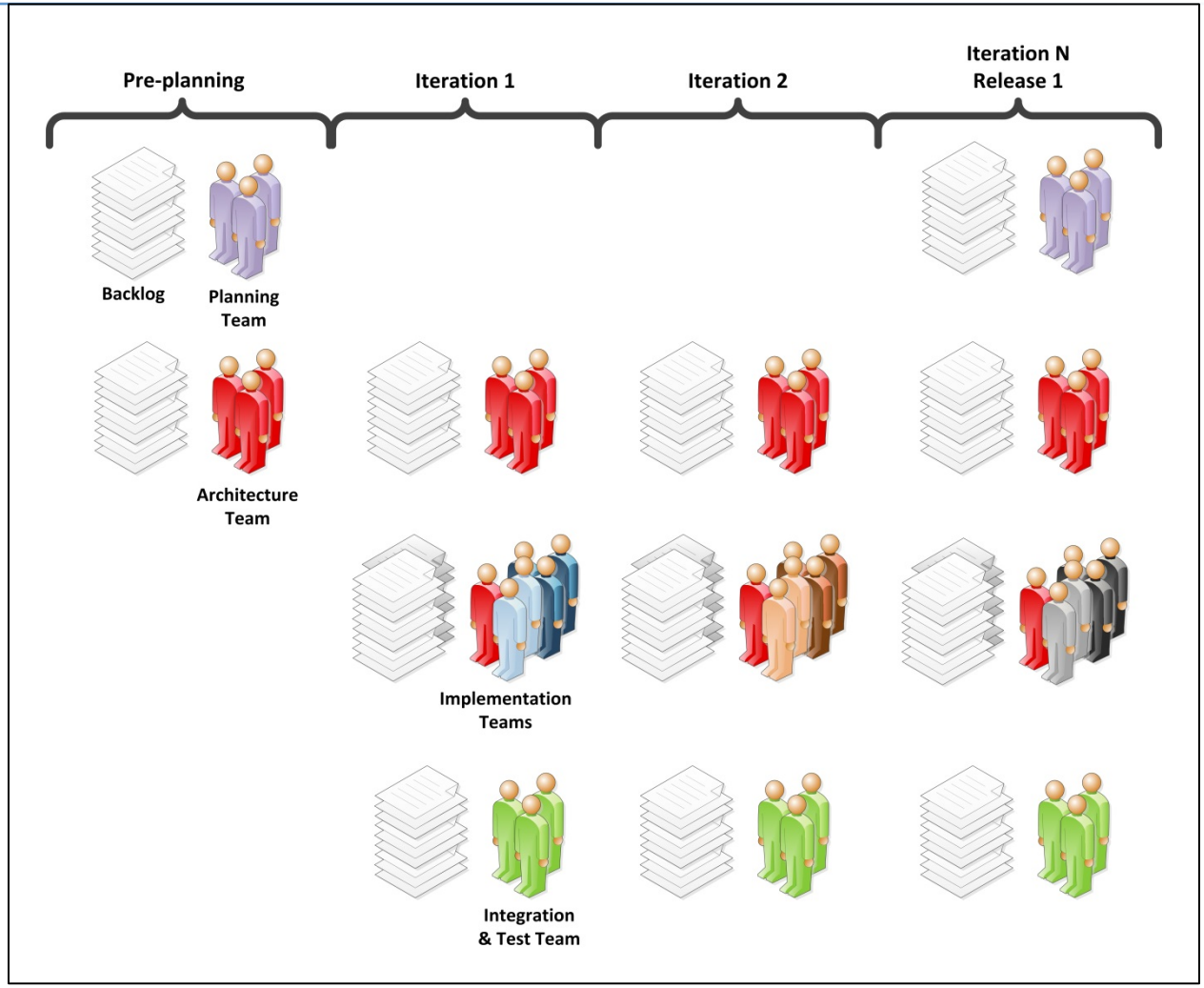
- SE and SWE work together to:
  - Define capabilities
  - Implement capabilities
  - Test capabilities
  - Inspect the results
  - Adapt capabilities as needed
  - Maintain system integrity
- Larger programs with several teams working in parallel need SE engaged
- Each aspect of development (requirements, design, implementation, test, verification) is continually revisited throughout the development lifecycle

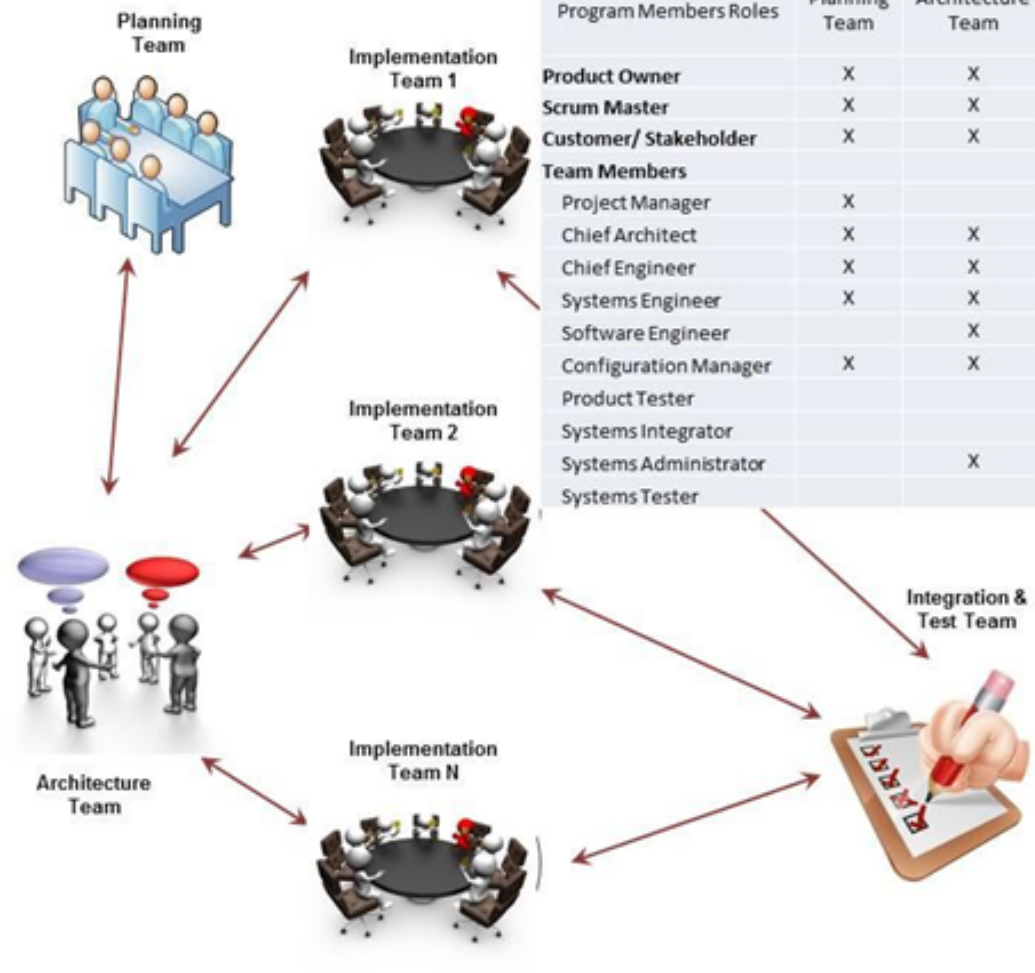

SE Process Differences

- SE and SWE work together throughout the iterations; rather than having SE define the capabilities and provides the system architecture to SWE and then go away until the software is done. Then SE starts integration and verification.
- Larger programs with several teams working in parallel need SE engaged; rather than having few SE available during implementation to answer questions.
- Each aspect of development is continually revisited throughout the development lifecycle; rather than having the requirements and design formally baselined even for areas of high risk or technical uncertainty

Agile SE Framework



Agile Teams

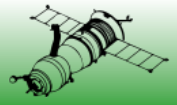


Program Members Roles	Planning Team	Architecture Team	Implementation Team(s)	Integration & Test Team
Product Owner	X	X	X	X
Scrum Master	X	X	X	X
Customer/ Stakeholder	X	X	X	X
Team Members				
Project Manager	X			
Chief Architect	X	X		
Chief Engineer	X	X	X	X
Systems Engineer	X	X	X	X
Software Engineer		X	X	
Configuration Manager	X	X	X	X
Product Tester			X	
Systems Integrator				X
Systems Administrator		X	X	X
Systems Tester			X	X

Example Planning Team RACI

PLANNING TEAM				
ROLES	Product Owner	Scrum Master	Team Member	Customer/ Stakeholder
Scope	C	C	R	A
Define Deliverables (Product Level)	C	C	R	A
Technical Management Mission	A	C	R	C
Needs Analysis	A	C	R	C
Requirement Articulation (Product Capability Backlog)	R	C	C	A
Requirements Management	A	C	R	C
Meeting Facilitator/ Impediment Remover	A	R	C	C

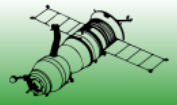
RACI Matrix Legend	
Responsible (one)	Leads the task completion with tangible deliverables
Accountable (one)	Delagated the responsibility for task, approves completion
Consulted (many)	Multiple contributors provide special knowledge or expertise
Informed (many)	Members that will be informed of the task status and deliverables


Example Arch Team RACI

ARCHITECTURE TEAM				
ROLES	Product Owner	Scrum Master	Team Member	Customer/ Stakeholder
Vision	A	C	R	C
Roadmap	A	C	R	C
Architecture Framework/ System Design	C	C	R	A
Define and Maintain Interfaces	A	C	R	C
Architecture Product Backlog	A	C	R	C
Concept of Operations (CONOP)	C	C	R	A
Perform Trade Studies	A	C	R	C
Meeting Facilitator/ Impediment Remover	A	R	C	C

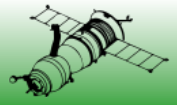
RACI Matrix Legend	
Responsible (one)	Leads the task completion with tangible deliverables
Accountable (one)	Delagated the responsibility for task, approves completion
Consulted (many)	Multiple contributors provide special knowledge or expertise
Informed (many)	Members that will be informed of the task status and deliverables


Example Implementation RACI

IMPLEMENTATION TEAM				
ROLES	Product Owner	Scrum Master	Team Member	Customer/ Stakeholder
Develop/ Maintain Software Design (Detailed Design	A	C	R	C
Software Implementation	A	C	R	C
Integration - unit test, SW integration as possible	A	C	R	C
Verification	A	C	R	C
Maintain/ Verify System Capabilities as possible	A	C	R	C
Maintain Interface Definitions of SW/ Component	A	C	R	C
Perform Trade Studies	A	C	R	C
Develop/ Maintain Test Procedures	A	C	R	C
Meeting Facilitator/ Impediment Remover	A	R	C	C

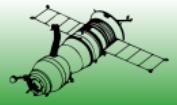
RACI Matrix Legend	
Responsible (one)	Leads the task completion with tangible deliverables
Accountable (one)	Delagated the responsibility for task, approves completion
Consulted (many)	Multiple contributors provide special knowledge or expertise
Informed (many)	Members that will be informed of the task status and deliverables


Example I&T Team RACI

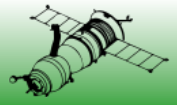
INTEGRATION AND TEST TEAM				
ROLES	Product Owner	Scrum Master	Team Member	Customer/ Stakeholder
Software Backup/ SW Baseline Test Baseline	A	C	R	I
System Integration	A	C	R	I
Validation	A	C	R	C
Meeting Facilitator/ Impediment Remover	A	R	C	C

RACI Matrix Legend	
Responsible (one)	Leads the task completion with tangible deliverables
Accountable (one)	Delagated the responsibility for task, approves completion
Consulted (many)	Multiple contributors provide special knowledge or expertise
Informed (many)	Members that will be informed of the task status and deliverables


Challenges from Traditional SE with Agile SWE

- Lack of Rapid Response
- Big Design Up Front
- Architecture Interpretation
- Non-Functional Requirements (NFR)
- Responding to Change at Scale
- Verification, Validation and Test

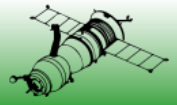




Lack of Rapid Response

- Challenge:
  - When systems engineering activities are performed in isolation from software development teams,
  - Important systems engineering activities are not informed by or responsive to findings from the software development team:
    - Definition of key performance parameters
    - Definition of testing scenarios
    - Definition of architecture principals
    - Risk analysis
    - Technical trade studies
- Enabler from the Agile SE Framework:
  - Continual Interfacing on cross-functional teams consisting of SE, SWE, and Testers co-develop one story/capability from concept through completed customer acceptance testing during an iteration

Big Design Up Front

- Challenge:
  - When systems engineering activities are performed on a traditional schedule it is assumed that development will not begin until the Big Design Up Front (BDUF) is released.
  - If the SE is “not finished” implementation is delayed or the software team may start to develop detailed design and code with no input from SE.
- Enabler from the Agile SE Framework:
  - Create a roadmap of capabilities to implement over time.
  - From that roadmap create a prioritized backlog.
  - Break down the capabilities until each high priority backlog item is sized so that it can be implemented in one iteration.
  - Iterative planning allows the Implementation Team to start into development of the detailed design and coding with input from the SE (who is on the Architecture Team).


Architecture Interpretation

- Challenge:
  - SE, as part of the Architecture Team, develops a detailed and comprehensive architecture and passes it over to the Implementation Team.
  - The risk is that Software implementation opportunities and constraints are not adequately considered in systems engineering thus limiting flexibility.
- Enabler from the Agile SE Framework:
  - Architecture modularity and an iterative process requires architecture design effort throughout the development lifecycle.
  - For large teams the integrity of the architecture needs to be maintained as the development proceeds.
  - A modular framework is sufficient to begin development.
  - Architectural tasks may be planned into releases to be worked over several iterations as needed.

Non-Functional Requirements

- Challenge:
  - The agile paradigm addresses functional requirements as backlog items or user stories.
  - However, common agile practices do not directly address non-functional requirements.
  - When quality attributes (i.e., “ilities” — reliability, speed, usability, flexibility, etc.) are not analyzed and tracked through design and implementation then the system may not perform as desired and confidence in the system’s ability to perform as desired may be limited.
- Enabler from the Agile SE Framework:
  - Quality attributes are planned into each iterative development user story when a team plans and performs work on agile cross-functional Implementation Teams as described in the Agile SE Framework.

Responding to Change at Scale

- Challenge:
  - When agile software development methods, used successfully on small projects, are applied to a very large effort, the processes fail to scale and SE activities and products are not effectively used in implementation.
  - Requirements may be interpreted differently by different Implementation Teams, architectural principles may not be universally applied, and interface definitions may develop gaps and overlaps.
- Enablers from the Agile SE Framework:
  - Larger teams need a team to integrate and test the products produced by the Implementation Teams. This is the I&T team shown in the Agile SE Framework and the Agile Teams figures
  - The Planning Team, with SE team members, maintains the requirements and capabilities
  - The Architecture Team, with SE team members, maintains the architecture integrity

Verification, Validation and Test

- Challenge:
  - Traditional SE practice for “pull” programs assumes that sell-off is based on Verification of compliance with requirements, not stakeholder (customer) satisfaction with deliverable functions.
  - This requires Validation that capabilities satisfy stakeholder needs.
  - Late Validation can result in customer dissatisfaction that must be dealt with late in the program, when modification is most expensive.
- Enablers from the Agile SE Framework:
  - Leverage the Agile software development practice of continuous integration
  - Create a situation in which stories are demonstrated, tested and even accepted as early as possible in the development cycle.
  - Share the testing artifacts with the customer to ensure a common understanding of the functionality to be developed.
  - Strive to automate testing when each function, feature, and feature set is submitted.

Agile SE Framework Conclusion

- SE and SWE work together to develop and evolve the work products iteratively
- Define “just enough” architecture and requirements prior to the beginning of implementation
- Release Planning and Iteration Planning are essential to detail the work and coordinate the teams
- Release products frequently
- Absorb changes to mission requirements
- Include requirements, architecture, system design and validation by SE on large scale agile projects

References

ADAPT 2013. *Achieving Better Buying Power 2.0 For Software Acquisition: Agile Methods.* The Agile Defense Adaption Proponents Group of The Association for Enterprise Information. http://www.afei.org/WorkingGroups/ADAPT/Pages/default.aspx

Brown et al. 2010. Brown Nanette, Nord Robert, Ozkaya Ipek. *Enabling Agility Through Architecture.* CrossTalk. http://www.sei.cmu.edu/library/assets/whitepapers/brown-nord-ozkaya-crosstalk-Nov10.pdf

DoD. 2010. *Better Buying Power.* Department of Defense. http://www.acq.osd.mil/docs/USD_ATL_Guidance_Memo_September_14_2010_FINAL.PDF

DoD. 2012. *Defense Acquisition Guidebook.* Department of Defense. https://acc.dau.mil/CommunityBrowser.aspx?id=289207&lang=en-US

Frank M. 2000. *Cognitive and Personality Characteristics of Successful Systems Engineering.* INCOSE International Symposium Proceedings. https://www.incose.org/ipub/00/contents/s_1_6/163_101.pdf

Honour, Eric. C. 2004. *Understanding the Value of Systems Engineering.* INCOSE International Symposium. http://www.incose.org/secoe/0103/ValueSE-INCOSE04.pdf

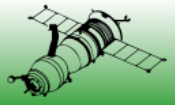
INCOSE. 2011. *INCOSE Systems Engineering Handbook.* International Council on Systems Engineering, v3.2.2. http://www.incose.org/ProductsPubs/products/sehandbook.aspx

ISO/IEC. 2008. *15288 Systems and software engineering — System life cycle processes*. ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission). http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43564

Leffingwell, Dean. 2011. *Agile Software Requirements, Lean Requirements Practices for Teams, Programs, and the Enterprise.* Pearson Education, Inc., Boston, MA. http://deanleffingwell.com/book-agile-software-requirements/

Standish Group. 1994. *The Chaos Report.* The Standish Group International. http://www.csus.edu/indiv/v/velianitis/161/ChaosReport.pdf

Standish Group. 2013. *The Chaos Manifesto.* The Standish Group International. http://versionone.com/assets/img/files/ChaosManifesto2013.pdf


Authors

Larri Rosser, Raytheon
Garland, TX
Larri.rosser@incose.org

Gundars Osvalds, Praxis Engrg
Annapolis Junction, MD
Gundars.osvalds@incose.org

Phyllis Marbach, Boeing
Huntington Beach, CA
Phyllis.marbach@incose.org

David Lempia, Rockwell Collins
Cedar Rapids, IA
David.lempia@incose.org