Principles for Agile Development

Conference Paper *in* INCOSE International Symposium · October 2015

DOI: 10.1002/j.2334-5837.2015.00079.x

4 authors, including:

Larri Rosser
Raytheon Technologies
18 PUBLICATIONS **68** CITATIONS

Principles for Agile Development

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

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

Gundars Osvalds, CSEP
Praxis Engineering
Annapolis Junction, MD
gundars.osvalds@incose.org

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

Abstract. The Manifesto for Agile Software Development [Beck 2001] and the Principles behind the Agile Manifesto were written specifically for software development teams. Since the manifesto does not appear to apply to systems or large programs with multiple teams of development, how it applies to systems engineers is not apparent. The Principles for Agile Development are recommended for cross-functional teams that include Systems and Software Engineers working on customer "pull" projects to produce software products. These principles provide a foundation for the working relationship across the teams. We believe these principles apply to programs that include both hardware development and software development; however, we limit our examples to software intensive development for this discussion. As we strive to produce products more affordably in this ever-increasing, global marketplace adopting these principles will serve as a foundation for working together to develop high-value capabilities incrementally.

Introduction

One definition of principle is "a fundamental truth or proposition that serves as the foundation for a system of belief or behavior or for a chain of reasoning". This paper proposes principles for agile development with examples of the ways that systems engineers contribute to their fulfillment. The concepts introduced in the paper "Systems Engineering for Software Intensive Programs Using Agile Methods" [Rosser 2014] hereafter referred to as the Agile SE Framework paper are used to elaborate on the Principles for Agile Development. The Agile SE Framework is an approach to integrating systems engineering practices into a software intensive program that is using agile methods for a typical DoD EMD phase program. The described

architecture, process and role changes enable a smooth working cadence to develop a product through iterative development using scrum, continuous integration and automated testing. Due to size constraints the Agile SE Framework paper did not include the principles that lay the foundation for the Agile SE Framework. This paper summarizes the Agile SE Framework and then provides more detail about the twelve agile software principles now tailored for more general development. Adopting these principles will serve as a foundation for SE working together with other team members to develop high-value capabilities more rapidly.

"The value realized in Scrum and Agile are practices that place humans not processes or techniques, at the center of an organization" [Kim 2013]. Making these Principles of Agile Development the foundation of an organization and the culture, will result in long-term success and a happier workforce. Humans are happier when they have meaningful work and some even measure that happiness at the end of each iteration as described by Scrum co-creator Jeff Sutherland [ScrumInc 2014].

Agile SE Framework

The Agile SE Framework proposes methods for cross-functional teams that include Systems and Software engineers working on projects with customer specified requirements to produce products that may include software or be mostly software. The proposed Agile SE Framework aligns with an agile software development framework, and describes the role of the Systems Engineer (SE) in this context. It presents an iterative approach to the aspects of development (requirements, design, etc.) that are relevant to systems engineering practice. This approach delivers frequent releasable products that result in the ability to absorb changes in mission requirements through collaboration between systems engineers and software engineers. The Agile SE Framework defines a way to scale agile from individual agile software teams with a few members to large projects that require a planned architecture and coordinated efforts.

The Agile SE Framework describes changes to the architecture, process, and roles required to move from traditional SE processes to an SE process that augments the agile software development teams. An issue with current agile methodologies is that the system architecture is not an explicit part of the agile software development methodology. When developing small software intensive systems the architecture design is the responsibility of the agile software development team. The architecture design emerges each time the software is refactored. This emergent design results in architecture uncertainty. For small systems, the refactoring cost is small. For large systems, refactoring resulting in architecture changes can ripple through multiple teams. For these larger systems there needs to be consideration for dependencies between the system capabilities and architectural elements developed by different implementation teams. In the Agile SE Framework it is the SEs' responsibility to balance the lessons learned from the implementation teams with new capabilities and quality attributes that the architecture must support. The Architecture team members, usually SEs, define and continuously update an architecture framework to support the software implementation. The Architecture Team stays

just ahead of the Implementation Team, incorporating lessons learned from the previous iteration as input to refactor and refine the architecture, followed by developing new SE artifacts to support the next iteration. This concept is shown in Figure 1, a figure that was originally presented in [Rosser 2014].

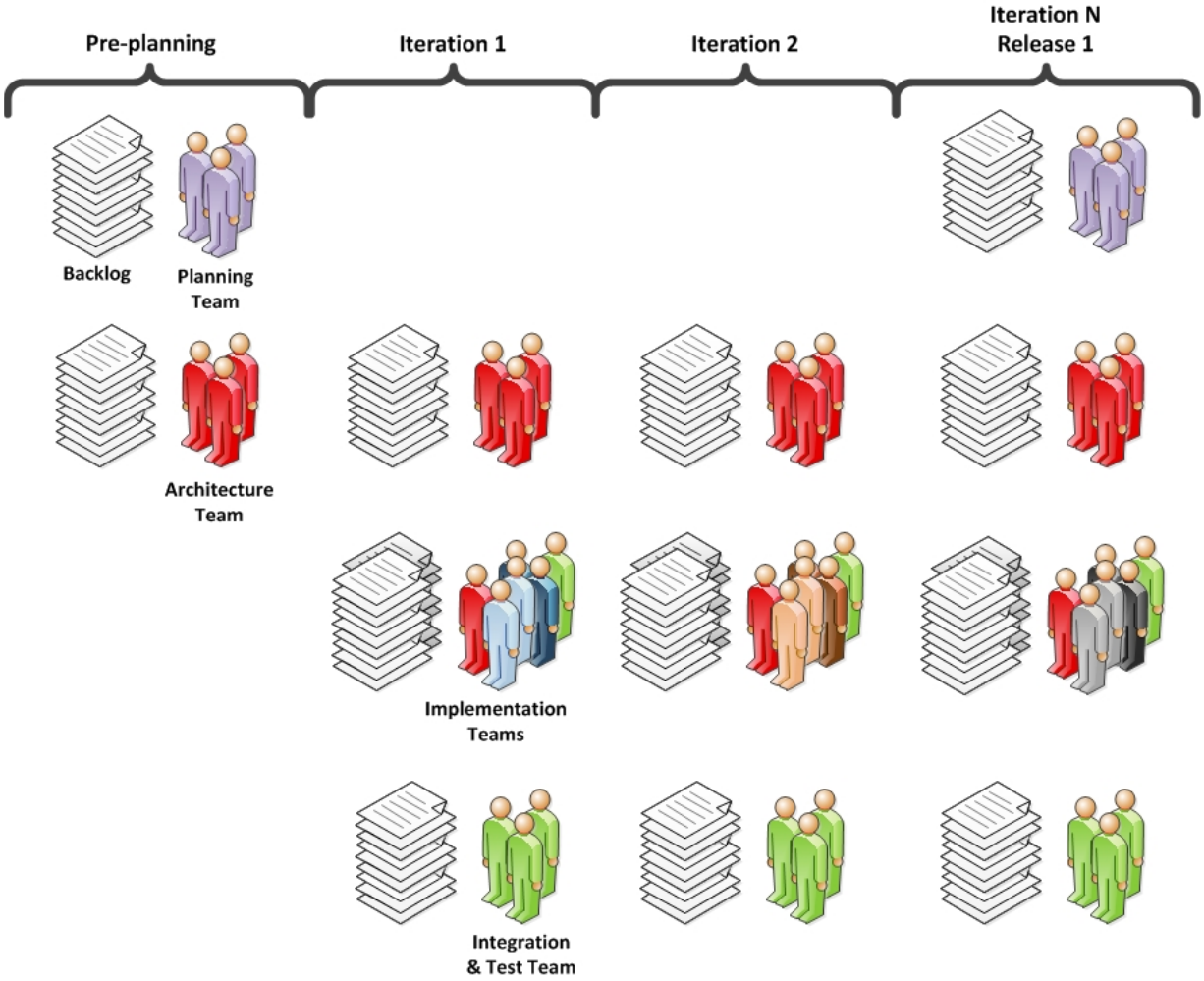

Figure 1. Agile SE Framework [Rosser, 2014]

The process flow of the teams starts with the planning and the architecture teams defining the capabilities for the system to be developed. The pre-planning period includes the technical planning, mission and needs analysis, requirements articulation, requirements management definition, and architecture framework creation. Input into this pre-planning step includes customer needed capabilities and the output is a vision, roadmap, architecture framework, and a prioritized backlog of significant capabilities to be developed. During the pre-planning phase the Planning Team defines the scope and deliverables of the project, while the Architecture Team, establishes the vision, the roadmap, architecture, and a product backlog. When working with agile software development teams, the level of detail of the design artifacts needed to start the first implementation iteration may be less than what is normally produced on traditional life cycle projects since the product owner will be involved by providing feedback during the

program development cycle. Some elements of the architecture or requirements may be identified for analysis and elaboration later in the implementation cycle. Depending on the level of formality of the project, the planning stage outputs could include a Concept of Operations Document (CONOP), planning artifacts, architecture diagrams and models, and a high level list of requirements. The outputs from the pre-planning phase flows into the first iteration where product development begins, artifacts are produced and maintained and interim releases are demonstrated. Feedback from those interim releases is implemented into the next release planning and iterative development continues until the final release is completed. The system engineers develop and maintain products such as requirements specifications, architecture descriptions, test procedures and results iteratively.

The principles or foundation for the belief in this Agile SE Framework are the topic of this paper and will be introduced after a review of the Manifesto for Agile Software Development in the next section.

Manifesto for Agile

In 2001 a group of individuals defined the Manifesto for Agile Software Development, provided here for context. "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."[Beck, 2001] The "Principles behind the Agile Manifesto" are fundamental truths that lay a foundation for a system of organizational behavior. A few modifications to the words in the agile principles are needed to lay the foundation for the Agile SE Framework. The authors found three other proposed Agile Principles, one set of principles in the INCOSE SE Handbook V3.2.2 [INCOSE 2011], another set of principles titled the Disciplined Agile Manifesto on the Disciplines Agile Delivery website by Scott Ambler [Ambler 2014] and the third set of principles introduced in The Agile Manifesto reworked for Systems Engineering by Hazel Woodcock [Woodcock 2013]. Each of these sets of principles changes a few words of the Twelve Principles of Agile Software so that the principles are more applicable to a development effort of a system or component. For example the word software has been changed to "other system elements", solutions, or capabilities. The word customer has been changed to stakeholder, business people has been changed to stakeholder, developer has been changed to project team and development has been changed to delivery. Is there value in establishing a more general set of Principles for Agile Development? We believe there is value in establishing foundational principles and to elaborate on the role of systems engineering on programs that

have more than one agile team developing product. We take the original Principles behind the Agile Manifesto and recommend changes to those principles. The recommended changes are based on the authors' experience with supporting programs with multiple agile teams developing product and anticipating the value that systems engineering has or could bring to the program by working within the implementation teams producing and maintaining the products that systems engineering typically produce. The Principles behind the Agile Manifesto are listed in the next section, along with a brief explanation of how a systems engineer supports that principle and recommended updates to the wording of the principles. Adapting these principles causes us to rethink the way we execute systems engineering activities, while at the same time affirming the critical role of systems engineering in the development of systems.

Principles for Agile Development

Early and Continuous Delivery of Capabilities

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software [Beck, 2001]. *The system engineer contributes to this principle by articulating stakeholder and customer needs into prioritized capabilities and verifying satisfaction of those capabilities as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We recommend the following wording: ***First, satisfy the stakeholder(s) through early and continuous delivery of valuable capabilities.***

Our rationale for this recommendation is that from a systems perspective, value resides in the capability, or ability to achieve some result, regardless of what elements (software, hardware, data, human activities, etc) are used to create it. Valuable capabilities may include mockups, models, demonstrations and prototypes. These type of artifacts benefit incremental discovery for larger and more comprehensive systems. Our vision is for the methods to apply to the entire system, regardless of the implementation medium.

As stated in the Agile SE Framework section the SE contributes to this principle by articulating stakeholder needs into prioritized capabilities and verifying satisfaction of those capabilities as an involved member of both the architecture team and one or more implementation teams. Ideally, each SE member participates in only one implementation team to minimize multi-tasking. One goal is that the SE maintains the architectural integrity as the detailed design evolves and the product is implemented during each iteration. Each iteration the implementation teams develop product capability and perform continuous integration so that the system in development passes systems and stakeholder acceptance tests. This helps minimize the time it takes to go from stakeholder need identification to passed systems or stakeholder acceptance tests.

Evolving Requirements

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. [Beck, 2001] *The system engineer contributes to this principle by identifying and balancing technical risks and cost/schedule drivers with flexibility in implementation as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We recommend the following wording: ***Plan for evolving requirements, and retain as much flexibility as is valuable throughout development especially when change leads to a competitive advantage.***

Our rationale for this recommendation is that complete accommodation of all requirements changes may be inconsistent with key customer concerns around cost and schedule, especially within the current DoD acquisition paradigm.

The SE contributes to this principle by identifying and balancing technical risks and cost/schedule drivers with flexibility in implementation as an involved member of the planning team, architecture team and one or more implementation teams. The requirements will still evolve as trades are completed and implementation decisions are made. Included in the trade analysis will be an assessment of the change impact to testing so that updates are made to automated tests, also. The program depends on the SE to perform the requirements trades, when needed, as the detailed design and implementation proceeds. The overall architectural integrity of the system is being monitored and at times updated by the SE as the requirements evolve.

Deliver Working Capabilities Frequently

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. [Beck, 2001] *The system engineer contributes to this principle by planning integration, testing, demonstration and (if required) delivery of capability increments as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We recommend the following wording: ***Deliver working capabilities frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.***

Our rationale for this, as in the Early and Continuous Delivery of Capabilities section, is that an emphasis on capabilities rather than a consumable solution is most appropriate for a systems engineering perspective especially for a large complex system that may need mockups and prototypes early in the development before a consumable solution is possible.

The SE contributes to this principle by planning the integration, testing, demonstration and (if required) delivery of capability increments as an involved member of both the architecture team and one or more implementation teams. The demonstration, at the end of each iteration, provides working capability that has been tested and traced to requirements by the SE. These frequently delivered working capabilities provide an opportunity for feedback from the customer to help assure that the implementation is on track to meet customer expectations. Adjustments to the work in the next iteration can be made based on that feedback.

Work Together Daily

Business people and developers must work together daily throughout the project. [Beck, 2001] *The system engineer contributes to this principle by orchestrating these interactions for maximum value and minimum disruption and providing a "single unified voice" for clear technical guidance as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We recommend the following wording: ***Business personnel, customers or their advocates, and implementers must work together daily throughout the project.***

Our rationale for this is to include all members of the implementation team, not just developers, and also to identify the stakeholders more specifically. Stakeholders include business people and customers or customer advocates in those situations in which a specific customer contracts for the development of capabilities.

The SE contributes to this principle by orchestrating these interactions for maximum value and minimum disruption and providing a "single unified voice" for clear technical guidance as an involved member of both the architecture team and one or more implementation teams. These daily interactions help reduce the feedback loop that can delay clear understanding of the intended architectural design and system requirements. With daily interactions the developer will perform detailed design and implementation with timely access to SE to support the high-business-value capability in development.

Provide the environment and trust the team

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. [Beck, 2001] *The system engineer contributes to this principle by providing a unifying vision consisting of key architecture and design principles, and by identifying and addressing barriers that require external assistance to overcome as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We accept this principle and its wording as stated. The SE, embedded in the teams, contributes to this principle by providing a unifying vision consisting of key architecture and design principles. The SE is careful to write requirements that do not over-constrain the design and they identify and address barriers that require external assistance to overcome as an involved member of the planning team, architecture team and one or more implementation teams. SE can help engage external assistance from other areas such as support teams that provide cost analysis, stress analysis or power and weight trades.

Personal Conversation

The most efficient and effective method of conveying information to and within a delivery team is face-to-face conversation. [Beck, 2001] *The system engineer contributes to*

this principle by ensuring that requirements, architecture, and testing, signoff and contractual needs are communicated clearly and frequently as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.

We recommend the following wording: ***The most efficient and effective method of conveying information is personal conversation.***

Our rationale for this is that distributed implementation teams have proven to be viable in cases where easy, informal communications are available via chat rooms, discussion forums, instant messaging and video chat.

The SE contributes to this principle by ensuring that requirements, architecture, and testing, signoff and contractual needs are communicated clearly and frequently as an involved member of both the architecture team and one or more implementation teams. Regular clear and frequent interactions between individuals, either face-to-face or over a teleconference call, helps break down barriers and prevent misunderstandings. SE's involvement as a member of both an implementation team and the architecture team will help ensure that requirements, architecture, and other areas of responsibility are included in the prioritized product backlog list. These items are then worked in the order needed to develop a demonstraitable capability each iteration.

Primary Measure of Progress

Working software is the primary measure of progress. [Beck, 2001] *The system engineer contributes to this principle as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We recommend the following wording: ***Working capabilities are the primary measure of progress.***

Our rationale for this, as discussed in the Early and Continuous Delivery of Capabilities section, is to focus on the capability valued by the customer or stakeholder as opposed to the specific developed product or consumable solution because early iterations and releases may be mockups or prototypes.

The SE contributes to this principle as an involved member of both the architecture team and one or more implementation teams. The teams focus and finish the highest priority capability in the current iteration. For large teams the completion of each team's product may flow into an integration effort that depends on each team's capability being completed to accomplish the integration and demonstration. The completion of that integrated high value capability from all of the teams is the measure of progress.

Sustainable Delivery

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. [Beck, 2001] *The system engineer contributes to this principle by coordinating the customers' milestones to the planned releases*

through management of customer expectations versus the team development capacity as an involved member of both the planning team as described in the Agile SE Framework.

We recommend the following wording: ***Agile processes promote sustainable delivery. The sponsors, developers, and users should be engaged and able to maintain a constant pace to the project completion.***

Every member of every team is responsible to estimate and perform their work to enable sustainable delivery iteration after iteration.

Contracting challenges include the conflict between the desire to contract only for the amount of work defined in the scope, versus providing a development capability that can take on additional work via backlog growth, without the loss of velocity that comes with additional procurement cycles. Defining a contract structure for agile capacity based contracting is needed, but not addressed in this paper.

Technical Excellence and Good Design

Continuous attention to technical excellence and good design enhances agility. [Beck, 2001] *The system engineer contributes to this principle by articulating key architectural, design and performance principles and ensuring the implementation of those principles into the system under development as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We accept this principle as stated. The nature of working within both an architecture team and an implementation team provides rigorous scrutiny of the architecture and design. Good design includes a modular architecture that reduces the impact of change by following agile architecture design patterns as described in [Dove, 2014]. This modular, agile architecture enables agile development and maintenance of the system. Verification of the performance requirements for each capability is demonstrated iteratively. This helps mitigate risk and improve customer confidence.

Simplicity

Simplicity "the art of maximizing the amount of work not done" is essential. [Beck, 2001] *The system engineer contributes to this principle by developing an architecture that is resilient to change, by writing requirements that do not over constrain developers, and by continually improving the process as an involved member of the planning team, architecture team, one or more implementation teams, and the integration and test team as described in the Agile SE Framework.*

We recommend adding to this principle with the following wording: ***Simplicity "the art of maximizing the amount of work not done" is essential, especially within the implementation team. A truly agile development project does not force artificial reporting and process requirements on the implementation team.***

Our rationale for this addition is to clarify that simplicity and parsimony apply not only to the system under development but also to the approach to planning, execution and reporting. [Dove-2, 2014] describes the agile systems-engineering process as a discipline based on feedback learning, designed to remove uncertainty and risk about the system to be engineered, and designed to increase productivity of the engineering process. Keeping the work in progress small and building on already demonstrated capability helps the team focus and finish. The team finishes the current planned work that is the next high value capability as defined in the prioritized backlog. Each team member including the SE commits to the work planned for each iteration. At the end of each iteration the entire team supports the retrospection where opportunities for improvement are identified and at least one high value improvement is included into the next iteration. Jan Bosch in [Mirakhorli 2013] stated, "The most useful forms of documentation are views of the software than can be automatically generated." For an SE, one example of simplicity is maintaining artifacts in tools that can automatically generate documentation when needed.

Self-Organizing Teams

The best architectures, requirements, and designs emerge from self-organizing teams. [Beck, 2001] *The system engineer contributes to this principle by defining and evolving the requirements and architecture as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We recommend the following wording: ***The best architectures, requirements, and designs emerge from self-organizing teams, based on a minimal set of guiding principles.***

Our rationale for this is to clarify the need for a core set of organizing principles that ensure key quality attributes of the system. These guiding principles may be modified as the implementation sprints forward, but the current version of these guiding principles must be practiced throughout the system under test. Examples of the guiding principles are modular, agile architecture design patterns are applied, the Agile SE Framework is applied, and quality attributes are designed in from the beginning. For more on agile architecture design patterns see [Dove-1, 2014].

The system engineer contributes to this principle by defining and evolving these guiding principles and the constraints and requirements associated with them as an involved member of both the architecture team and one or more implementation teams. The SE may also provide leadership to the multi-discipline team as a visionary, customer advocate, and process specialist. The SE as a committed member of an implementation and architecture team is involved in selecting the work to be included into the iteration being planned and estimating that work. The cross-functional team members support each other in completing the planned work during the iteration and demonstrating the completed work at the end of each iteration. As innovative architecture improvements are identified in the implementation and testing teams, the SE assesses their impact and if appropriate, includes them in the next architecture update.

Tune and Adjust Team Behavior

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. [Beck, 2001] *The system engineer contributes to this principle by participating fully in the agile practices, including the retrospections, as an involved member of both the architecture team and one or more implementation teams as described in the Agile SE Framework.*

We accept this principle as stated. New team members attend training to learn the principles and practices that agile embraces. All team members, including SE participate fully in the agile practices.

Conclusion

These Principles for Agile Development (shown in Figure 2) have been adapted from the original Principles behind the Agile Manifesto. As we strive to produce products more affordably in this ever-increasing, global marketplace adopting these principles will serve as a foundation for working together to develop high-value capabilities incrementally. This will help the teams focus on the work that has been committed to, commit to the goal, respect and communicate openly with all members of their team, and have the courage to commit, to act, to be open and to expect respect. These five scrum values of Focus, Commitment, Respect, Openness and Courage [Adkins 2010] help the team apply the principles describe in this paper to affordably develop the complex systems of today with the ability to modify and change as development proceeds.

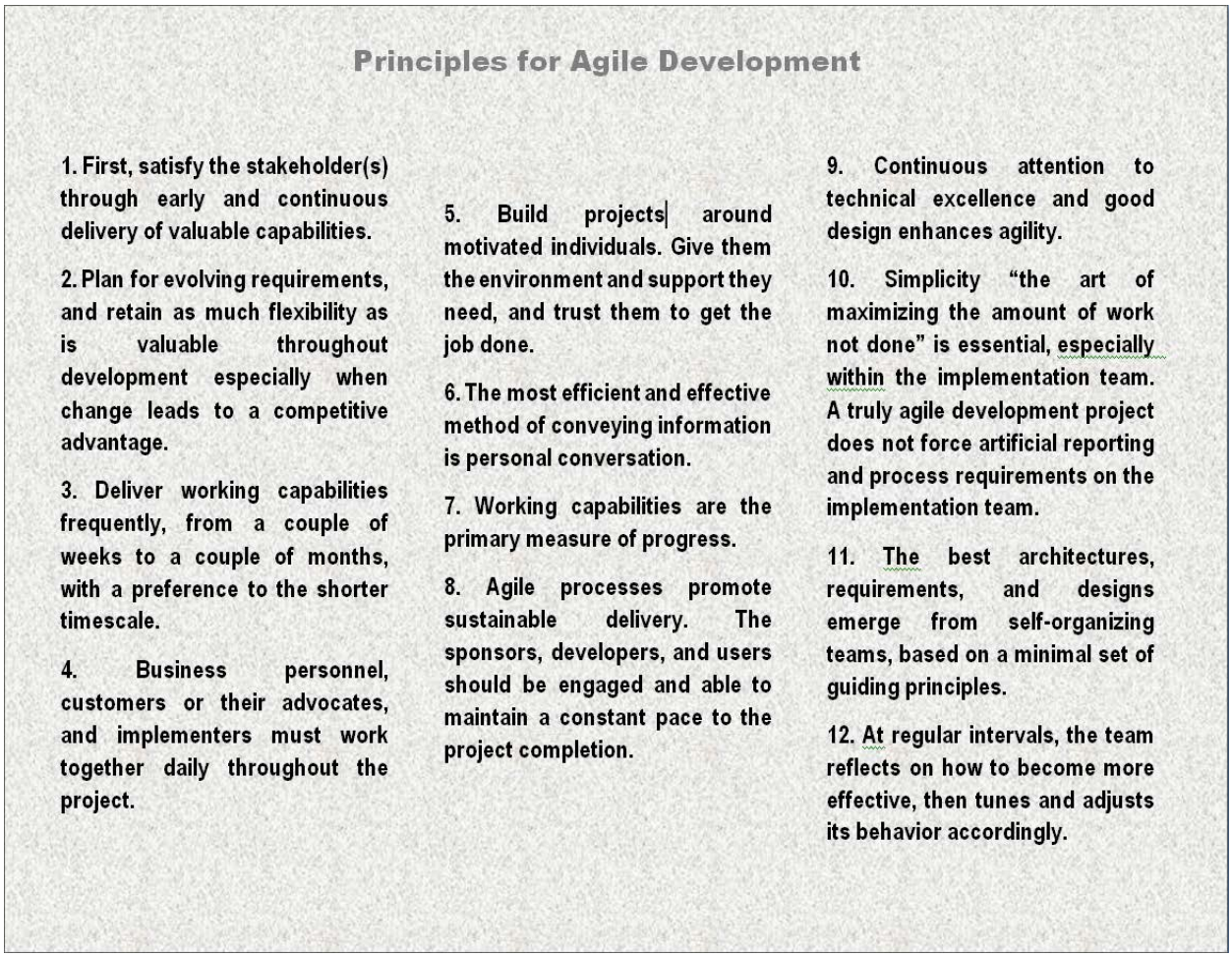

Figure 2. Principles for Agile Development

References

[Adkins, 2010] Book Excerpt: Coaching Agile Teams by Lyssa Adkins, Jun 15, 2010. http://www.infoq.com/articles/adkins-coaching-agile-teams-chapter2

[Ambler 2014] Ambler, Scott April 10, 2014 The Disciplined Agile Manifesto http://disciplinedagiledelivery.wordpress.com/disciplinedagilemanifesto/

[Beck, 2001] Agile Manifesto for Agile Software Development http://agilemanifesto.org/

[Dove-1, 2014] Fundamentals of Agile Systems Engineering – Part 1 Dove, Rick and Ralph LaBarge 2014. International Council on Systems Engineering IS14 Conference, Los Angeles, CA, 30-Jun-03Jul. http://www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part1.pdf

[Dove-2, 2014] Fundamentals of Agile Systems Engineering – Part 2 Dove, Rick and Ralph LaBarge 2014. International Council on Systems Engineering IS14 Conference, Los Angeles, CA, 30-Jun-03Jul. http://www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part2.pdf

[INCOSE, 2011] INCOSE Systems Engineering Handbook v.3.2.2, INCOSE-TP-2003-002-03.2.2 October 2011 Section 3.4.4 Agile Development, Agile Development Principles (adapted for SE) based on "Principles behind the Agile Manifesto,"□15□December□2009, https://connect.incose.org/products/SAWG%20Shared%20Documents/Forms/

[Kim, 2013] The State of Scrum: Benchmarks and Guidelines, Kim, Don, Released June 2013 from ScrumAlliance, https://www.scrumalliance.org/why-scrum/state-of-scrum-report

[Mirakhorli 2013] Mirakhorli, Mehdi; Cleland-Huang, Jane, "Traversing the Twin Peaks," *Software, IEEE* , vol.30, no.2, pp.30,36, March-April 2013 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6470590&isnumber=6470580

[Rosser, 2014] Systems Engineering for Software Intensive Projects Using Agile Methods, Rosser, Larri; Marbach, Phyllis; Osvalds, Gundars; Lempia, David; International Council on Systems Engineering IS14 Conference, Los Angeles, CA, 30Jun-03Jul, 2014.

[ScrumInc, 2014] Happiness Metric, ScrumLab Prime, Jul 21, 2014. http://www.scruminc.com/happiness-metric/

[Woodcock, 2013] The Agile Manifesto reworked for Systems Engineering by Hazel Woodcock, Nov 13, 2013. https://www.ibm.com/developerworks/community/blogs/07f4f478-c082-4196-8489-e83384d85a70/entry/agile_se_manifesto?lang=en referenced on October 12, 2014.

Biography

Phyllis Marbach is a Senior Software Engineer in Boeing's Defense Space & Security (BDS). Phyllis has over 30 years' experience in aerospace programs; including Satellites, chemical lasers, the International Space Station, and various propulsion systems. Currently she is the project manager with Boeing's Lean-Agile Software Services (LASS) and an active BASP Coach for Unmanned Air Systems, Radio, and research programs. Ms Marbach holds an MS degree in Engineering from UCLA.

Larri Rosser is a Raytheon Certified Architect in Raytheon Intelligence, Information and Services. She has 30 years industry experience in aerospace, defense and computing technology, multiple patents in the area of human-computer interface and a BS in Information Systems and Computer Science from Charter Oak State College. Currently, she holds the role of Systems Engineering, Integration and Test lead for the DCSG-A family of programs, where she practices Agile Systems Engineering with a cross functional team.

Gundars Osvalds, ESEP and Certified Scrum Master is a Principal Systems Engineer with over 40 years of experience is currently at BCT-LLT. He provides systems engineering support to DoD programs ranging from Research and Development to large scale transformation utilizing Information Technology with architecture design, architecture framework development, engineering process creation, model based system design, agile software and engineering design. He is an author of 6 papers and 11 presentations on systems and architecture engineering presented to local engineering groups, international conferences and publications.

David Lempia is a Principal Systems Engineer in the Engineering Infrastructure Development & Lean organization of Rockwell Collins with over 20 years of experience in systems development. He is currently leading lean for the EID&L organization and working as a skilled modeler. As a skilled modeler he leads workshops, develops best practices and develops training materials for Rockwell Collins. He is an author for papers on Requirements Engineering Management, Model Based Development, and Lean.