Agile Essence

The 29th International Conference of the System Dynamics Society

Washington, DC

July 24 – July 28, 2011

Table of Contents

List of Figures

Introduction

Agile software development (Agile) has emerged as a successful software development methodology. This paper investigates what is the essence of Agile that has enabled its success.

Since the publication of the Manifesto for Agile Software Development approximately a decade ago [1], many software development projects have used Agile. There are several variants of Agile, e.g., Extreme Programming (XP), Dynamic Systems Development Method (DSDM), and Scrum, which will be addressed in this paper.

Agile Methodology

Agile is associated with rapid, successful software product development. Agile software development is based on adaptive, iterative, and incremental software deliveries. Requirements and their solutions evolve through collaboration between stakeholders and developers. The stakeholders and developers meet periodically, typically monthly, to discuss the stakeholders' desired product. An understanding and agreement is reached and Sprints are defined to deliver pieces of the desired product in short scheduled time frames, e.g., weekly, with usually a demonstrable capability monthly called a Release. Each Sprint is tested and integrated together as they are completed and become parts of a Release. The Releases accumulate incrementally into a final delivered product. There is minimal paperwork or documentation associated with Agile; face-to-face interaction is emphasized.

As each Sprint is completed and demonstrated, the stakeholder has the opportunity to make course corrections to the short-term and long-term product needs. Likewise, the developers have the opportunity to make corrections to software and system errors to keep the overall project on track. Agile enables software development, teamwork, collaboration, and process adaptation throughout the development life-cycle.

Agile and Waterfall Methodology

Agile is often contrasted with plan-driven software methods sometimes referred to as Waterfall. Waterfall specifies up-front the software to be developed, serially follows a schedule of events, e.g., design, develop, test, and delivers a final product in one typically large process cycle.

Waterfall requires an abundant amount of paperwork and documentation up front and during its life-cycle. Once the product specification is completed and documented by paperwork, the development team has little interaction with the stakeholders until the product is ready for delivery. Waterfall is associated with large cost overruns and failed projects. Given the widely held view that there are numerous examples of failed and overly expensive systems attributed to Waterfall, at least one author has questioned how these failures and overruns could continue to happen (McCabe and Polen [2]) without some attempt to improve the process, e.g., using Agile.

Agile and Other Domains

Interestingly, Agile has reached across multiple domains beyond software development. Turner [3] discussed Agile and its application to system engineering. He postulated that traditional system engineering may not fit today's and tomorrow's Agile systems because

of its inherent Waterfall orientation. Cockburn [4] wrote that Agile software development is similar to agile manufacturing given that decisions in Agile software development corresponds to a part in an agile manufacturing line, e.g., both flow through a network, wait in queues at bottlenecks, and have throughput delays. With this equivalence, he contended that there is a real parallel between Agile software development and agile manufacturing. McMahon [5] examined how Agile could be applied to address Waterfall shortfalls under today's defense acquisition regulations, DoD/National Security Space Acquisition Policy 03-01. Lastly, Hicks and Foster [6] have adapted Agile toward more efficient management of academic research groups. They call the process SCORE (SCrum fOr REsearch). They note the following benefits: more efficient time use for faculty, improved student productivity, and improved group identity and shared knowledge.

Statement of the Problem - The Hypothesis

The purpose of this paper is to examine what it is that makes Agile agile, i.e., its essence. The hypothesis is that the essence of Agile is leveraging process feedback loops, and using single and double loop learning.

Agile SCRUM Background and Description

In 1986 Hirotaka Takeuchi and Ikujiro Nonaka [7] published "The new new product development game" and launched a holistic approach to new product development with six characteristics: built-in instability, self-organizing project teams, overlapping development phases, "multilearning," subtle control, and organizational transfer of learning. The approach was modeled after Rugby's Scrum because "the product development process emerged from the constant interaction of a hand-picked, multidisciplinary team whose members work together from start to finish" [7, p138]. From this foundation, the "Manifesto for Agile Software Development" (Manifesto) emerged [1].

Ken Schwaber was a signature of the Manifesto. From the software development perspective, Schwaber described Scrum as "…devised specifically to wrest usable products from complex problems" [8]. Scrum guides the software development process toward the most valuable outcome possible.

As illustrated in the activity diagram, Figure 1, the Scrum software development process starts with a vision of the system to be developed. The Stakeholder is responsible for delivering the vision in a cost-effective manner. The Stakeholder manages the Product Backlog. The Product Backlog is a list of requirements that will deliver the vision. The Product Backlog has the items most likely to generate value as top priority. The Product Backlog may change frequently to reflect changing business requirements and how quickly or slowly the Sprint developers can transform Product Backlog into functional Releases. It is generally assumed that the Vision is correct and remains constant.

Work is done in Sprints, an iteration of approximately 30 consecutive calendar days. Each Sprint is initiated with a Sprint planning meeting. The Stakeholder explains to the development team what is desired, and the development team informs the Stakeholder how much of what is desired is feasible over the next Sprint.

At daily Scrums, the development team discusses these questions: What have you done since the last meeting? What do you plan on doing before the next Scrum meeting? What impediments stand in the way of what you plan to achieve? The daily Scrum synchronizes the work of all development team members.

At the end of the Sprint, a demonstration is given, where the development team presents what was completed to the Stakeholder. Informally, the functionality is presented and discussed with the intention to release it and collaboratively determine what the Scrum team should do next.

After the Sprint review and prior to the next Sprint planning meeting, a Sprint retrospective meeting is held to encourage the development team to revise its development process to make it more effective and enjoyable for the next Sprint.

Scrum addresses the complexity of software development by using empirical process control with a set of simple practices and rules [8]. In sum the Sprint planning meeting, the Daily Scrum, the Sprint review, and the Sprint retrospective constitute the empirical visibility, inspection and adaptation elements.

Empirical process control enables Scrum to establish a repeatable software development process with acceptable quality. By visibility, Schwaber means that the process that affects the outcome must be visible and true to those controlling the process. The empirical process must be inspected frequently enough to detect unacceptable variances. The inspector must possess the requisite skills

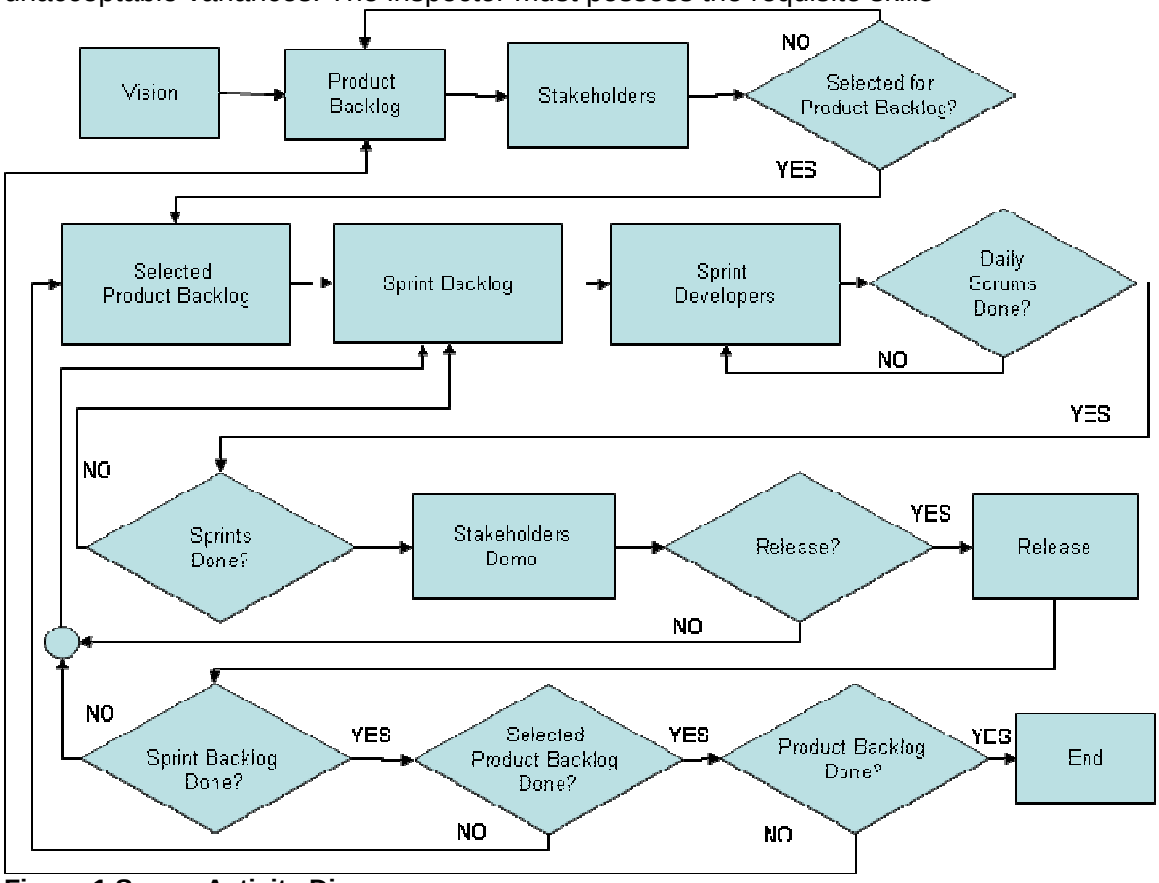

Figure 1 Scrum Activity Diagram

to assess what is being inspected. The last element of empirical process control is adaptation. The inspector must adjust the process if from the inspection the software development process is outside acceptable limits and that the resulting product will be unacceptable. The adjustment must be timely to minimize further deviation.

Analysis of Agile SCRUM

This section investigates Agile from the following perspectives: feedback loops, and single and double-loop learning. There is recognition within the Agile community that feedback has a role, but there is little explicit description of the feedback or its role. The sections below will describe some feedback and roles for consideration. System Dynamics is used to analyze Agile feedback mechanisms. The feedback concept is at the heart of the system dynamics approach. Causal Diagram (CD) models and Structure Chart (SC) models are tools used to visualize feedback.

Agile Process Feedback Mechanisms

Tarr, Williams, & Hailpern [9] presented four feedback loops involving the stakeholders of an Agile software development project and the development team as illustrated in Figure 2. This is significant as an initial recognition of the role feedback plays in Agile.

Ferreira & Cohen [10] indicated that the greater the degree of customer feedback within the systems development process, the greater the stakeholder satisfaction. Regular feedback helped organizations recognize necessary requirements changes by allowing customers ample time to voice their desired changes, which in turn, allowed customers to get what they wanted. The significance is the recognition that feedback is central to a move toward user-centered design.

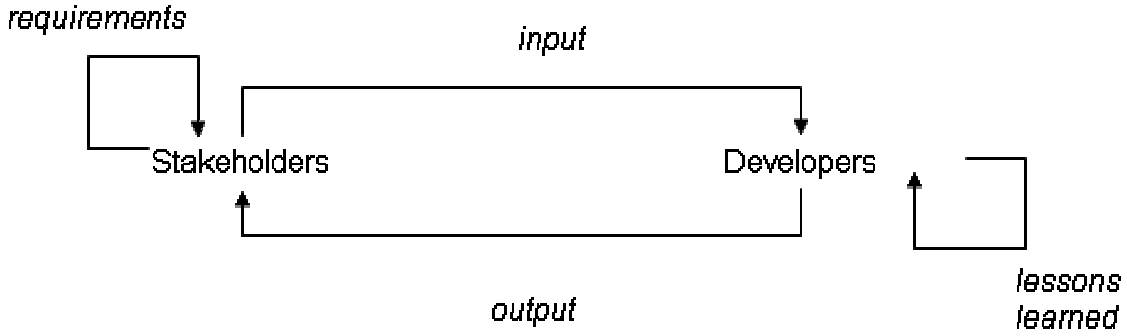

Figure 2 Adaptive Development Feedback Loops (Adapted from: [9, p23])

Vanderburg [11, p543] regarded Scrum as an overall feedback structure with a series of iterations. He described Scrum feedback, as gathered in a whole-team setting, to probably have an amplifying effect. The recognition of the possibility that feedback could have an amplifying effect is significant.

Tignor [12] questioned whether agile project management had a unique feedback structure or would fit within the generic conceptually formed system dynamic project management structures identified by Lyneis and Ford [13]. He found evidence in the literature that there were many areas of agile project management that could benefit from knowledge about and application of the Lynies and Ford model. Overall, the literature showed that there is attention applied to the Agile project management process. The holistic approach encapsulated by Lyneis and Ford could benefit Agile,

particularly as a way to monitor and control the potential impact of recursive error detection and rework during the Agile development process.

Chichakly [14] created a System Dynamics model and simulated Scrum. An Agile advantage is that the early ability to discover errors reduces the overall schedule and project cost.

A view of the feedback loops derived from the Scrum activity diagram is presented in Figure 3 as a causal diagram model. A CD shows links with arrows going from cause to effect [15, p102]

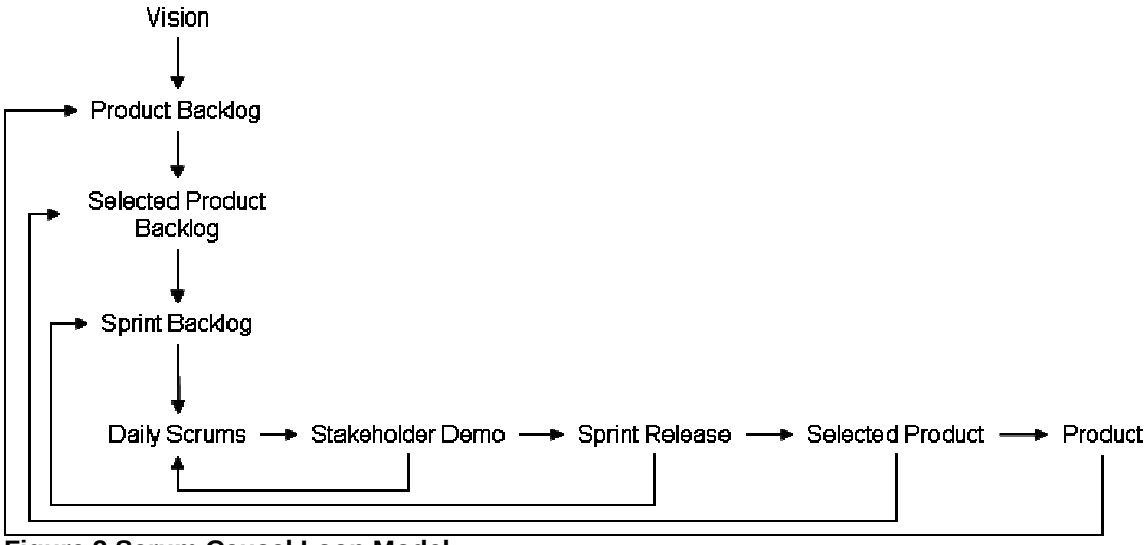

Figure 3 Scrum Causal Loop Model

Figure 3 illustrates that there is not, generally, a causal loop connection to Vision. The loops identified are the primary causal loops that govern the Agile process. In general the forward flowing arrows are positive causal loops, i.e., the more x increases, the more y increases. For example, the more Selected Product Backlog increase, the more Sprint Backlog increases. In contrast, the backward flowing arrows are negative loops, i.e., the more x increases, the more y decreases. For example, the more that Selected Product increases, the more that the Selected Product Backlog decreases. Overall, the feedback loops of the Scrum Causal Loop model are negative.

A Scrum structure chart model was developed from the causal loop diagram, see Figure 4. An SC shows the underlying physical structure of a system and consists of stocks and flows, where a stock accumulates and flows move entities through the system [15, p102].

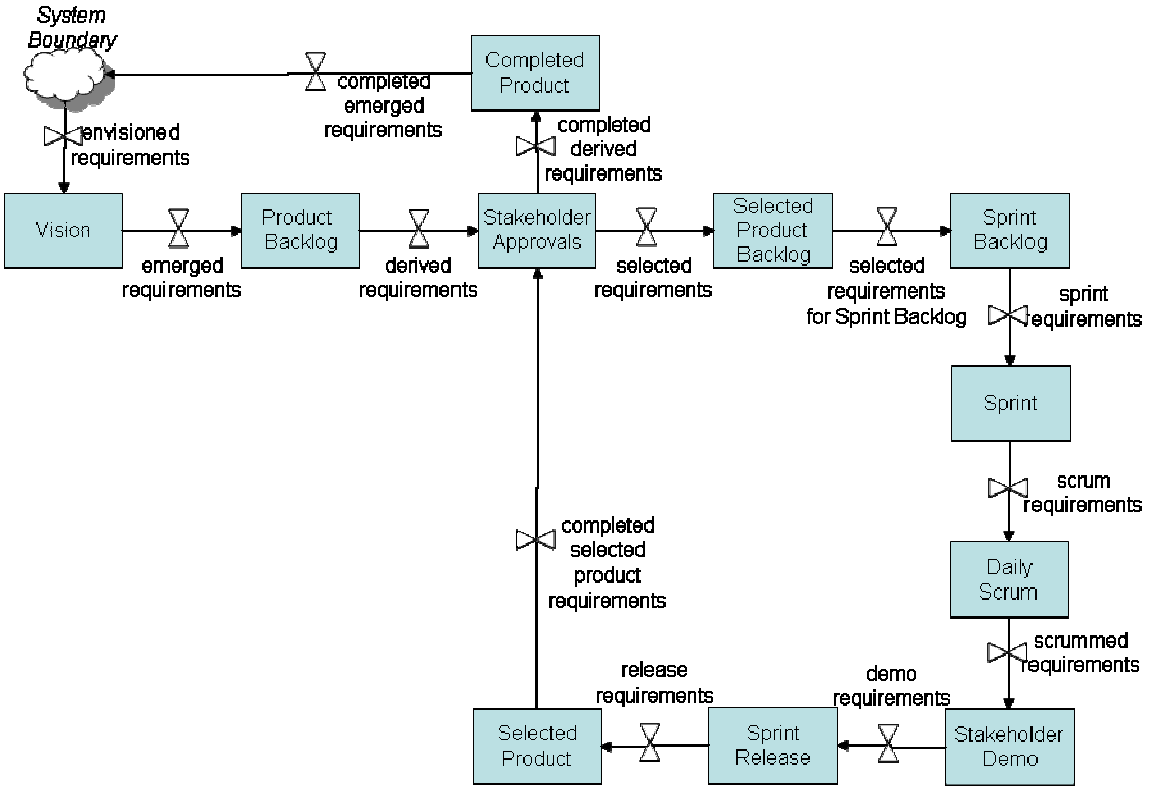

Figure 4 Scrum Structure Chart Model

This SC presents the Agile workflow of requirements through the system and the points where the requirements are accumulated. For example, emerged requirements flow from Vision where they are first gathered together and next are accumulated in the Product Backlog. From the Product Backlog, derived requirements flow and are accumulated in Stakeholder Approvals. There are numerous variables that need to be considered for the SC model to be effective, e.g., the number of requirements envisioned, the rate requirements emerge from Vision, the initial requirements in the Product Backlog, and the rate derived requirements are produced. For instance, there may be an auxiliary variable, number of stakeholders that will influence the rate of review and analysis of the Product Backlog.

Agile SCRUM Single and Double Loop Learning

Sterman explained that System Dynamics is a method to help us learn about complex systems and that learning depends on feedback [15, p4]. As discussed earlier in this paper, negative feedback loop denotes self-correction or balancing. In contrast a positive feedback loop denotes self-reinforcement or growth.

Decision makers use information feedback (qualitative and quantitative) to make decisions to influence or close gaps between actual and desired states in the real world. Chris Argyris [16] described single loop learning as a process that enables an organization to carry on its present policies or achieve its objectives. For this paper, think of “objectives” as analogous to “requirements”. He compared single loop learning to a thermostat that learns when it is too hot or too cold and then turns the heat on or off. The thermostat performs this task because it can receive information feedback, room temperature, and therefore take corrective action.

He continued with a description of double loop learning. If the thermostat could question itself about whether regarding what the temperature should be, it not only would be detecting error but questioning the underlying policies and goals as well as its own algorithm.

Questioning the underlying policies and goals and its own purpose, is a comprehensive inquiry that he called double loop learning. For this paper think of "purpose" as "vision". Together, single and double loop learning were molded into "Theories of Action" [16, p118].

Polat [17] created a thermostat simulation experiment of single and double loop learning. This is an early simulation model of double-loop learning. Diane McGinty Weston identified two of Senge's [18] core learning organization practices as "mental modeling" and "action learning".

The Agile single loop learning feedback, illustrated in Figure 5, is like the thermostat model. For example, it learns when a selected requirement is completed or having difficulty being completed through Scrums and then starts a new selected requirement or makes Scrum adjustments to enable completion. The Agile single-loop model performs this task because it can receive information feedback from Sprints and Scrums and therefore take corrective action.

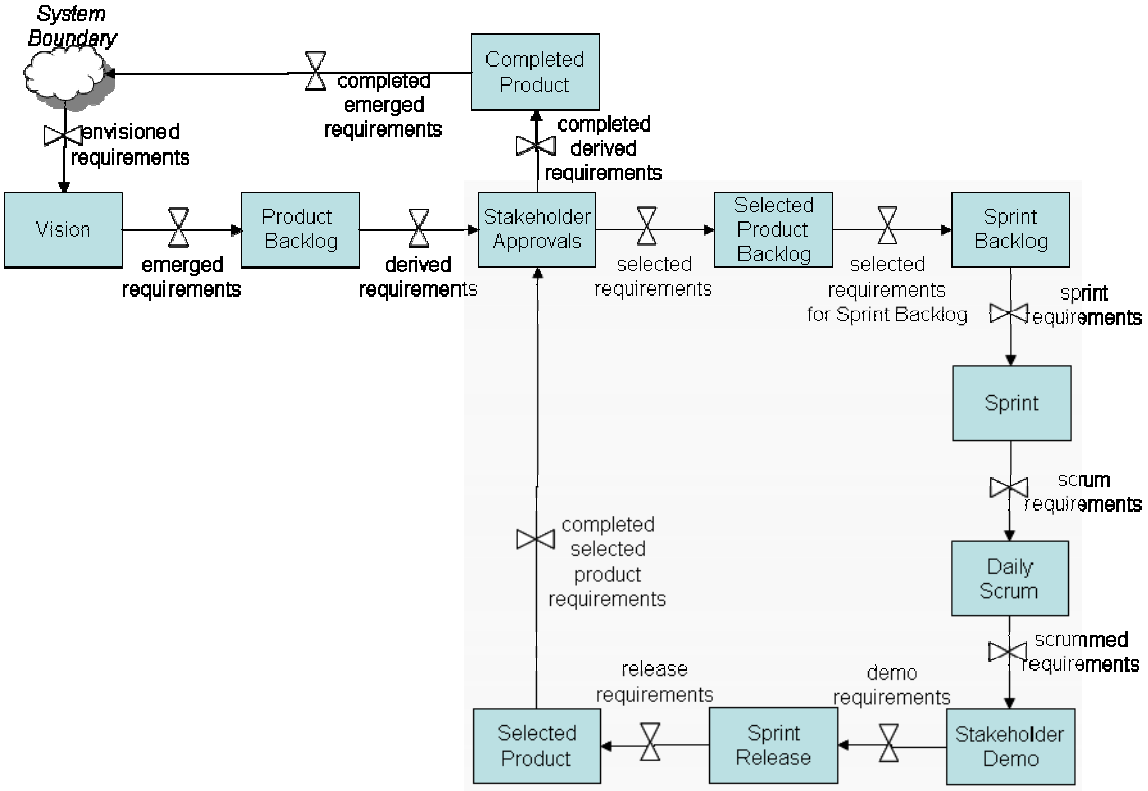

Figure 5 Single Loop Learning Structure Chart Model

According to Sterman [15], single-loop learning is limited as it does not address the following: changing our mental models (e.g., vision), contributing to our understanding of the causal structure of the system, defining the boundaries of the system, identifying

the relevant time horizon, or goals and values. Single-loop learning reinforces our current system view, and does not contribute to expanding it.

The Agile double loop learning feedback, illustrated in Figure 6, is like the thermostat model that could question itself regarding what the temperature should be, and questioning the underlying policies and goals. The Agile double loop model would perform this task because if it could question the Vision, emerged requirements and Product Backlog as wells as receive information feedback from Sprints and Scrums and therefore take corrective action.

Agile information feedback has the capacity to alter our decisions within the context of the Agile framework and decision rules (single loop feedback), but it also has the capacity to alter our visions (double loop feedback). A change of vision in turn has the capacity to change our view of the causal diagram model and structure chart model of systems, creating new decision rules and strategies. In this context, the same Agile feedback information, processed and interpreted by a different vision and decision rules may result in a different product, see Figure 7.

From a double loop learning perspective, altering the structure of a system will change a pattern of behavior over time. For double-loop learning to be effective, the cycle time around the loops must be nearly synchronous with changes in the real world in order to replace an existing obsolete vision with a new one. For example, from the time Lancaster performed a controlled experiment of the ability of lemon juice to prevent scurvy to its use routinely to eradicate scurvy took approximately 265 years [15];

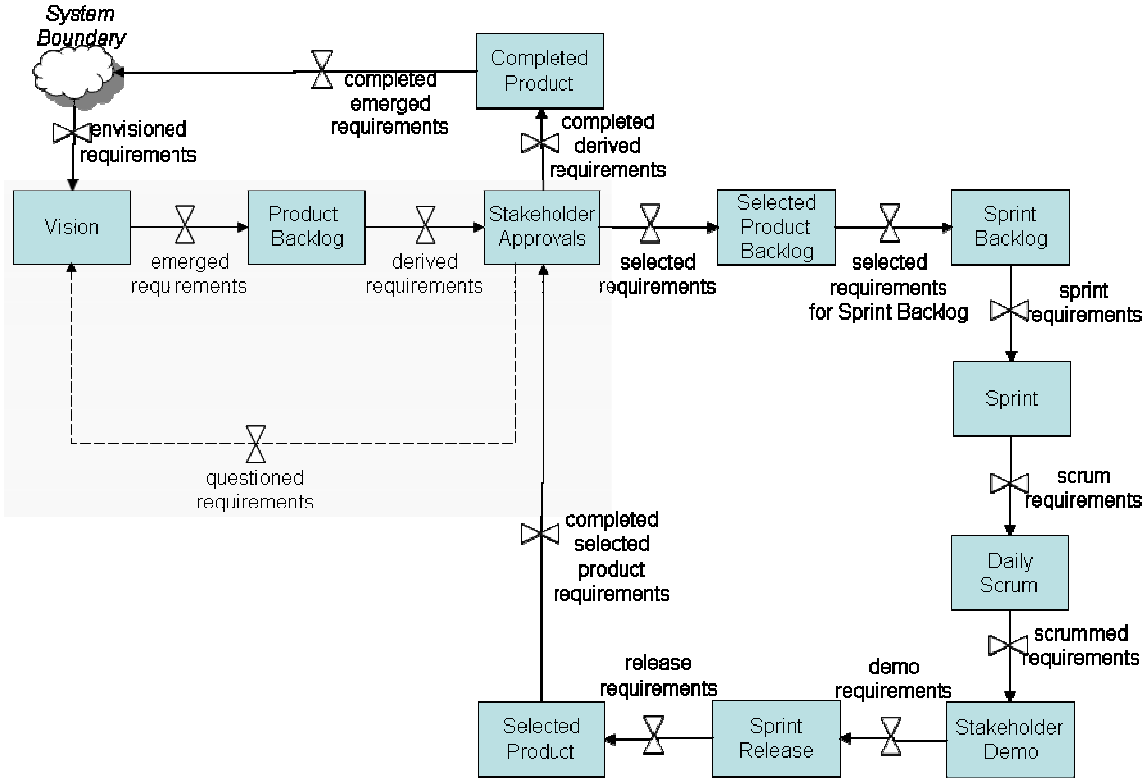

Figure 6 Double Loop Feedback area

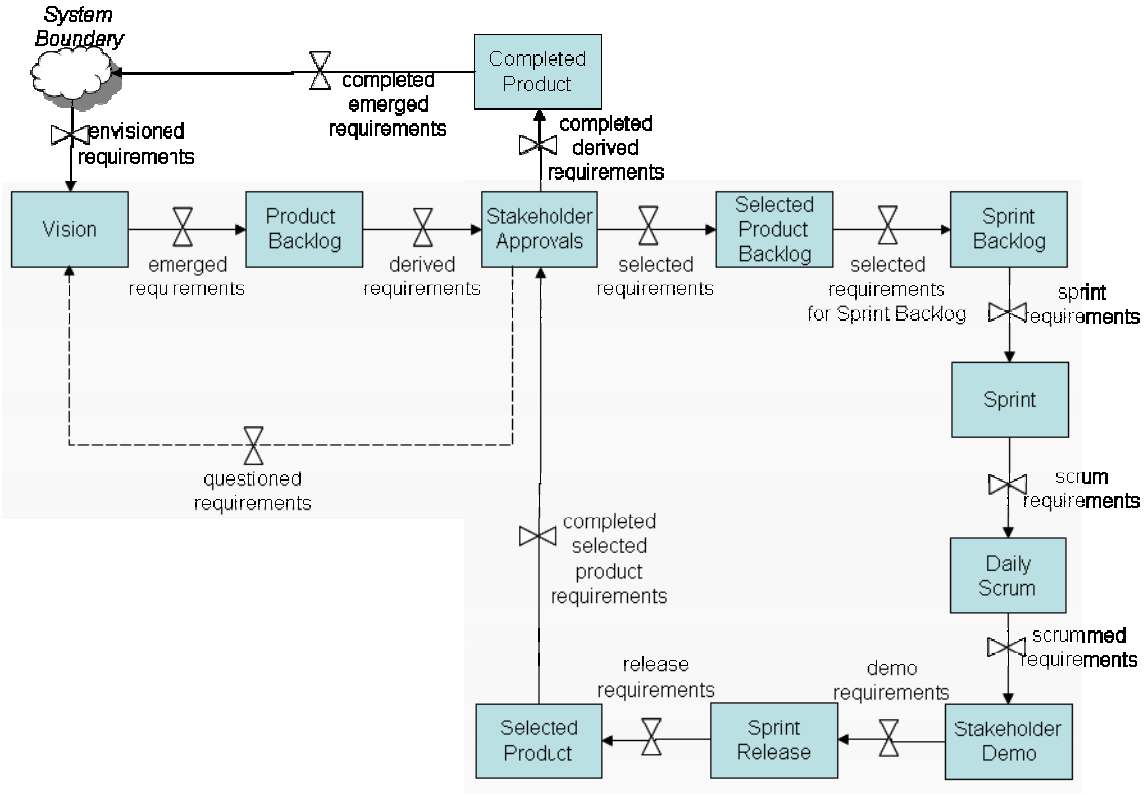

Figure 7 Double Loop Structure Chart Model

clearly a time delay out of synchrony with the need.

Conclusions

Agile is a software methodology for solving complex problems based on its adaptive, iterative, and incremental properties. Agile's underpinnings are subtle and powerful. Because the essence of Agile is feedback, and single and double loop learning, Agile has the flexibility to adapt to complex problems, and cross over to other domains, e.g., system engineer, based largely on its flexibility.

Although the descriptions of Agile and examples of its application acknowledge that feedback plays a role; feedback is generally overlooked as an Agile detail. The degree that feedback underpins Agile is significant upon closer inspection. Seeing that there are opportunities for single and double loop learning is powerful. Single loop learning will help Agile manage its backlogs. Double loop learning will help Agile manage its vision. The future is Agile. Recognition and usage of feedback as a part of the Agile methodology will further advance its successfulness in the future.

References

1 Beck, Kent, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Arie Van Bennekum, Andrew Hunt, Ken Schwaber, Alistair Cockburn, Ron Jeffries, Jeff Sutherland, Ward Cunningham, Jon Kern, Dave Thomas, Martin Fowler, and Brian Marick. "Manifesto for Agile Software Development." Manifesto for Agile Software Development. 2001. Web. 22 Sept. 2010. <http://agilemanifesto.org/>.

2 McCabe, Rich, and Michael Polen. "Should You Be More Agile?" CROSSTALK The Journal of Defense Software Engineering (2002): 30. Web. 22 Sept. 2010. <http://www.stsc.hill.af.mil/crosstalk/2002/10/mccabe.html>.

3 Turner, Richard. "Toward Agile Systems Engineering Processes." CROSSTALK The Journal of Defense Software Engineering (2007): 11-15. Web. 23 Sept. 2010. <http://www.stsc.hill.af.mil/crosstalk/2007/04/0704CrossTalk.pdf>.

4 Cockburn, Alistair. "What Engineering Has in Common With Manufacturing and Why It Matters." CROSSTALK The Journal of Defense Software Engineering (2007): 4-7. Web. 23 Sept. 2010. <http://www.stsc.hill.af.mil/crosstalk/2007/04/0704CrossTalk.pdf>.

5 McMahon, Paul. "Defense Acquisition Performance: Could Some Agility Help?" CROSSTALK The Journal of Defense Software Engineering (2009): 14-17. Web. 23 Sept. 2010. <http://www.stsc.hill.af.mil/crosstalk/2009/02/0902CrossTalk.pdf>.

6 Hicks, Michael, and Jeffrey S. Foster. "SCORE: Agile Research Group Management." Communications of the ACM 53.10 (2010): 30-31. Print.

7 Takeuchi, Hirotaka, and Ikujiro Nonaka. "The New New Product Development Game." Harvard Business Review Jan-Feb (1986): 137-46. Web. 12 Feb. 2011. <http://www.sao.corvallis.or.us/drupal/files/The%20New%20New%20Product%20Development%20Game.pdf>.

8 Schwaber, Ken. Agile Project Management with Scrum. Microsoft, 2004. Books24x7. Web. 23 Jan. 2011. <http://common.books24x7.com/book/id_8392/book.asp>.

9 Tarr, P., Williams, C., & Hailpern, B. “Toward Governance of Emergent Processes and Adaptive Organizations.” SDG’08, Leipzig, Germany, 2008.

10 Ferreira, C. & Cohen, J. “Agile systems development and stakeholder satisfaction: a South African empirical study.” Proceedings of the 2008 Annual Research Conference of the South African institute of Computer Scientists and information Technologists on IT Research in Developing Countries: Riding the Wave of Technology, Wilderness, South Africa (October 06 - 08, 2008). SAICSIT '08, vol. 338. ACM, New York, NY, 48-55.

11 Vanderburg, G. “A Simple Model of Agile Software Processes– or –Extreme Programming Annealed.” OOPSLA’05 (2005). San Diego, California.

12 Tignor, Warren W. "Agile Project Management." Proceedings of The 27th International Conference of the System Dynamics Society 2009 Albuquerque, New Mexico (2009): 1148. Web. 16 Jan. 2011. <http://www.systemdynamics.org/conferences/2009/proceed/papers/P1148.pdf>.

13 Lyneis, J., Ford, D. “System dynamics applied to project management: a survey, assessment, and directions for future research.” System Dynamics Review, Vol.23.No. 2/3 (2007): 157-189.

14 Chichakly, Karim. "Modeling Agile Development: When Is It Effective?" Proceedings of The 25th International Conference of the System Dynamics Society and 50th Anniversary Celebration 2007 Boston, Massachusetts (2007): 385. Web. 16 Jan. 2011. <http://www.systemdynamics.org/conferences/2007/proceed/papers/CHICH380.pdf>.

15 Sterman, John D. "Learning in and about Complex Systems." Business Dynamics: Systems Thinking and Modeling for a Complex World. Boston: McGraw-Hill, 2000. 3-39. Print.

16 Argyris, Chris. "Double Loop Learning Organizations." Harvard Business Review Sep-Oct (1977): 115-25. Web. 12 Feb. 2011. <http://www.westernsnowandice.com/09-Presos/DoubleLoop.pdf>.

17 Polat, Seckin. "The Simulation of Learning: Comparison of Double and Single Loop Learning." Proceedings of The 15th International Conference of the System Dynamics Society Istanbul, Turkey (1997). Web. 29 Jan. 2011. <http://www.systemdynamics.org/conferences/1997/default.htm>.

18 Senge, Peter M. "Introduction to the Paperback Edition." Introduction. The Fifth Discipline. New York: Doubleday, 1990. xi-xii. Print.