2012

Improving Systems Engineering Effectiveness in Rapid Response Development Environments

Turner, Richard; Madachy, Raymond; Ingold, Dan; Lane, Jo Ann

IEEE

Turner, Richard, et al. "Improving systems engineering effectiveness in rapid response development environments." Proceedings of the International Conference on Software and System Process. IEEE Press, 2012.

https://hdl.handle.net/10945/60708

Improving Systems Engineering Effectiveness in Rapid Response Development Environments

Richard Turner
Stevens Institute of Technology
Hoboken, NJ, 07030, USA
Richard.Turner@stevens.edu

Raymond Madachy
Naval Postgraduate School
Monterey, CA, 93943, USA
rjmadach@nps.edu

Dan Ingold, Jo Ann Lane
University of Southern California
Los Angeles, CA, 90089, USA
dingold@usc.edu, jolane@usc.edu

***Abstract*—Systems engineering is often ineffective in development environments where large, complex, brownfield systems of systems are evolved through parallel development of new capabilities in response to external, time-sensitive requirements. This paper defines a conceptual framework to improve that effectiveness and better integrate the systems engineering and software engineering processes. The framework is based on a services approach to systems engineering and the use of kanban techniques to schedule scarce enterprise systems engineering resources across multiple related systems and software development projects. The framework also addresses the differing value of work items to multiple stakeholders in the scheduling and coordination processes. Models and simulations are being used to capture, refine and validate the framework prior to in vivo experimentation.**

***Keywords*-systems engineering process; process integration; service-based systems engineering; value-based engineering; integrating software and systems engineering; kanban processes**

I. Introduction and Background

Traditional systems engineering (SE) developed half a century ago, primarily driven by the challenges faced in the aerospace and defense industries. The environment was fairly uniform – hardware-driven, long lived, single mission. The result of this uniformity was practices that worked well in that specific context were seen as "best practices," and came to define the discipline of systems engineering. Engineering principles involving agility and leanness have been adopted to address non-determinism in software systems. [1] [2] [3]. Combining agile-lean software experience with system engineering fundamentals can provide practical, principle-driven agile-lean systems engineering approaches for the design of complex or evolving hardware-software-human systems [4]. This may help alleviate the observed poor performance of systems engineering in meeting schedule and resource constraints [5] [6] [7].

This research proposes marrying the ideas of a services perspective with a lean-inspired pull scheduling technique such as kanban, to create a radical departure from the normal concepts of systems engineering. In an environment where there is an existing complex system constantly evolving through rapid-response software application development, systems engineering is the glue that holds all of the various projects together. It is critical that it be integrated into the various projects without unduly delaying them, and that the limited resource of systems engineering skills be efficiently and effectively deployed so as not to unduly delay any particular project and still meet the overall system priorities. The services approach better integrates SE into the development cycle, and the kanban-based scheduling maximizes the value flow of the systems engineering tasks performed. This project has developed an example of the combined approach and is simulating it with a hybrid of discrete event, continuous flow, and agent-based models and typical work streams to determine if the idea is sound enough to actually pilot in an operational environment.

II. Beginning with Kanban

A. Background

Kanban is a method associated with lean manufacturing and the Toyota Production System. A kanban (signal card) approach is a form of on-demand scheduling that provides a visual means of managing the flow within a process. The signal cards are created to the agreed capacity of the process and one card is associated with each piece of work. In manufacturing, work can mean the creation of a part, the integration of a part into an assembly, the completion of a particular analysis process, or whatever bounded and completeable activity you wish to track through the process. Once all of the cards have been associated, no more work in that process can begin until some piece of work is completed and the card becomes available. A common example of a simple kanban is the use of a limited number of tickets for entry into the Japanese Imperial Gardens [8]. The fundamental idea is to use visual signals to synchronize the

flow of work with process capacity, limit the waste of work interruption, minimize excess inventory or delay due to shortage, prevent unnecessary rework, and provide a means of tracking work progress.

In knowledge work, the components of production are ideas and information [10, 11]. In software and systems, kanban systems have evolved into a means of smoothing flow by balancing work with resource capability. The concept was extended to include the limiting of work in progress according to capacity. Work cannot be started until there is an available appropriate resource. In that way, it is characterized as an on-demand or "pull" system, since the work is pulled into the activity as capacity is available rather than "pushed" via a schedule.

B. Concept

The following concept was derived from [8] [9] [10] [11] [12] [13], workshops, and discussions with an industry working group. A kanban system is a visually monitored set of activities, where each activity has its own ready queue and set of resources to add value to work units that flow through it. The fact that queues are explicit in the system allows costs of delay and other usually invisible aspects of scheduling to be front and center in decision making. Queues also provide a vast body of experience and underlying science from the queuing theory discipline. Control of the kanban system is generally maintained through *batch size*, *Work in Progress (WIP)* limits and *Classes-of-Service* (COS) definitions that prioritize work with respect to risk.

The visual representation of work is critical to kanban success, because it provides immediate understanding of the state of flow through the set of activities. This transparency makes process anomalies (both common and special cause) or resource issues easily visible, enabling the team to recognize and react immediately to resolve the issue. Flow through the kanban system is measured and tracked through statistical methods that support tuning the control parameters to improve the system. Flow measures also provide a good handle for effectiveness comparison. Because the team and management interact with the kanban board and collectively solve problems, this aspect is important in achieving continuous improvement (kaizen).

WIP is partially-completed work, equivalent to the manufacturing concept of parts inventory waiting to be processed by a production step. WIP accumulates ahead of bottlenecks unless upstream production is curtailed or the bottleneck resolved [12]. WIP in knowledge work can be roughly associated to the number of work items that have been started and not delivered. Limiting WIP is a concept to control flow and enhance value by specifically limiting the amount of work to be assigned to a set of resources (a WIP Limit). WIP limits accomplish several goals: they lower the context-switching overhead that impacts individuals or teams attempting to handle several simultaneous work items; they accelerate useful value by completing work in progress before starting new work; and, they provide for reasonable and sustainable resource work loads.

Using small batch sizes is a supporting concept to WIP. Reducing batch size limits rework and provide flexibility in scheduling and response to unforeseen change. Smaller batch sizes help stabilize the process flow and allow downstream processes to consume the batches smoothly, rather than in a start-and-stop fashion that makes inefficient use of resources. The move from "one step to glory" system initiatives to iterative, deployable increments is an example of reducing batch size. Incremental builds and ongoing, continuous integration also approximate the effect of small batch sizes.

So as not to confuse readers with the traditional understanding of kanban in manufacturing, we refer to an implementation of such a system in systems or software engineering as a Kanban-based Scheduling System, or KSS.

III. Defining a KSS for Systems Engineering

A. An Elemental KSS

In Figure 1 we define our core building block concept of a KSS. We intend that this model be recursive at many levels to allow for complex implementations. More about the specifics of the model can be found in [8] and [19].

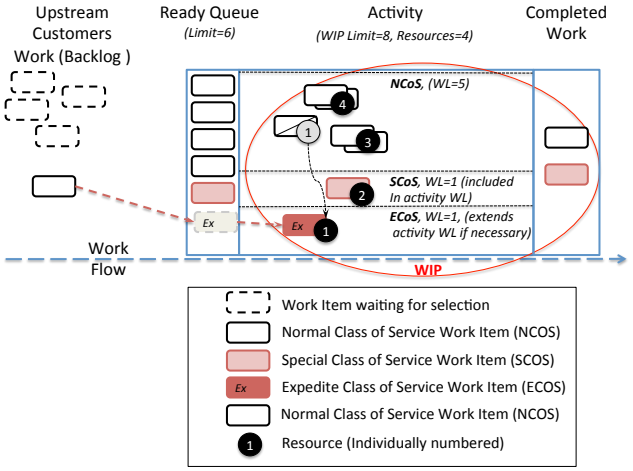

Figure 1. Kanban Scheduling System Model

This core concept can be thought of as a building block or even a recursive application of the fundamentals discussed in Section 2. In general, the upstream customer for the service provided is responsible for selecting the work that enters the KSS. This is usually done collaboratively with the KSS to make sure that significant dependencies, date certain events, and other special concerns are understood. As a resource becomes available, the highest value work item is executed until it is complete, and then added to the completed work. Depending on the delivery cadence, it may go directly to the downstream consumer or it may be held until the next delivery date.

A scheduling cadence provides regular meetings of the KSS team to assess the work flow and determine if resources should be moved between activities, WIP limits adjusted, or other actions taken. Often, this is a daily activity, but the actual planning horizon selected and the nature of the work items should be used to establish the most cost effective cadence. Planning horizon is based on the visibility into

upcoming work and is dependent on the WIP and ready queue limits.

In this illustration, the KSS consists of a single activity – and that is generally how the upstream customer would view it. However, as shown in Figure 2, it is easy enough to see that the activity and its associated ready queue could be subdivided into multiple linked instances. These could be linked sequentially or could represent different specializations for different types of services, each representing a full KSS. For example, there could be an initial activity that determines the relative value of a work item (its precedence given the current status of resources) and assigns it to the appropriate specialized service KSS.

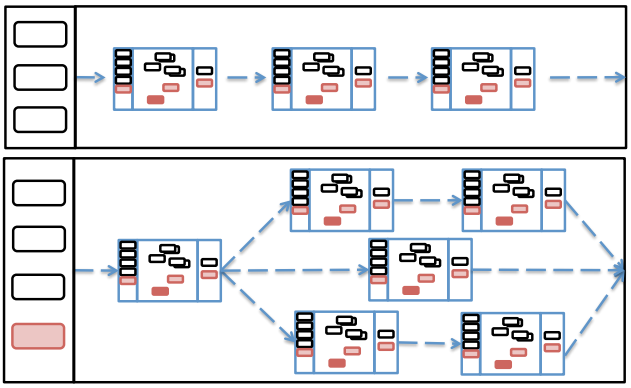

Figure 2. Kanban Scheduling System Hierarchy

There is much evidence suggesting a KSS such as this will work in a software development project, but applying it to systems engineering, particularly where the SE practitioners coordinate their work across multiple systems is unique in our experience, and requires a fundamentally different understanding of systems engineering.

B. Systems Engineering as a Service

Systems engineering has struggled with acceptance in rapid-response environments, partly because it tends to operate with a broader scope and with the assumption that a holistic view requires a deeper and fuller level of knowledge than is often available in the rapid response time frame. In rapid response environments, the time scale constrains the project scope, and detailed analysis up front is perceived as less achievable.

Agile and lean assume holism comes from a learning process and is valuable even when incomplete. The idea of using a pull system for systems engineering is an attempt to merge the breadth of SE into the rapid development rather than lay it on top of the activities. Our idea of a KSS for systems engineering is shown in Figure 3. We believe it will support better integration of SE into the rapid response software environment, better utilize scarce systems engineering resources, and improve the overall system-wide performance through a shared, more holistic resource allocation component.

In general, systems engineering is involved in three kinds of activities in rapid response environments: Up front, continuous, and requested. While critical in greenfield projects, up front work is important in all systems evolution and includes creating operational concepts, needs analysis, and architectural definitions. Continuous SE activities are ongoing, system–level activities (e.g. architecture, environmental risk management). These require not only substantial time, but also the maintenance and evolution of long-term, persistent artifacts that support development across multiple projects. Requested activities are generally specific to individual projects (e.g. trade studies, interface management), but will certainly draw on the persistent SE artifacts and knowledge.

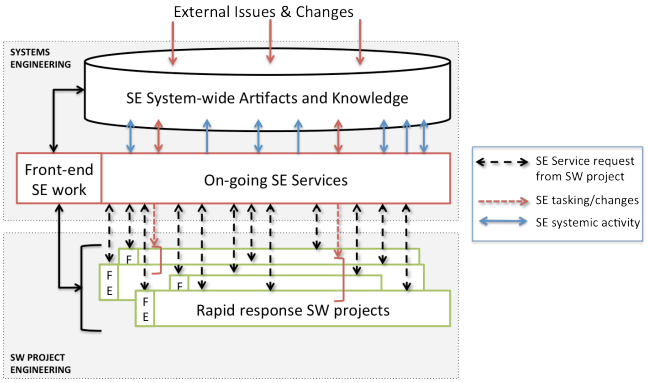

Figure 3. Overview of SE as a Service concept

By viewing the development and use of persistent artifacts as key components of services provided to various projects, SE can be opportunistic in applying its cross-project view and understanding of the larger environment to specific projects individually or in groups. It can also broker information between individual projects where there may be contractual or access barriers. When a system-wide issue or external change occurs, SE can negotiate or unilaterally add or modify tasks within affected projects to ensure that the broader issue is handled in an effective and compatible way. This is reminiscent of the agile management layer described in the iteration management approach in [13], and the approach envisioned can extend that concept throughout the rapid response lifecycle and across the multiple projects.

SE performs its services in parallel to those activities in the requesting project and then pushes the results to the requestor as soon as available. This is aimed at supporting the timeliness of projects, so that work can continue, even if at a higher risk of rework, unless waiting for the results is blocking all other work in the project (not a good thing).

SE services require persistent artifacts and knowledge for both requestor-specific and total system artifacts/ understanding. The quality of a requested service could be pre-specified, specified as a parameter or input with service request, or could be negotiated as a function of typical value and time available to provide the service. In a KSS, SE services can be thought of as a single activity. The value function used to select the next request to be handled must be designed to identify the highest cost of delay among the queued requests in terms of the overall system value. This allows SE to be a effective as possible in providing its services across the enterprise. The function could be based

on several parameters that are attributes of individual projects, individual requests, or system-wide activities. Possibilities include the maturity of the requesting project, lifecycle point of requesting project, criticality of the requesting project, and value/cost of delay/priority/class of service or other characteristics of the work impacted by the service requested. The details will be critical to achieve system wide benefits without impacting individual project timeliness. Only through modeling is the impact of various approaches to the value function determinable. In fact, modeling should be able to help identify the sweet spot of the amount and type of SE activity that produces the most value with the lowest impact to quality. Statistical and other measures will be needed to track the performance and improve the value function in vivo.

Table 1 is based on the US DoD Systems Engineering Guide for Systems of Systems [20]. It describes categories of services, specific characteristics, and likely usage in terms of type of service. A number of services can be defined in each category, and such definitions will be part of follow on research as the models are evolved. It should be noted, however, that developing the concept of SE services is outside the scope of the currently funded work. Actual service definition depends on project context and development organizations. Our early simulations do not incorporate specific task subject matter.

IV. Expected Benefits

A workshop was held January 27-28 2010 to discuss the development of a 3-year roadmap for transforming systems engineering. A number of issues identified and discussed in that meeting are addressed by the following benefits likely to accrue from the application of this research.

A. More Effective Integration and Use of Scarce Systems Engineering Resources

This approach is a value-based way to prioritize the use of scarce SE resources across multiple projects. The next-work selection policies can be tailored to provide efficient and effective scheduling that maximizes the value provided by the resource based on multiple, system-wide parameters. Having service requests including time vs. value parameters can help determine if the delay of other service requests fulfillment is warranted by the current service request.

B. Flexibility and Predictability

SE activities are generally designed for pre-specifiable, deterministic (complete and traceable) requirements and schedules. There is often an overdependence on unnecessary formal ceremony and fairly rigid schedules. Using cadence rather than schedule can provide efficient SE flow with flexibility by operating with shorter planning horizons and on-demand services. We believe that the CoS concept not only handles expedite and date-certain conditions, but also supports cross-kanban synchronization. Even though the planning is dynamic and the selection of the next piece of work to do asynchronous, we believe the use of a value-based selection function, a time-cognizant service request, customized Classes of Service, and a statistically controlled cadence provide a sufficient level of predictability where necessary.

TABLE I. Systems Engineering Service Categories (Based on [20]

Category	Description of activities within the category	Usage
Translating Capability Objectives	Proxy for customer; translating needs into requirements and specifications; support for requirements management activities; enterprise and system level requirements allocation	Up front; Continuous; Requested
Understanding Systems and Relationships	View across multiple projects; Persistent memory across time and teams	Up front; Continuous; Requested
Assessing Performance Against Capability Objectives	Validation of TPMs or other performance requirements; typical V&V type activities; operational assessments	Up front; Continuous; Requested
Developing and Evolving Architecture	Providing design guidance and supporting common architectural patterns across multiple projects; optimizing performance, throughput, maintenance through interoperable systems and system components	Continuous; Requested
Monitoring and Assessing Changes	Supporting flexibility, resilience and agility; providing surveillance of the external environment and identifying issues and changes that might affect projects, the system or the enterprise (e.g. changes to COTS products, external interfaces, systems, operating environments)	Continuous; Requested
Trade Studies And Decision Support	Supporting system-informed decision making by providing independent, competent analytical services	Up front; Requested

C. Visibility And Coordination Across Multiple Projects

In highly concurrent engineering, the KSS provides a means of synchronizing activities across mutually dependent teams by coordinating their activities through changing value functions (work item priority) according to the degree of data completeness and maturity (risk of change). It also provides an excellent way to show where work items are and the status of work-in-progress and queued or blocked work. A common visualization of work item status also encourages a sense of collective responsibility. In addition, the on-demand/limited planning horizon of the KSS actually reduces the impact of long latency dependency between work items by not beginning work on items that would then languish until another work item was complete.

D. Low Governance Overhead

Implementing a KSS doesn't require major changes in the way work is accomplished or imply specific organizational structures like other agile methods (e.g.

Scrum). Such systems can be set up in individual projects and allowed to evolve into more effective governance over time as the project and the organization as a whole understand the best way to attain value from the practices. Even the systems engineering resource scheduling can be implemented with very little organizational impact. Practitioners make most decisions using parameters set by management (e.g. WIP limits) and their own understanding of the needs. Issues are usually identifiable from walking the visible representation of the flow status and so are made clear to all who take part in the scheduling, including management. Metrics are inherent to the system, clearly identify problems, and track improvements. Most problems tend to be self-correcting.

E. Increased Project and System Value Delivered Earlier

Most lean and agile approaches provide value to the customer as quickly as possible. In rapid development environments this is particularly important. By limiting WIP, more closely integrating the SE and project engineering activities, and providing both specific project and system-wide work item value determination, the KSS provides an intentional approach to achieving early value. Nevertheless, through Classes of Service, the KSS still provides for intangible or long-term investment activities to flow through the system with minimal impact on urgent activities.

V. FUTURE WORK

This paper has described the development of a new approach to managing systems engineering in an environment where rapid response software development projects incrementally evolve capabilities of existing systems and/or systems of systems. A second part of our research is the modeling and simulation of this approach to determine whether it represents a more effective way than traditional scheduling and management paradigms. Initial simulation efforts are described in a separate paper [19]. We continue to iterate the concept, the models, and the simulations to determine if the approach is sufficiently likely to provide the hypothesized benefits in live implementation. Using the work so far, we will gather additional baseline data to refine and calibrate the models and simulations. We will also evolve the model of SE services and include human/social aspects of the processes; this is particularly important in implementing more closely coupled systems, software, and stakeholder development collaborations.

ACKNOWLEDGEMENT

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contract H98230-08-D-0171. SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

REFERENCES

[1] Boehm, Barry and Turner, Richard (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Boston, MA: Addison Wesley.

[2] Larman C. and Vodde, B. (2009). Scaling Lean & Agile Development. Boston, MA: Addison Wesley.

[3] Poppendiek, Mary. (2007). Implementing Lean Software Development: Boston, MA: Addison Wesley.

[4] Turner, Richard and Wade, J. (2011). Lean Systems Engineering within System Design Activities, Proceedings of the 3rd Lean System and Software Conference, May 2-6, 2011, Los Angeles, CA.

[5] NDIA-National Defense Industrial Association (2010). Top Systems Engineering Issues In US Defense Industry. Systems Engineering Division Task Group Report, http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Studies/Top%20SE%20Issues%202010%20Report%20v11%20FINAL.pdf. September, 2010.

[6] Turner, Richard, Shull F., et al (2009a) "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs: Phase 1 Final Technical Report," Systems Engineering Research Center, SERC-2009-TR002, September 2009.

[7] Turner, Richard, Shull F., et al (2009b) "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs: Phase 2 Final Technical Report," Systems Engineering Research Center, SERC-2010-TR004, December 2009.

[8] Anderson, David. (2010). Kanban: Successful Evolutionary Change for Your Technology Business. Sequim, WA: Blue Hole Press

[9] Reinertsen, Donald G. (2010). The Principles of Product Development Flow. Redondo Beach, CA: Celeritas Publishing.

[10] Poppendieck, Mary, and Tom Poppendieck. (2003). Lean Software Development: An Agile Toolkit. The Agile Software Development Series. Boston: Addison-Wesley.

[11] Morgan, James M, and Jeffrey K Liker. (2006). The Toyota Product Development System: Integrating People, Process, and Technology. New York: Productivity Press.

[12] Goldratt, Eliyahu M., and Jeff Cox. (2004.) The Goal: a Process of Ongoing Improvement. Great Barrington, MA: North River, 2004.

[13] Anderson et al., "Studying Lean-Kanban Approach Using Software Process Simulation." A. Sillitti et al. (Eds.): Agile Processes in Software Engineering and Extreme Programming, Part 1, Lecture Notes in Business Information Processing, Volume 77, 2011.

[14] Heath, B. et al. (2009.) A survey of agent-based modeling practices (January 1998 to July 2008). Journal of Artificial Societies and Social Simulation. 12:4 2009.

[15] Borshchev, A., and A. Filippov. 2004. From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In Proceedings of the 22nd International Conference of the System Dynamics Society, 25–29.

[16] M. Kellner, R. Madachy and D. Raffo, Software Process Simulation Modeling: Why? What? How?, Journal of Systems and Software, Spring 1999

[17] R. Madachy, Software Process Dynamics, Wiley-IEEE Press, Hoboken, NJ, 2008

[18] Boehm, B.: Applying the Incremental Commitment Model to Brownfield Systems Development, Proceedings, CSER 2009, April 2009.

[19] Turner, R., Madachy R., Ingold D., and Lane J., "Modeling Kanban Processes in Systems Engineering," submitted to International Conference on Software and System Process 2012, 2012.

[20] Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering (2008). *Systems Engineering Guide for Systems of Systems, Version 1.0*. Washington, DC: ODUSD(A&T)SSE, 2008.