Improving Systems Engineering Effectiveness in Rapid Response Development Environments

Richard Turner
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ, 07030, USA
Richard.Turner@stevens.edu

Raymond Madachy
Department of Systems Engineering
Naval Postgraduate School
Monterey, CA, 93943, USA
rjmadach@nps.edu

Dan Ingold, Jo Ann Lane
Center for Systems and Software Engineering
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 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. Here, 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 task 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. An often used 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. 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 a "pull" system, since the work is pulled into the process 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 task queue and set of resources to add value to work units that flow through it. The fact that queues are included 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 delays or resource issues easily visible and enables the team to recognize and react immediately to resolve the cause. Kanban is also an embodiment of the continuous improvement concept (kaizen). 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.

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. WIP in knowledge work can be roughly associated to the number of tasks that have been started and not completed. *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 tasks; 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 an SE KSS for Rapid-Response Development

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; this is shown in Figure 2. While we currently believe tasks and their associated parameters coupled with the visual representation of flow are sufficient, we may introduce new concepts to enable better communications and synchronization between the various interacting systems. More about the specifics of the model can be found in [8] and [19].

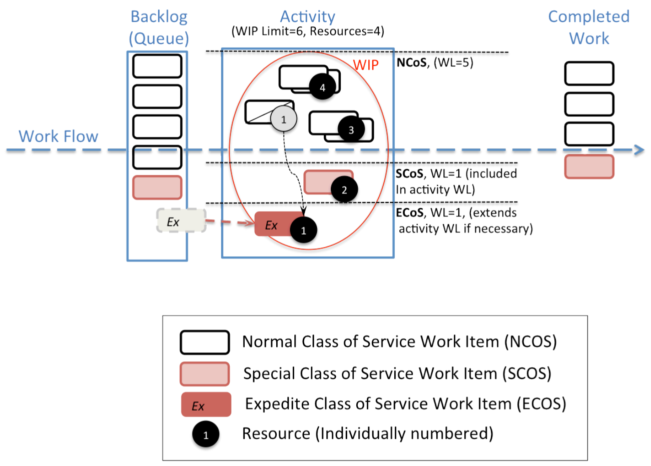
Backlog (Queue)
Activity
(WIP Limit=6, Resources=4)
Completed Work
WIP
NCoS, (WL=5)
Work Flow
SCoS, WL=1 (included In activity WL)
ECoS, WL=1, (extends activity WL if necessary)
Ex
Normal Class of Service Work Item (NCOS)
Special Class of Service Work Item (SCOS)
Expedite Class of Service Work Item (ECOS)
Resource (Individually numbered)

Figure 1. Kanban Scheduling System Model

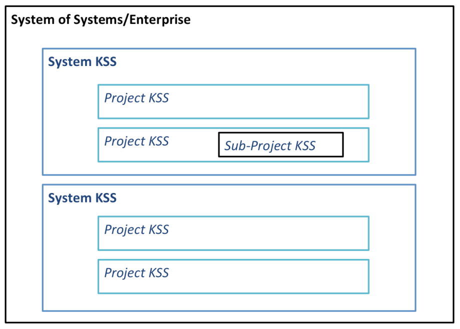

Figure 2. Kanban Scheduling System Hierarchy

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.

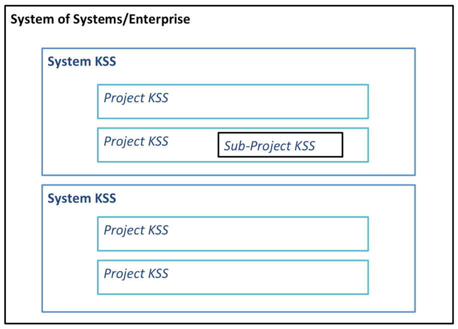

Figure 3. Kanban Scheduling System Hierarchy

In general, systems engineering is involved in three kinds of activities in rapid response environments: Up front, continuous, and taskable. Up front activities are critical in greenfield projects, but are important in all systems and system of systems evolution. They include 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. Taskable activities are generally specific to individual projects (e.g. trade studies, interface management), but will certainly draw on the persistent SE artifacts and knowledge.

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).

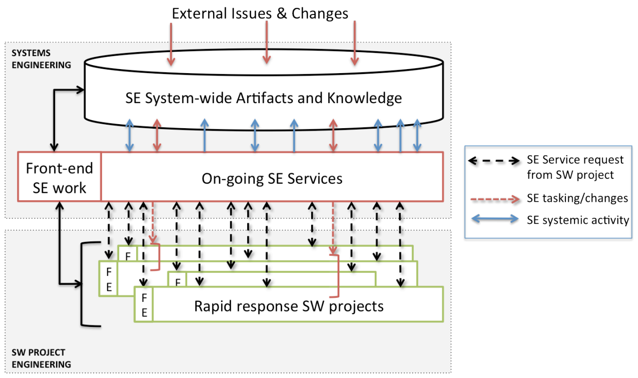

Figure 4. Overview of SE as a Service concept

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 describes categories of services. A number of services can be defined in each category, and if needed, 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. The actual definitions of services will depend on the context of the projects and the development organizations. In our simulations, we have used the more general value of work effort rather than detailing specific task subject matter.

TABLE I. SYSTEMS ENGINEERING SERVICE CATEGORIES

Category	Description	Usage
Translating Capability Objectives	Proxy for customer; support for requirements management activities	Continuous; Taskable
Understanding Systems and Relationships	View across multiple projects; Persistent memory across time and teams	Continuous; Taskable
Assessing Performance Against Capability Objectives	Validation of TPMs or other performance requirements; typical V&V type activities	Continuous; Taskable
Developing and Evolving Architecture	Providing design guidance and supporting common architectural patterns across multiple projects	Continuous; Taskable
Monitoring and Assessing Changes	Supporting flexibility and agility by providing surveillance of the external environment and identifying issues and changes that might affect projects	Continuous; Taskable
Trade Studies And Decision Support	Supporting system-informed decision making by providing independent, competent analytical services to the projects	Taskable

IV. EXPECTED BENEFITS

A workshop was held at the Stevens Institute offices in Washington, DC on January 27-28 2010 to discuss the development of a 3-year roadmap for transforming systems engineering. The meeting identified issues currently observed in instances of the rapid-response environment addressed in this paper. We believe, and are working to show, that the following benefits are reasonable to expect from the approach, and that they address a number of the issues that were discussed in that meeting.

A. More effective integration and use of scarce systems engineering resources

Using a KSS and applying a model of SE based on continuous activities and taskable services is a value-based way to prioritize the use of scarce SE resources across multiple projects. The value function within the next-work selection process can be tailored to provide efficient and effective scheduling that maximizes the value provided by the resource based on multiple, system-wide parameters. Additionally, 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. This is addressed further under the value function discussion.

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 minimal planning. 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.

C. Visibility and coordination across multiple projects

In highly concurrent engineering tasks, the KSS provides a means of synchronizing activities across mutually dependent teams by coordinating their activities through changing value functions (task priority) according to the degree of data completeness and maturity (risk of change). It also provides an excellent way to show where tasks are and the status of work-in-progress and queued or blocked work.

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

The core rationale of most lean and agile approaches is to 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 task value determination, the KSS provides an intentional approach to achieving early value.

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. Those efforts are described in a separate paper [19].

We have concurrently iterated the concept, the models, and the simulations, hoping to determine if the approach modeled in vitro is sufficiently likely to provide the hypothesized benefits in an in vivo implementation. Using the work so far, we will gather additional baseline data to refine and calibrate the models and simulations, and are already discussing instrumented pilots of the approach with a number of companies in the US.

We are also looking to improve the model of the SE services to include negotiation and the other human/social aspects of the processes. We believe this is particularly important in solving issues around implementing more closely coupled systems, software, and stakeholder development collaborations.

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, Pages 12-26 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.