Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 8 (2012) 309 – 314

Procedia Computer Science

New Challenges in Systems Engineering and Architecting
Conference on Systems Engineering Research (CSER)
2012 – St. Louis, MO
Cihan H. Dagli, Editor in Chief
Organized by Missouri University of Science and Technology

Effectiveness of kanban approaches in systems engineering within rapid response environments

Richard Turner[a*], Dan Ingold[b], Jo Ann Lane[b], Ray Madachy[c], David Anderson[d]

[a]*Stevens Institute, Hoboken, NJ, 07030, USA*
[b]*University of Southern California, Los Angeles, CA, 90089, USA*
[c]*Naval Postgraduate School, Monterrey, CA, 93943, USA*
[d]*David J. Anderson Associates, Seattle, WA, 98113, USA*

Abstract

Effective application of systems engineering in rapid response environments has been difficult, particularly those where large, complex brownfield systems or systems of systems exist and are constantly being updated with both short and long term software enhancements. This paper proposes a general case for solving this problem by combining a services approach to systems engineering with a kanban-based scheduling system. It provides the basis for validating the approach with agent-based simulations.

Keywords: Agile systems engineering; lean systems engineering; kanban; rapid response; scheduling; systems engineering management

1. Introduction

Traditional systems engineering (SE) developed half a century ago, primarily driven by the challenges faced in the aerospace and defense industries. In rapid or continuous deployment environments, where requirements are not precise and can change or emerge quickly, SE has often performed poorly, not meeting available schedule and resource bounds [1, 2]. New and flexible methods, processes and tools are required for effective SE in these environments. Engineering principles involving agility and leanness have been adopted to address similar shortcomings in software systems, but integrating these concepts into the SE workflow has proven difficult. To address this issue, we are leveraging earlier research [3, 4, 5, 6, 7], and new experience with lean approaches [8, 9], to investigate the use of flow-based pull scheduling techniques (kanban systems) in rapid response development environments.

1.1. Kanban as a starting place

In manufacturing, a kanban approach provides a visual means of managing the flow within a process. Signal cards are created to the agreed capacity of the process and one card is associated with each piece of

1877-0509 © 2012 Published by Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.procs.2012.01.065

work. Work may be the creation of a part, the integration of a part into an assembly, or any completeable task. Once all cards have been associated, no more work can begin until some piece of work is completed and its card becomes available. An 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, the kanban concept has evolved into a means of smoothing flow by balancing work with resource capability. It focuses more on limiting 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. In this research, we define a visually monitored set of process steps adding value to work units that flow through them. Each step has its own queue and set of resources. The fact that queues are included in the system allows costs of delay and other usually invisible aspects of scheduling to be integral inputs to decision making. The visual representation of the work is critical to kanban success, because it provides immediate understanding of the state of flow through the set of process activities. This transparency makes apparent process delays or resource issues and enables the team to recognize and react immediately to resolve the cause. on. The process is managed through *Work in Progress (WIP)* limits, *small batch siz*es, and *Classes-of-Service* (COS) definitions that prioritize work with respect to value and risk. Flow is measured and tracked through statistical methods that provide insight to tune and improve the system.

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 is 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 can lower the context-switching overhead that impacts individuals or teams attempting to handle several simultaneous tasks; they can accelerate value by completing higher value work before starting lower value work; and, they can provide for reasonable resource work loads over time.

Using *small batch sizes* is a supporting concept to WIP to further limit rework and provide flexibility in scheduling and response to unforeseen change. Smaller batch sizes even out 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.

In the remainder of the paper we will refer to the proposed approach as a *kanban-based scheduling system (KSS)*. While not a true kanban in the manufacturing sense, the characteristics are sufficiently similar to support the name.

1.2. Predicted Benefits of the Proposed Approach

1.2.1. 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.

1.2.2. 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.

1.2.3. 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.

1.2.4. 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.

2. Defining the Approach

In Figures 1 and 2, and Table 1, we define our KSS concept. We intend that this model be recursive at many levels to allow for complex implementations. 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.

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 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.

2.1. Systems Engineering as a Service

In general, systems engineering is involved in three kinds of activities: Up front, continuous, and taskable. While up front activities are critical in greenfield projects, they exist in all system developments. They include creating operational concepts, needs analysis, and architectural definitions. Continuous SE

activities are ongoing, system–level activities (e.g. architecture, risk management). These include maintenance and evolution of long-term, persistent artifacts that support multiple projects. Taskable activities are generally specific to individual projects (e.g. trade studies, interface management), but draw on SE artifacts.

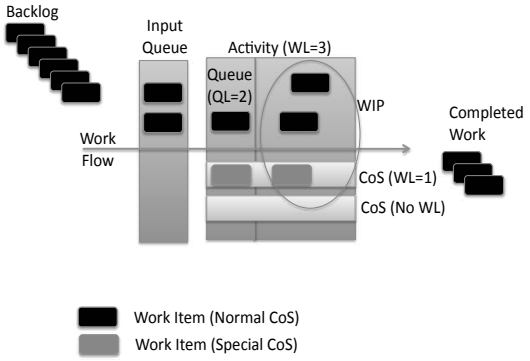

Figure 1. Kanban Scheduling System Model

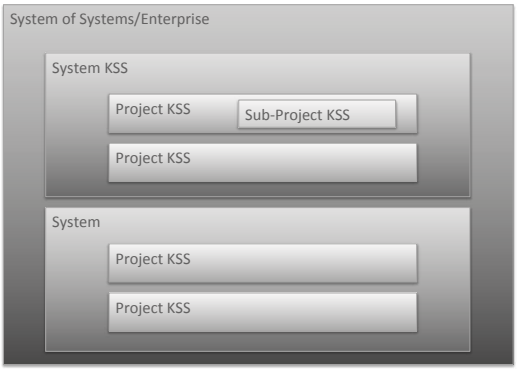

Figure 2. Kanban Scheduling System Hierarchy

Table 1. Kanban Scheduling System Definitions

Work Item	The item controlled in the kanban system; essentially, the kanban carrier
Effort Required	Determines the approximate size of work in person-units of time. May be a negotiated function of desired quality.
Backlog	A non-WIP-limited queue containing work items items awaiting service by the initial activity in a kanban system.
Cadence	The rhythm of the production system. Not necessarily an iteration. Kanban still allows for iterations but decouples prioritization, delivery and cycle time to vary naturally according to the domain and its intrinsic costs. The average transit time of a work item through a kanban system.
Activity	Value-adding work that can be determined as complete. Includes: activity queue, a set of resources, and a WIP Limit. Represents an allocation of the effort required to complete a work item.
Resource	An agent for accomplishing work; may be generic or have specialized expertise. Includes: expertise-productivity pair(s), where productivity is in effort per unit time. Usually associated with a specific activity, but may be shared across activities.
Next Work Item Selection Function	Rule for selecting the next work item from a queue when an activity has less work than its WIP limit; depends on both Class of Service and Value Function, and leads to specific flow behaviors.
Class of Service	Provides a variety of handling options for work items. A CoS may have a corresponding WIP limit for each activity to provide guaranteed access for work of that class. A CoS WIP limit must be less than the activity's overall WIP limit. Examples are expedite, date-certain and normal. CoS may be disruptive (such as expedite) and is the only way to suspend work in progress.
Value Function	Estimates the current value of a work item within a CoS for use in the selection algorithm. Can be simple (null value function would produce FIFO) or a complex, multiple kanban-system, multi-factor method considering shared scarce resources and multiple cost/risk factors. The means of prioritizing work items.
Activity Queue	Holds work items within an Activity that are awaiting processing. The sum of items in process and items in activity queue must be within the WIP limit for each CoS.
WIP Limit	Limit of work items allowed at one time within an activity.
Visible Representation	A common, visual indication of work flow through the activities; Often a columnar display of activities and queues. May be manual or automated. Shows status of all work-in-progress, blocked work, WIP limits. It is a characteristic that provides transparency enabling better management. Difficult to model.
Flow Metrics	Includes cumulative flow charting and average transit (lead) time.

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 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.

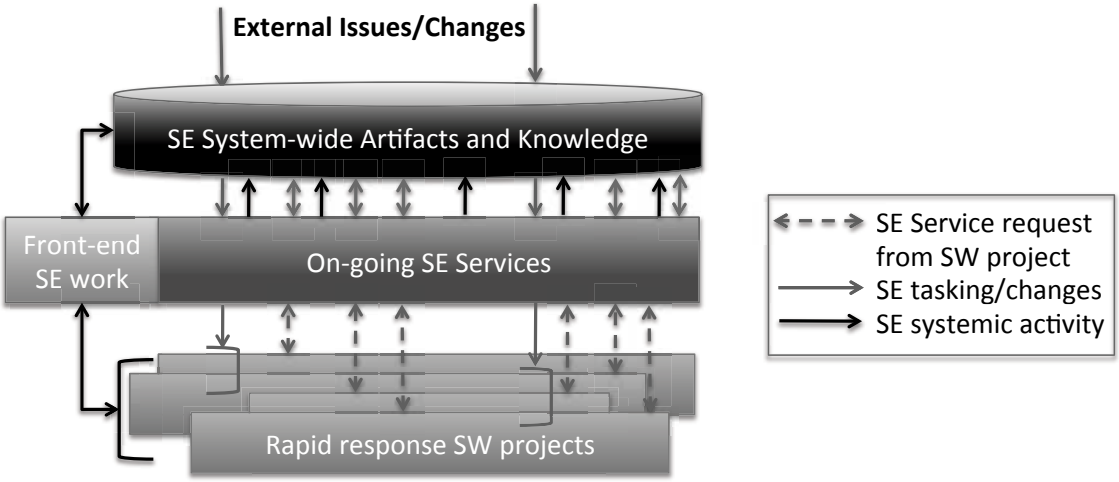

Figure 3 – Overview of SE as a Service concept

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 2 describes categories of services, specific characteristics. A number of services can be defined in each category; such definitions will be part of follow on research as the agent-based validation models [14] are developed.

Table 2 – Systems engineering service categories (Derived from [15])

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

3. Conclusions and ongoing work

In an existing complex system constantly evolving through rapid-response software application development, SE is the glue that holds all of the various projects together. It must be integrated into the various projects without unduly delaying them. Limited SE resources must be efficiently and effectively deployed. We believe our services approach integrates SE into the development cycle, and the kanban-based scheduling maximizes the system value flow from the systems engineering tasks performed. Agent-based simulations are under development to validate the concept.

References

1. NDIA-National Defense Industrial Association (2010). Top Systems Engineering Issues In US Defense Industry. Systems Engineering Division Task Group Report, September, 2010.
2. 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.
3. 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.
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. Boehm, B. and Turner, R. (2004). Balancing Agility and Discipline: A Guide for the Perplexed. Boston: Addison Wesley.
6. Larman C. and Vodde, B. (2009). Scaling Lean & Agile Development. Boston, MA: Addison Wesley.
7. Poppendiek, Mary. (2007). Implementing Lean Software Development: Boston, MA: Addison Wesley.
8. Anderson, David. (2010). Kanban. 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, E.M., and J. Cox. (2004.) The Goal: a Process of Ongoing Improvement. Great Barrington, MA: North River..
13. Boehm, B.: (2009) Applying the Incremental Commitment Model to Brownfield Systems Development, CSER 2009.
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. 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.