Modeling Kanban Processes in Systems Engineering

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 processes using pull scheduling methods (kanban) are being evaluated with hybrid modeling and simulation. We are assessing integrated systems and software engineering at the enterprise level, where rapid response software development projects incrementally evolve capabilities of existing systems and/or systems of systems. A kanban-based scheduling system was defined and implemented with connected discrete, continuous and agent-based models. We are simulating the process performance vs. traditional methods of sharing systems engineering services across projects, and whether the overall value of the systems of systems over time is increased.**

Keywords-systems engineering processes; process modeling and simulation; kanban processes; integrating systems and software engineering; hybrid modeling

I. INTRODUCTION AND BACKGROUND

Engineering principles involving agility and leanness have been adopted to address non-determinism in software systems. They use iterative and spiral concepts, require less traditional ceremony, maintain closer interaction with stakeholders, and are based on best practice, underlying theory and overarching principles [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 evaluates the use of pull scheduling systems (kanban) in combined processes for systems engineering (SE) and software engineering (SWE), where rapid response software development projects incrementally evolve capabilities of existing systems and/or systems of systems. It is hypothesized that such systems could provide more effective integration and use of scarce systems engineering resources, enhance flexibility and predictability over complex master schedules, improve visibility and coordination across multiple projects, lower governance overhead, and achieve higher system-wide value earlier.

We developed a general kanban approach, a specific kanban-based process for supporting SE in rapid response environments, and simulated that process as well as traditional processes to determine if there were gains in effectiveness and value.

A general Kanban-based Scheduling System (KSS) was defined and coupled with a service-oriented approach to systems engineering to develop an approach for integrating multiple related projects with a resource pool of systems engineers. This approach was modeled in vitro using a variety of simulation tools to investigate whether the hypothesized benefits seemed likely to result from an in vivo implementation.

II. THE KANBAN-BASED SCHEDULING SYSTEM

Our concept definition of a KSS is illustrated in Figure 1 and detailed in Table 1. The workflow model is interpreted as recursive at any number of levels to allow for complex implementations.

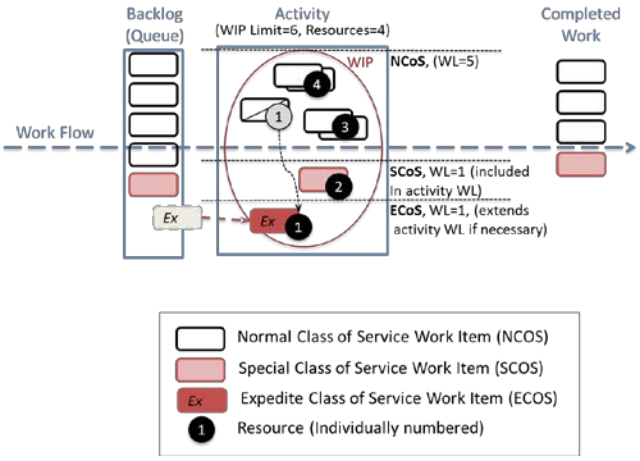


Figure 1. Kanban scheduling system model.

TABLE I. KANBAN SCHEDULING SYSTEM DEFINITIONS

Term	Description
Work Item	The item controlled in the kanban system. A work item had a definition, a Class of Service, and often a rough estimate of work effort required. The value of a work item is determined by a value function.
Effort Required	The approximate size of work in person-units of time. May be a negotiated function of desired quality.
Backlog	An infinite 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 allows for iterations but decouples prioritization, delivery and cycle time to vary naturally. Often 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 (backlog in the initial activitiy), a set of resources, and a Work-In-Progresss (WIP) Limit. Represents an allocation of the effort required to complete a work item.
Resource	An agent for accomplishing work; may be generic or multiple. Productivity can vary by expertise.
Procedure for Selecting Next Work Item	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. May have an allocated WIP limit for each activity to provide guaranteed access for work. CoS may be disruptive and may suspend work in progress. Examples: expedite, date-certain and normal.
Value Function	Estimates the current value of a work item within a CoS for use in the selection algorithm; the means of prioritizing work items. 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. Can vary ovewr time. There may be multiple value functions that return independently established values for each hierarchical layer within the KSS.
Activity Queue	Holds work items within an Activity that are awaiting processing. The items in activity queue are not part of the WIP Limit calculation in this model. Rather, the activity queue acts as a buffer between activities.
WIP Limit	Limit of work items allowed in progress 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. Provides transparency enabling better management.
Flow Metrics	Includes cumulative flow charting and average transit (lead) time.

These concepts were derived from [8] [9] [10] [11] [12] [13], workshops and discussions with collaborators. The displayed tasks in Figure 1 and their parameters coupled with the visual representation of flow have been sufficient thus far to define our modeling, but we may introduce new concepts to enable better communications and synchronization between the various interacting systems.

III. MODELING AND SIMULATION

The goal of the modeling in this research is to verify whether organizing projects as a set of cooperating kanbans results in better project performance. Performance is measured through a value function, and better performance is defined as achieving value along one or more of the following scales, which seem most relevant to the rapid-response environment:

- Shortest-time to useful-value
- Highest-value for a given-time.

The research question is whether value can be improved through a KSS that controls the interaction of a resource-limited systems engineering team with one or more development teams via a service-oriented interface implemented.

Three approaches to modeling were considered for this research:

- System dynamics
- Discrete-event
- Agent-based.

Each of these modeling approaches has advantages for the problem domain and level of abstraction.

Discrete event entities are needed because individual task characteristics are critical in an actual Kanban management scheduling process. The different priorities of the tasks are used for scheduling, and the WIP itself is managed as a discrete quantity. Individual performers are also mapped to tasks and this aspect can be modeled with discrete attributes.

There are also important continuous parameters that drive people behavior (the “agents”), including perception delays, feedback effects, schedule pressure and deadlines, motivation and other management pressures. A combined approach could provide a richer and more holistic perspective with interacting model compartments.

While the behavior of individual agents, and actions that can be taken by objects are pre-specified, system-level behavior may emerge from the interaction of agents with objects and other agents [14] [15], that may be impossible to predict using the other modeling approaches. This aspect of agent-based models is well-suited since the intentional behavior of the human agents in projects is relatively simple and well-known, the emergent systemic results of their interactions in a KSS are not.

We recognize that in different applications any of the modeling paradigms may be more efficient. Agent-based modeling may be less efficient than system dynamics or discrete-event, harder to develop and not a good match for a given problem [15]. In this phase we have found that agent-based modeling has been difficult requiring workarounds. For example, multiple resources working on the same task necessitated extra logic and will probably not scale up with the modeling scenarios.

Combined hybrid modeling is often applicable [15] [16] [17]. For example, in this phase we are using discrete events from the task scheduling to drive continuous flows. Agent-based and/or discrete approaches could be used for event generation.

We are not limiting our modeling approach to a single view, because needs dictate that all approaches are valuable and they can be connected. A comparison of modeling approaches for software processes was described in [16], and we have supplemented this knowledge with agent-based modeling in the context of systems and software engineering.

We are initially employing three standard simulation scenarios to compare processes:

- Common SE: Reduced SE involvement up-front, some involvement through project execution as change traffic, heavy involvement in back-end.
- Traditional SE BDUF: Traditional up-front/back-end SE, with Big Design Up Front (BDUF) delaying the start of development, which results in lower change traffic and defect incidence.
- KSS: Incremental SE, with some design up-front and design continuing throughout development, interacting with projects using service-oriented model.

A. *Discrete Event and Continuous Models and Tools*

The discrete-event and continuous model is implemented in a web-based tool. It is parameterized for users to input the number of tasks, effort per task (deterministic or probabilistic), WIP limit, staffing level and value parameters for the tasks.

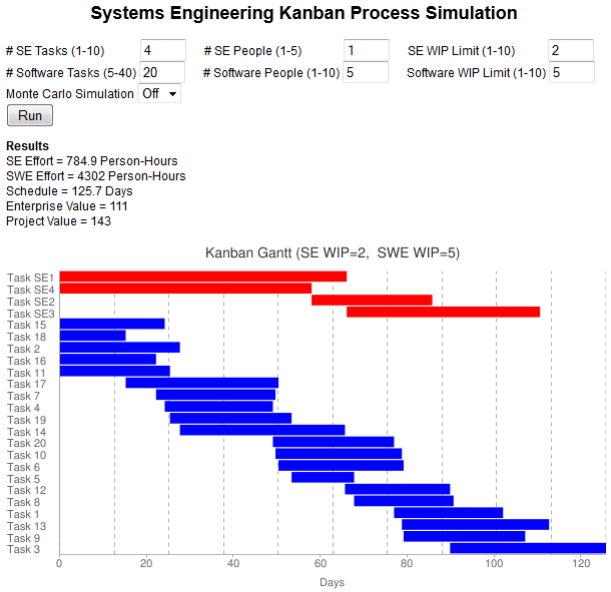
It models the Kanban WIP limit for a variable staff size with non-linear productivity. The non-linearity is due to context-switching losses when resources are split across multiple tasks. It contains two levels of value for the project and organization:

- Project Value – Value of the task towards fulfilling the project objectives (0-10).
- Enterprise Value – Value of the task for the systems engineering enterprise at the organizational level. (0-10).

Continuous flows for tasks and value accumulation are driven by the discrete events. The corresponding rates are pulsed at the event times for task completion and value attainment. The aggregate accumulations are used for continuous quantities such as schedule pressure due to do progress gaps. The continuous parameters are in turn available to the agent-based model compartment for simulating individual agent behaviors (e.g. peoples' delayed perceptions of trends and their reactions).

Sample runs of the DE simulation are shown in Figures 2-5. Figures 2-4 include Gantt charts with tasks down the page and time running horizontally. The current WIP size at any time can be determined by visualizing a vertical line on the Gantt chart and then counting the intersected tasks. The number of tasks concurrently active cannot exceed the WIP limit.

For each run a normal distribution is used to generate task effort, and the duration is calculated using the available staff and WIP size. These figures represent a baseline case of a nominal 90 day project.



SE Task	Project Value	SE Value	Effort (Person-Hours)	Resource(s)	Duration	Start	Finish	Cumulative Value
SE1	9	4	264.1	0.5 SE	66	0	66	107
SE4	5	1	231.5	0.5 SE	57.9	0	57.9	108
SE2	4	2	111.2	0.5 SE	27.8	57.9	85.7	110
SE3	3	1	178.1	0.5 SE	44.5	66	110.5	111

SWE Task	Project Value	SE Value	Effort (Person-Hours)	Resource(s)	Duration	Start	Finish	Cumulative Value
18	10	1	121	1 SWE	15.1	0	15.1	10
16	9	5	176.7	1 SWE	22.1	0	22.1	19
15	10	2	193	1 SWE	24.1	0	24.1	29
11	9	8	202.4	1 SWE	25.3	0	25.3	38
2	10	4	220.4	1 SWE	27.6	0	27.6	48
4	9	7	198.8	1 SWE	24.9	24.1	49	57
7	9	1	220.1	1 SWE	27.5	22.1	49.6	66

Figure 2. Project simulation example (SE WIP=2, SWE WIP=5).

Figure 3 shows a case where 10% of the software tasks require rework. In subsequent models the percent of rework will be directly impacted by systems engineering.

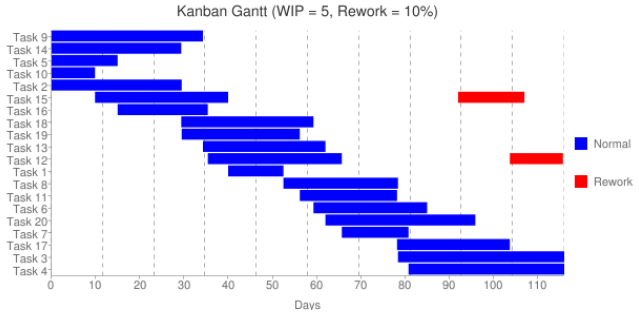


Figure 3. Example project Gantt with rework.

A multiple project scenario at the enterprise level is shown in Figure 4. The top tasks are systems engineering service tasks that support the three software projects below it. The initial tasks numbered 1.0, 2.0 and 3.0 are project initiation tasks. The other tasks are in continuous support of software engineering who has "kicked back" some tasks requiring more work. These tasks are numbered the same in both swim lanes (e.g. Task 2.2, Task 3.2, etc.)

The diagram also displays project priorities among the projects competing for systems engineering support. In this scenario Project 1 in red has higher priority. This is why Task 3.3 in Project 3 has to wait for the Project 1 tasks in red to complete first.

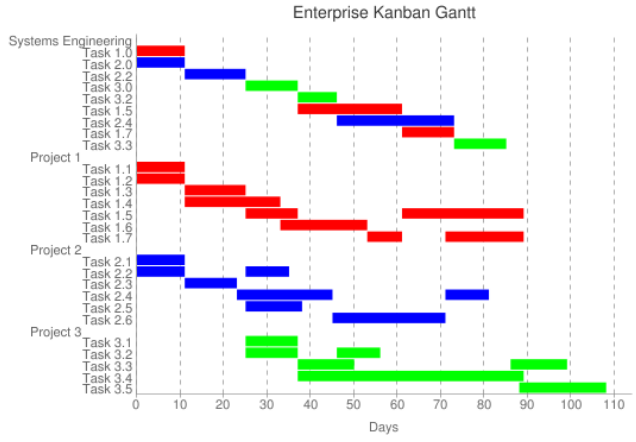


Figure 4. Example enterprise Gantt with SE services.

Monte Carlo simulation can be invoked for multiple runs. Figure 5 shows sample results with output probability distributions for the four primary indicators. This capability will be enhanced in the next phase including normalizing the value outputs.

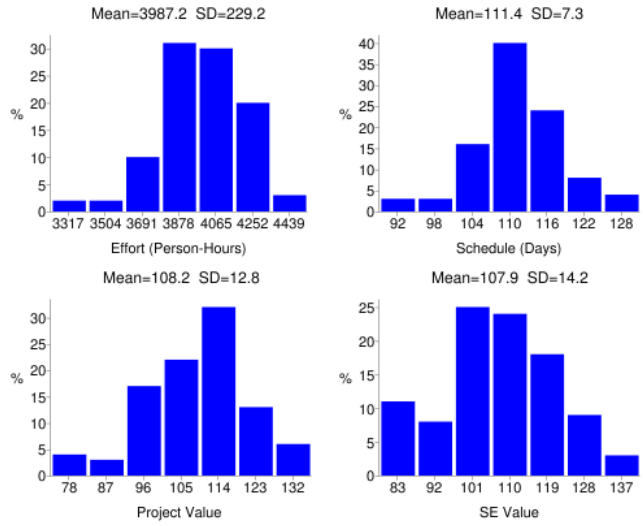


Figure 5. Example Monte Carlo results.

B. *Agent-Based Model and Tool*

The agent-based model workflow is depicted in Figure 6. It diagrams the relationship of the Kanban concepts within the agent-based model for the KSS. The model is composed of one or more kanbans, each of which represents a project or, as will be seen, a pan-project team. Each kanban is composed of a backlog queue, one or more serialized activities, and a release queue. Resources work within an activity, pulling completed work items from the next upstream activity (or incomplete items from the backlog, if the resource is in the first activity of a kanban), and taking some amount of time to complete each work item. The release queue pulls completed work items from the last activity of a kanban, at which point the work item is considered fully complete. The customer is the source of all work items that enter the system, which are pushed onto the backlog of one of the kanbans for processing.

Resources are the human agents whose actions take incomplete work items and transform them, with more or less fidelity and taking varying amounts of time, into completed work items. The activity within which each resource works is constrained to a maximum work-in-process (WIP) limit, and at any point in time each activity contains no more than the WIP-limited number of work items, queued or in-process. Within each activity, some work items are queued awaiting the next available resource, and some are being processed. Work items are assigned an estimated duration and a value function at creation, and move through the system by being pulled from upstream activities into the next downstream activity, or the release queue.

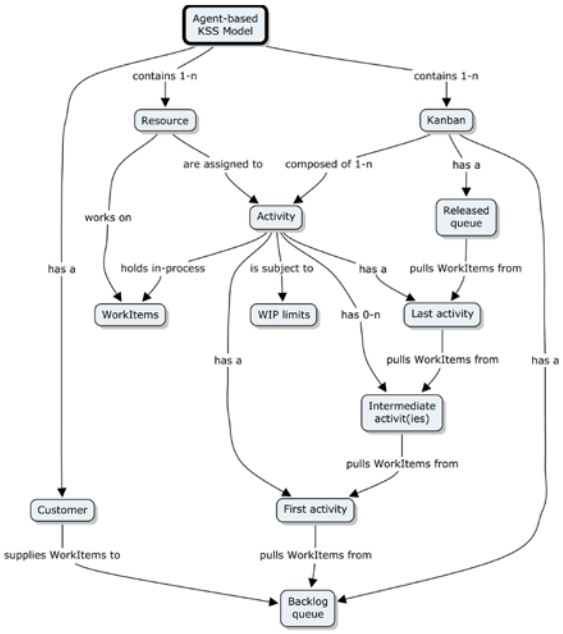


Figure 6. Agent-based model of kanban-based scheduling system.

IV. Summary and Next Steps

This research has developed the fundamental definitions and an initial set of simulation tools to evaluate a new approach to managing systems engineering where rapid response software development projects incrementally evolve capabilities of existing systems and/or systems of systems. It defines and models a kanban-based scheduling system and a services approach to systems engineering among software projects in such an environment.

In developing the simulations it became clear the complexity of the environment and the nature of both kanban scheduling and service-oriented systems engineering dictated a hybrid model with discrete-event, agent-based and continuous components. We have explored interactions between the modeling components as a basis for further integrated simulation capabilities.

We are running statistical experiments against the scenarios to identify the sensitivity of policies and parameters in process outcomes. Examples of current experiments include varying the number of people and WIP size. These impact various effort and schedule tradeoffs due to multitasking overhead. Other experiments vary policies for task prioritization and rules for feature dropping to meet schedule with associated value tradeoffs.

Current simulations make quantitative assumptions for the baseline cases and approximate well-tuned kanban processes executed by proficient practitioners and the results have been in line with expectations.

In order to improve our confidence for in vivo experimentation, we are obtaining better project data to parameterize and calibrate the models for the sponsor's rapid response environment.

The existing work will be expanded by adding team-related components in the KSS definition, and corresponding parametric representations for the key simulation variables. The expansion will include a completed set of SE services including value and quality functions.

We will refine the simulations by including behavioral and team-interaction components. We will add the ability to model specific representation of sponsor teams including multi-level SE authorities, and investigate cross-project effects.

We are integrating the best aspects of each type of model into a toolset which can take actual or estimated data from an organization. It will indicate how using the kanban service-oriented approach could improve their current process performance, and help estimate projects.

The intent is to provide the toolset to a variety of organizations with similar environments to gather additional baseline data to improve the simulations.

The capabilities and insights of the completed and validated simulation toolkit can lead to:

- Increased understanding of the value of various SE services
- Better integration of SE and software engineering through the services concept
- Clarity in the value of SE as a knowledge broker and analysis service in brownfield evolution environments [18].

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.