Integrating Development with Systems Engineering using Lean/Agile Methods

Dan Ingold
6 March 2012
The Problem

System-of-Systems Context

Systems Engineering

System A

System B

System C
General Context

• Rapid-response, incremental-development
• “Acknowledged” system-of-systems
• SE separate from projects, limited control
• Projects mutually exclusive, little/no insight
• SE still needs to coordinate/inform projects
  - Develop high-level architecture goals
  - Coordinate interfaces among projects
  - Inform projects as customer “proxy”
Sponsor Environment

• Rapid-response teams on 90-day “spins,” using agile (Scrum-like) development model

• Claimed utility is to “show progress to customer,” but reality is “because requirements change rapidly”

• Two levels of SE teams: project and enterprise; SE a scarce resource, and largely ignored by developers

• How might SE be better integrated, without wholesale changes to a resistant development community?
Desirable Solution Characteristics

- Relatively “lightweight” (agile) mechanism
- Insight into (limited control of) schedule performance, without dictating schedule
- Affect value decisions dynamically during project execution
- Provide time- vs. quality-tradeoff communication between projects and SE
On-Demand Scheduling System

- “Pull” or “kanban-based scheduling system” (KSS) approach
- Service-oriented systems engineering interface
- Hypotheses:
  - 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
  - Achieve higher system-wide value earlier
KSS Characteristics

- Kanban (signal card) is a form of on-demand scheduling that provides a visual means of managing the flow of work items within a process
- Visual signals synchronize the flow of work, and limit work-in-process (WIP) to the agreed capacity of the process
- In knowledge work, a kanban-based scheduling system (KSS) smooths workflow by balancing work with resource capability
- Work items (tasks) represented by kanban-like “card”
- Work cannot be started until there is an available appropriate resource
- Work is pulled into the activity as capacity is available rather than “pushed” via a schedule
- KSS limits the waste of task switching, minimizes excess inventory or delay due to shortage, and provides a means of tracking work progress
In Figures 1 and 2, and Table 1, we define our concept of a KSS. We intend that this model be recursive at many levels to allow for complex implementations. While we currently believe work items 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.

Figure 1 shows the core concept of the KSS. 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.
KSS Activity Concept

- Work items placed into backlog: needed, but as yet unscheduled
- An activity can accommodate only a limited number of work items at one time (WIP limit)
- When activity below WIP limit, item pulled from backlog based on value
- Work items may also have a class-of-service (CoS) indicating priority, with WIP limits per CoS
- Downstream activities pull completed items from upstream activities
KSS System Hierarchy

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.

The illustration shows a work item with a CoS of expedite coming into the KSS. According to the policies established for this KSS, expedite is allowed to bump up the WIP limit for the activity, but the activity is itself limited to only one expedite CoS work item at a time. The entry of the expedited work item blocks the activity from pulling any additional work items, and causes resource #1 to suspend work on their current work item, thus blocking it as well. In this case, the team felt that resource 1 was sufficient to accomplish the expedited work item, and that allowing the remaining resources to continue their current work items best served the KSS flow. If this turned out to be wrong, adjustments could be made immediately to resolve the imbalance.

In this illustration, the KSS consists of a single activity— and that is generally how the upstream customer would view it. However, 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.
The idea of using an on-demand scheduling system for systems engineering in the rapid development environment is an attempt to merge the SE flow and the software development project flow rather than simply laying SE functions on top of project activities without concern for the rapid-response constraints. Our initial model of such a system-wide KSS that includes both systems and software 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. 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. 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.

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.
KSS Modeling Approaches

- SD models require knowledge of global relationships
- DE models require knowledge of local processing, inter-arrival time
- AB models require knowledge of individual actions, from which global behavior emerges

* (Borshchev & Filippov 2004)
Progress to Date

• Research supported by SERC RT-35, Agile Lean Software Engineering (ALSE) Evaluating Kanban in Systems Engineering

• Developed general model of KSS, systems engineering services, recursive relationship of SE and development (see QE Report)

• Implemented and run initial AB & DE simulation models
  - Creation of work items (tasks)
  - Assignment of resources to activities
  - Performance of resources within activities on work items
  - Scalable components representing overall kanban process
Questions / Comments

AND THAT’S MY LAST SLIDE. ANY COMMENTS?

YOU STOLE AN HOUR OF MY LIFE. SOMETHING INSIDE ME DIED. I WILL NEVER HAVE ANOTHER GOOD DAY.

I WENT IN WITH LOW EXPECTATIONS. THEY CAN’T HURT YOU IF YOU’RE ALREADY DEAD.
Backup Charts
# Seven Wastes in Development

<table>
<thead>
<tr>
<th>Waste</th>
<th>What is it?</th>
<th>PD/SwD Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overproducing</td>
<td>Producing more or earlier than the next process needs</td>
<td>Batching; extra features</td>
</tr>
<tr>
<td>Waiting</td>
<td>Waiting for materials, information, or decisions</td>
<td>Waiting for decisions, work items</td>
</tr>
<tr>
<td>Conveyance</td>
<td>Moving material or information from place to place</td>
<td>Task switching</td>
</tr>
<tr>
<td>Processing</td>
<td>Doing unnecessary processing on a task, or an unnecessary task</td>
<td>Stop-and-go tasks; low-value tasks</td>
</tr>
<tr>
<td>Inventory</td>
<td>A build up of material or information that is not being used</td>
<td>Partially-done work; lower-priority needs</td>
</tr>
<tr>
<td>Motion</td>
<td>Excess motion or activity during task execution</td>
<td>Task interruption; lack of focus</td>
</tr>
<tr>
<td>Correction</td>
<td>Inspection to catch quality problems or fixing an error already made</td>
<td>Defects</td>
</tr>
</tbody>
</table>

*Adapted from (Morgan & Liker 2006, p 72; Poppendieck & Poppendieck 2003)*
Kanban in (Simple) Practice

• Work items represented as placeholder cards (kanbans)

• Columns represent activities

• Activities have WIP limits (numbers at top), related to available resources

• Downstream activities pull completed items from upstream activities, as their own work is pulled by still further downstream activities

* (Anderson & Reinertsen 2010)
Role of Systems Engineering

• Acting as a proxy for the customer
  - Relating to “on-site customer” principle of agile
  - Exhibiting issues of customer: availability, depth of understanding

• Providing project memory across time and teams

• Supporting holistic, system(s)-wide architectural vision

• Partitioning requirements among projects/teams

• Mediating among conflicting development decisions

• Prioritizing project requirements by value
# Systems Engineering Services

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

*Adapted from (Lane & Dahmann 2008)*
Elements of a Kanban Task

- What should be done? (the requirement)
- When should it be done? (the value/time relationship)
- How well should it be done? (the quality/effort relationship)
Agent-based Modeling

- Well-suited for works system design (Sierhuis et al. 2009)
- Software process simulation should represent not just activities, policies, and procedures, but resources, preferences, and cognition of staff, together with functional and social organization (Yilmaz & Phillips 2009)
- Characteristics of ABM (Tolk & Uhrmacher 2009)
  - Autonomy - have control over actions and internal state
  - Social ability - agents interact with other agents
  - Reactivity - agents perceive and respond to environment
  - Proactiveness - agents exhibit goal-directed behavior
KSS Overview

Literature

- Theory of constraints describes constraint-limited system, bottlenecks which are discovered and eliminated, revealing the next constraint (Goldratt 1999)

- Kanban is outgrowth of Toyota lean manufacturing technique to produce intermediate products “just-in-time” using “pull” system (Liker 2003)

- (Poppendieck & Poppendieck 2003) briefly discusses kanban, but distinguishes little from XP story cards

- (Morgan & Liker 2006) discusses lean goals to reduce work-in-process (WIP), inventory, variability; provide visual process controls

- (Reinertsen 2009) provides general queueing theory discussion of WIP-limited product development activities, including kanban

- (Ladas 2009; Anderson & Reinertsen 2010) combine true kanban WIP limits with Scrum-like, but not time-boxed, development concept
References


