Understanding the status of evolutionary capability development in large operational systems is often difficult. Schedules are rarely stable due to

- Size and complexity of capabilities
- Operationally imposed unexpected changes in priorities
- Deep supplier chains and contract structures
- Variety and availability of special engineering resources
- Generally complex nature of the operations.

Lean approaches strive to maximize flow through a process, often by using on-demand (kanban) scheduling techniques. This research evaluates if on-demand scheduling techniques in SE (SE) can provide:

- Better status visibility managing multiple concurrent development projects
- More effective integration and use of scarce SE resources
- Increased project and enterprise value delivered earlier
- More flexibility while retaining predictability
- Less blocking of product team tasks waiting for SE response
- Lower governance overhead.

RT-35 developed a concept for SE as a service and defined a general Kanban-based scheduling system (KSS). We are now describing and simulating the implementation of a KSS Network for a complex, multi-site health care information system.

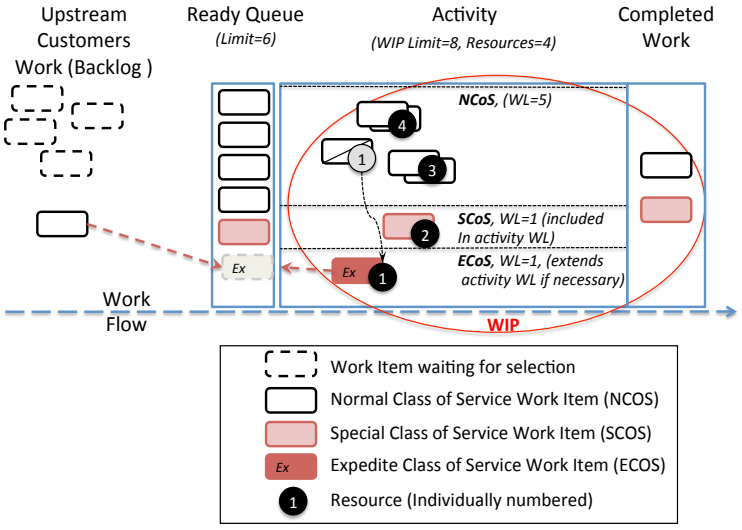

Figure 1. KSS Building Block

The KSS Concept.

A KSS is a means of visually controlling workflow. It consists of a set of activities, where each activity has its own ready queue, a set of resources to add value to work units that flow through it, and a done queue.

Visual representation provides immediate understanding of the state of flow through the set of activities. This transparency makes resource issues and process anomalies (both common and special cause) easily visible, enabling the team to recognize and react immediately to resolve issues locally. Because the team and management interact with the visualization and collectively solve problems, this aspect is important in achieving continuous improvement (kaizen). Control of the KSS is generally maintained through W*ork in Progress (WIP) limits, small batch siz*e, and *Classes-of-Service* definitions that prioritize work with respect to risk.

WIP is partially completed work, equivalent to the manufacturing concept of parts inventory waiting to be processed by a production step. WIP in knowledge work can be roughly associated to the number of work items that have been started and not delivered. *WIP Limits* specifically cap the amount of work assigned to a set of resources. This lowers the context-switching overhead that impacts individuals or teams attempting to handle many simultaneous work items; accelerates useful value by completing work in progress before starting new work; and, provides for reasonable and sustainable resource work loads.

Classes of service (CoSs) provide a variety of handling options for different types of work items and affect the next task selection value of specific items of work for KSSs. They allow the WIP limits to be distributed in such a way that certain types of work will always take priority, will have more consistent access to resources, or will only be selected under certain circumstances.

The fundamental KSS building block is shown in **Figure 1.** 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. A scheduling cadence provides regular meetings of the KSS team to assess 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.

SE as a Service

Defining SE as a service and using on-demand scheduling is designed to better allocate scarce SE resource and integrate the SE flow with the SW development project flow.

In general, SE is involved in three kinds of activities in rapid response environments: lifecycle, continuous, and requested. **Lifecycle activities** are critical in greenfield projects, but are important in all systems and system of systems evolution. They include front-end work like creating operational concepts, defining architectures, and capability and requirement decomposition and allocation, as well as final verification, validation, integration, and deployment activities. **Continuous activities** are ongoing, system–level activities (e.g. architecture analysis, performance analysis, configuration and risk management, and incremental verification, validation and integration). These require regular resources for analysis and the maintenance and evolution of long-term, persistent artifacts that support multiple projects. **Requested activities** are generally specific to either individual projects or capability engineering (e.g. issue triage, trade studies, impact assessments, needs analyses, cost analyses, interface support, and specialty engineering support), and draw on the persistent SE artifacts and knowledge.

By viewing 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 work items within affected projects to ensure that the broader issue is handled in an effective and compatible way. The quality of a service may be pre-specified, specified as a parameter of the service request, or negotiated as a function of typical value sought and time available to provide the service. SE services may be thought of as a single activity, although many activities are complex enough to have their own set of value adding activities and specialized resources.

To support timeliness, SE performs its services in parallel to those activities in the requesting project, prioritizing individual project work across the entire system, and then provides the results to the requestor as soon as available. Selecting the next request to be handled is designed to identify the highest cost of delay among the ready work items in terms of the overall system value. This allows SE to be as effective as possible in providing its services across the enterprise.

Implementing a KSS Network

The initial implementation uses a network of integrated KSSs that are intended to:

- Make work in progress visible at all levels through Dashboards and KSS flow boards
- Monitor organizational capacities at all levels
- Limit WIP to improve value flow (identify resource issues, cause of blocked work)
- Coordinate multiple levels of SE activity
- Communicate progress with respect to senior management goals
- Support analysis and decision making at every level of management
- Make visible status development and deployment of capabilities
- Establish a basis for continuous improvement in a rapidly changing environment

The KSS Network shows the relationships between the SW development tasks and the SE tasks. It also clearly captures the relationships between the SW and SE tasks and the capabilities. Understanding the information needs for decision making, including scheduling and flow monitoring/control, at each level of SE activity or utilization, is a key to a successful KSS design. **Figure 2** shows the conceptual design of the hospital system KSS Network.

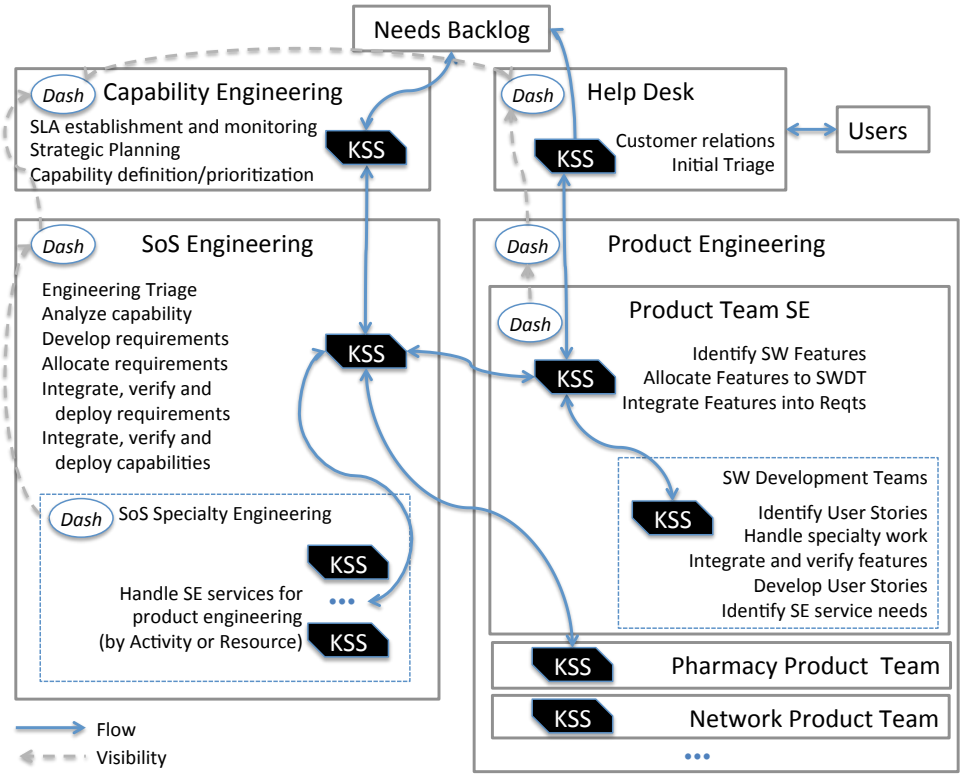

Figure 2. Healthcare KSS Network

SERC RT-35 Research Team:
Richard Turner, PI, Stevens
Ray Madachy, Co-PI, NPS
Barry Boehm, Jo Ann Lane,
Dan Ingold, USC

Industry Working Group: David Anderson (David J. Anderson and Associates), Jabe Bloom (The Library Corporation), Hillel Glazer (Entinex), Curtis Hibbs (Boeing), Suzette Johnson (Northrop Grumman), Larry Maccherone (Rally Development), Don Reinertsen (Reinertsen & Associates), David Rico (Boeing), Garry Roedler (Lockheed Martin), Karl Scotland (Rally Software, UK), Alan Shalloway (NetObjectives), Neil Shirk (Lockheed Martin), Neil Siegel (Northrop Grumman), James Sutton (Jubata Group)