Agile/Lean in Systems Engineering Project

Richard Turner, Stevens Institute, PI
rturner@stevens.edu

SERC Sponsor Research Review
4 December 2014. Washington, DC
Agile/Lean in SE Overview

- Improved agility/efficiency critical to system development/evolution
- Many approaches from agile and lean development communities
- Two major tasks under the SEMT Research Area
- Agile/Lean Enablers for SE (RT-124)
  - Search out and evaluate potential value to SE of adaptive (e.g. agile, lean) methods, processes and tools
- KEVAS (RT-126)
  - Demonstration and research system to support kanban-based scheduling, SEaaS, and other agile/lean management techniques in systems of systems evolution
  - Serves as a basis for simulating promising enablers
- Continues research from MPT and RT-35/35A tasks
Agile/Lean Enablers for SE
[RT-124]

Agile Enablers Team

- Richard Turner, Stevens Institute, PI
- Ye Yang, Stevens Institute
- Forrest Shull, Carnegie Mellon (SEI)

- Student Researchers:
  - Keith Barlow, Stevens Institute
  - Joshua (Jabe) Bloom, Carnegie Mellon
  - Richard Ens, Stevens Institute
  - Dan Ingold, USC
• Seek out lean, agile, and other adaptive techniques potentially applicable to systems engineering
  — Participate in conferences, symposia, and industry working groups
  — Keep track of publications and blogs by thought leaders
  — Create opportunities by leveraging thought leader networks

• Evaluate their potential and identify research strategies
  — Develop an evaluation methodology for potential candidates

• Include technologies identified in earlier work

• Products are white papers and activities
Ongoing work

- **Evaluation process**
  - Adapting process from early SERC MPT work [Turner and Shull]

- **Further investigation of previous approaches**
  - Quantitative schedule acceleration model [Lane, Yang and Ingold]
  - SE as a Service [Barlow]*
  - Evidence-/value-based and collaborative decision processes [Turner and RT-126]*
  - Complexity theory and sensemaking [Bloom]*

- **New approaches**
  - Virtual and physical visible flow indicators [Ens (Dissertation)]
  - Schedule acceleration through crowdsourcing (extends QSAM) [Yang]*

*More detail provided
• Primary researcher: Keith Barlow

• Developing a framework for defining SE services

• Based on INCOSE/IEEE SEBOKwiki.org taxonomy

• Other sources include
  — CMMI (Services and Development)
  — OASIS
  — SOA Reference Model
  — SERC MPT Phase 2 Bridge diagrams
  — RT-35a Final Report

• White paper to be published by 31 December
Complexity theory and sensemaking

• Primary Researcher: Joshua (Jabe) Bloom

• Based on work by Dave Snowden and others
  — Kalawsky; Flach; Juarrero; Sheard & Mostashari; Stevens, Brook, Jackson & Arnold; Luzeaux, Ruault & Wippler; Jones

• Early possibilities for application to SE
  — Differentiating Complexity (Computational, Cognitive, Social, Relational)
    o Define a Sense Making process to understand how different types of complexity are impacting Systems Engineering projects
  — Differentiating Design approaches from Engineering Approaches
    o Align Design and Engineering approaches to Complexity and to each other
    o Apply Systemic Design Research Processes to Systems Engineering
  — Co-Design Approaches for Systems Engineering
  — Actor-Network-Theory application to Systems Engineering

• Initial white paper planned for December-January
Towards Agile Schedule Acceleration Through Crowdsourcing

Ye Yang, Richard Turner
Stevens Institute of Technology
ye.yang@stevens.edu
December 4, 2014

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC) under Contracts H98230-08-D-0171 and HQ0034-13-D-0004. The SERC is a federally funded University Affiliated Research Center (UARC) managed by Stevens Institute of Technology consisting of a collaborative network of over 20 universities. See www.SERCuarc.org.
Definition and Motivation

• Software crowdsourcing (Stol and Fitzgerald, 2014)
  – The accomplishment of specified software development tasks on behalf of an organization by a large and typically undefined group of external people with the requisite specialist knowledge through an open call.

• Characteristics
  – Millions of online developers
  – Software development mini-tasks
  – 2 weeks completion
  – $750 task prize for winning participants
  – Higher quality through broad participation

• Research question
  – How can defense domain benefit from this emerging paradigm?
Growth Trend in Crowdsourcing

- Growth of workers
- Top rated business trend
  — From workforce to crowdsource

Source: \( \text{http://sandfishdesign.co.uk, © 2012, Crowdsourcing, LLC} \)

<table>
<thead>
<tr>
<th>Platform</th>
<th>Crowd size (s/w)</th>
<th>Prize</th>
</tr>
</thead>
<tbody>
<tr>
<td>TopCoder</td>
<td>710K</td>
<td>$81K open</td>
</tr>
<tr>
<td>eLance</td>
<td>600K</td>
<td>$1B total</td>
</tr>
<tr>
<td>TaskCN</td>
<td>3.5M</td>
<td>$6M total, $144k open</td>
</tr>
</tbody>
</table>

Sources: topcoder.com, elance.com, taskcn.com (as of Nov. 2014)
Software Crowdsourcing: State of Art

• Crowdsourced activities
  — Conceptualization
  — Specification
  — UI Prototypes
  — Architecture
  — Component Design
  — Component Development
  — Code
  — Test Scenarios
  — Bug hunt

• Applications
  — Mobile applications
  — Analytics and optimization
  — Scientific algorithm development
  — Online communities
  — Open platforms
  — Digital media
  — Business systems

• Examples
  — 2009 DARPA Network Challenge: $40000
  — 2013 Darpa FANG challenge: million dollar
How TopCoder Works

Top platform for crowdsourced software development

Typical Duration is 19 - 24 Weeks

Build “Factory”
Virtual Global Work Force
TopCoder Community
Competition-based

Requirements
- Specification
- Use Cases
- Activities
- Prototype
- Site Map
- QA Plan

Architecture
- System Diagrams
- Logical ER Model
- Component Model
- Sequence Diagrams
- Application Design
- Component Specs

Component Library

Component Production

Application Assembly

Application Certification

UAT & Deploy
- QA Plan (Traceable Requirements)
- Walk Thru
- Deploy to Production

Credit: www.topcoder.com, © 2007, TopCoder, Inc
Software Development Crowdsourcing Task Profiles

- Task Size
  - #Component spec pages:
  - #Reqt's spec pages:

- Cost and Schedule
  - Amount of prize:
  - Schedule:

- Outcomes
  - #Delivered LOC:
  - DevScore:

- Participation
  - # Registrants:
  - # Submissions:
Evidence on Cost and Schedule Reduction

• Schedule reduction
  — Compared with Parkinson’s law
    ○ “Work expands so as to fill the time available for its completion.”

• Cost reduction
  — Compared with COCOMO
    ○ \( EFFORT = a \times SIZE^b \)
Software Crowdsourcing Tasks are ACCURATELY Predictable!

- 9/29/2003-9/2/2012
- 2859 design and 3015 dev. Tasks
- 980 successful ones used

Define new drivers
- 16 drivers modeling task types, complexity, and participation

Train Models
- 9 learners

Validate Models
- 3 baselines

Our models can predict the cost of crowdsourced tasks within 35% of actuals for 80% of the tasks.
Agile Acceleration through Crowdsourcing

• Where applicable in defense domains?
  — Idea exploration to coping with emerging changes
  — Rapid delivery of non-core features
  — Specific functional or performance upgrades
  — Where problems can be generalized, needing more flexible and competitive broad participation than outsourcing

• Investment
  — Formulate the crowd: within existing DoD suppliers scope, or leveraging on general public crowdsourcing platform
  — Setting up software development environment/platform dedicated for the crowd
  — Dedicated personnel responsible for crowdsourcing assessment, task decomposition, Q&A, and monitoring
From Workforce to Crowdsourcing: a Viable Option to System Engineering

Incremental View of Risk-driven Spiral Model
For large scale software projects

1. Define tasks
2. “Pick-up” teams
3. Communicate and monitor
4. Review crowd deliverables

0. Setup environment
Crowdsourcing Decision Framework: An Elaborated Agile Rebaselining Triage Model

Stabilized Increment-N Development Team

Defer some Increment-N capabilities

Negotiate change disposition

Accept changes

Handle Accepted Increment-N changes

Agile Future-Increment Rebaselining Team

Filter changes

Crowdsourcing?

No

Assess changes

Propose Handling

Yes

Propose Handling

Crowdsourcing?

No

Filter Handling

Propose Handling

Crowdsourcing?

Yes

Plan tasks

Manage tasks

Assemble, consolidate results

Future Increment Managers

Proposed changes

Recommend no action, provide rationale

Discuss, revise, defer, or drop

Change Proposers

Prepare for rebaselined future-increment development

Rebaseline future-increment Foundations packages

Discuss, resolve deferrals to future increments

Recommend deferrals to future increments

Analyze options in context of other changes

Recommend no action, provide rationale
• Crowd labor: an emerging alternative to schedule acceleration

• Crowdsourcing decision framework helps to address key management concerns

• Many new challenges need to be addressed through further investigation
  — Predictive models: cost, schedule, quality, crowd participation and behavior
  — Architecting and optimization approaches: task decomposition and configuration, multi-objective optimization
  — Collaboration techniques: large population, interactions, monitoring and control

• Look forward to feedbacks, discussions, and esp. collaborators!
KSS Evaluation and Analysis System (KEVAS) [RT-126]

KEVAS Team

- Richard Turner, Stevens Institute, PI
- Jo Ann Lane, University of Southern CA
- Levent Yilmaz, Auburn University
- Forrest Shull, Carnegie Mellon (SEI)
- Alice Smith, Auburn University
- Jeff Smith, Auburn University
- **Student Researchers:**
  - Donghuang Li, Auburn University
  - Alexey Tregubov, University of Southern CA
• Impetus for KSSN research (What we heard from the trenches):
  — IMSs are killing me - I can’t make them work. Help?
  — How do I make sure I am delivering high value across the SoS?
  — Where do I stand implementing my capabilities?
  — How do I know how to balance my [organizational|contract] resources?
Targeted Results

- Better visibility and coordination 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
Fundamental Resource: Product Development Flow (Lean)

• Grew out of Toyota Production System success
• Later applied to knowledge work (imperfectly)
• Key source: *Principles of Product Development Flow* (Reinertsen)
• Summary (essence) of 172 principles
  — Take an economic view [economic framework]:
    o Minimize cost of delay
    o Prioritize the work with the most value
  — Variation of flow is inevitable, so anticipate it
    o Actively manage queues; pull rather than push
    o Reduce batch size
    o Accelerate feedback
    o Actively manage work-in-progress
    o Empower teams to decentralize control

Summary adapted from Murray Cantor
Overview of the KSS Concept

A Multi-level Network of Kanban-based Scheduling Systems

- Pull (kanban) scheduling
  - Value-based selection
  - Limited WIP
  - Classes of Service
- SE as a Service
  - Value negotiation
  - Scarce resource-driven
  - Collaborative/Negotiated
- Integrated work and data flow
- Information radiators at all levels
- Appropriate organizational structures
Where we are

• RT-35
  — Researched issues
  — Developed initial components of the concept
  — Studied agile, lean, kanban, and SEaaS fundamentals
  — Attempted simulation development

• RT-35a
  — Continued refinement of concept
  — Developed a semi-populated healthcare system example
  — Aimed at piloting as a means of learning and validating

• RT-126
  — Must validate the concept for any reasonable adoption/transition to take place
  — Piloting deemed not feasible as primary transition strategy
  — New transition strategy based on experimentation using stronger simulation
  — Enable potential pilot organizations to gain understanding of concept benefits based on their own environments
KSSN Experimental Validation and Analysis System (KEVAS) Operational Concept

• Vision

• Outcomes/Benefits

• Results Chain and Risks

• Initial Concept

• Sample Scenario
Vision

• KEVAS is envisioned to be a flexible, web-based modeling and simulation capability to:
  —demonstrate the principles and mechanisms of the KSS and KSS Networks (KSSN)
  —advance the understanding of the KSSN value-based approach
  —evaluate the effectiveness of the KSSN in a variety of organizational environments
  —investigate existing and proposed mechanisms for KSSN implementation
  —provide a learning and evaluation environment to support organizations that are interested in applying the concept to their environment.

• KEVAS will be a core asset in the transition of KSSN concepts and benefits to industry and government
Outcomes/Benefits

• Validate the KSSN concept

• Support a broader understanding of the benefits of KSSN and thus enhance the pool of possible adopters

• Provide potential adopters a means to “try before you buy” via simulations offered on a transition portal

• Accelerate generating transition materials required for interested organizations to conduct successful pilots

• Enable significant empirical experimentation into product development management approaches, particularly value and scheduling strategies, including the work of Reinertsen and others applying lean concepts to knowledge work

• The ultimate benefit is improved visibility and flow through SoS development and evolution organizations, and higher value delivered sooner to SoS customers and users.
Initial Concepts

• Experiments to simulate scheduling and management approaches as applied with a specific work flow and organization

• Experiment components including, but are not limited to:
  — Various simulation engine(s): initially a discrete event-based simulation and an agent-based simulation.
  — Organizational models
  — Scheduling models; these may include traditional, orthodox, agile, lean, KSSN, SE as a Service, and/or other mechanisms for determining schedule and priority
  — Simulation control, dashboard, and measurement options
  — Pre-defined, statistically created, or random work flows
  — Experiment components may be created or selected/adapted from libraries

• Tools
  — Create or edit experiment components
  — Produce graphics/visualizations of the simulation results
  — Database System and associated Data Products
Sample Scenario

• Transition Candidate creates and runs an experiment
  — [Pre-condition] Transition Candidates much have acquired appropriate credentials to access the system
  — Select a simulation engine for their experiment
  — Select or create an organizational model to represent their organization
  — Select or create a scheduling model to evaluate
  — Select, create or import a work flow
  — Select the data to be captured, displays, and controls
  — Run the experiment
  — Generate reports and analyses of the data produced
  — Save the experiment and results in a private database
  — [Optional]: Sanitize the data and add it to the historical database
Proposed KEVAS Architecture

- **Components**
  - Domain-specific Language (DSL)
  - Simulation Engine(s)
  - Output Processor
  - Model Database
  - Web Browser Interface

!![Diagram showing the proposed KEVAS Architecture]!!
Validation

• Two validation targets: The simulations and the KSSN Concept

• How do you validate a simulation, and against what?
  — Industry standard practice is difficult to determine
  — Actual piloting is difficult contractually
  — Data is always hard to come by

• Options being considered at this time
  — Alternative paths for piloting (e.g. USC CSSE Member, help from Rob Flow)
  — Tool vendors
    o Mine their industry performance and sample work flows
    o We have entre to LeanKit, VersionOne, Rally, and Rational
  — Relativity
    o Validate the orthodox/traditional models within the simulation using industry contacts.
    o Define a threshold delta for the KSSN approach to meet for successful concept validation
• Reinertsen provides an economic model as well as a queuing and information theory approach to managing flow

• Comparative measurements and established baselines key to validation

• Deciding what to measure in any development effort is difficult

• Improvement involves evaluating the value of visibility and flow, measured at various points across the SoS and value delivered to the customer

• We are currently looking at capturing a few key measures that should support validation
  — Value delivered over time
  — True organizational capacity (as of the time work enters the organization)
  — Percentage capacity utilization for queues
  — Queue size
  — Size of work items (batch size)
  — Work in progress (at all levels)
  — Estimating cost of adoption and delta cost of use
A Note About Value

• Orthodox methods do not use value the same way as KSSN

• Value must consider cost, but should not be equated with cost

• The various meanings of value mean we need a common measure to compare different mechanisms for value delivery over time while considering costs

• Reinertsen uses “life-cycle profits” as the common unit for his economic model for product development; we are researching a way to use this concept, modified to account for non-profits and government
KEVAS Results Chain

1. **KSSN Models**
   - Models and simulations are suitable
   - Sufficient interest in evolution
   - Validity determination is possible

2. **KSSN Concept Validated**
   - New models, simulations, and mechanisms
   - Improved KSSN concept & mechanisms

3. **Validation**
   - Tools for estimating benefit in a specific organization
   - New models, simulations, and mechanisms

4. **An organization can try out the concept**
   - Simulations easy to use and convincing
   - Cost of adoption is acceptable

5. **Orgs see KSSN as valuable**
   - Adoption support is suitable
   - Improved organizational processes and infrastructure

6. **KSSN adoption**
   - Shortfalls, needs, desires
   - New ideas, benefits

7. **KSSN is seen as valuable in industry and government**
   - Changes and costs are acceptable
   - Orgs trust the simulations
   - Orgs take time to use the simulations

8. **KSSN is seen as valuable in industry and government**
   - Better flow and visibility in developing & sustaining SoS

9. **Request for direction**
   - KSSN Models
   - Validation
   - Data
   - Contribution to achieve benefit
Assumptions (Risks) from Results Chain

• Concept is valid
  — Modeling is tractable and possible
  — Validity determination is possible

• Models and simulations are suitable
  — Simulations and models are valid

• Simulations are easy to use and convincing
  — Organizations will take time to use the simulations
  — Organizations will trust the simulations

• Organizations see KSSN as valuable

• Cost of adoption is acceptable [Changes and costs are acceptable]

• Sufficient interest in evolution of the KSSN Concept

• Adoption support is suitable for organizational use
Other Programmatic Risks

- No ongoing support for hosting the transition web site and the simulation infrastructure
- “Best Practice-Silver bullet syndrome” [success in one context does not necessarily imply universal success]
KSS simulator demonstration

By
Alexey Tregubov
6th Annual SERC Sponsor Research Review
December 4, 2014
Georgetown University
School of Continuing Studies
640 Massachusetts Ave NW,
Washington, DC

www.sercuarc.org
Simulation model

• Discrete event simulation (activity simulation)

• WIs Network (work graph) consists of:
  — WIs (id, status, required effort, completeness) – describe tasks, requirements, capabilities
  — Resources (id, specialties, assigned WI)
  — Teams (is, resources, WIs backlog) – groups of resources

• Scenario consists of triggers:
  — Trigger is an if-then rule:
    o Example: “If wi_x is 50% complete then add new WIs”.
    o Example: “If time is 30 then change value of capability C1”.
  — Triggers describe all relationships between WIs and how they change over time.

• Configuration: resources allocation, prioritization algorithm, ...
Simulation work flow

**Scenario Configuration**
- number of teams,
- number of Wis,
- complexity of dependencies,
- etc.

**Simulation configuration**
- Resources allocation,
- Prioritization algorithm,
- etc.

User input → **Scenario Generator**

User input → **KSS Simulator**

Scenario → Performance indicators → **Simulation Results**
Questions?

• Rich Turner
Backups
Demo: Scenario generator & KSS simulator
Demo: resource configuration

```
New Simulation Configuration for Scenario from_generator

<table>
<thead>
<tr>
<th>TeamNo</th>
<th>sp0</th>
<th>sp1</th>
<th>sp2</th>
<th>sp3</th>
<th>sp4</th>
<th>sp5</th>
<th>sp6</th>
<th>sp7</th>
<th>sp8</th>
<th>sp9</th>
<th>sp10</th>
<th>sp11</th>
<th>sp12</th>
<th>sp13</th>
<th>sp14</th>
<th>sp15</th>
</tr>
</thead>
<tbody>
<tr>
<td>SytEng</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam13</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam14</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam15</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam16</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam7</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam9</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam10</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam11</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam12</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam13</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam14</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam15</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DevTeam16</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop Condition</td>
<td>Time: 200 days</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Work Completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Priorization Algorithm:</td>
<td>KS Scheduling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Random Work Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIFO Work Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FIFO Work Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context Switching Delay:</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name of the configuration:</td>
<td>gen_fifo (conf)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Please wait, Processing...</td>
<td>Run</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Demo: results
Results: complex scenario

Value

Time

KSS
Value-neutral (random selection)
LIFO
Results: complex scenario

Number of Suspended Tasks

| Time | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 |
|------|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| KSS  | 0 | 1 | 2 | 3 | 4 | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 |
| Value-neutral (random selection) | 0 | 1 | 2 | 3 | 4 | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 |
| LIFO | 0 | 1 | 2 | 3 | 4 | 0 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 46 | 48 | 50 | 52 | 54 | 56 | 58 | 60 | 62 | 64 | 66 | 68 | 70 | 72 | 74 | 76 | 78 | 80 |
Results: complex scenario - total time spent

Total schedule (calendar days)

- KSS
- Value-neutral (random selection)
- LIFO
Results: complex scenario - total effort

Total effort (person-days)

- **KSS**: Effort required if there are no interruptions
- **Value-neutral (random selection)**
- **LIFO**
Results: capability completeness

Number of 100% complete capabilities

- KSS
- Value-neutral (random selection)
- LIFO
- FIFO