Principal Investigator: Dr. Richard Turner, Stevens Institute of Technology
Co-Principal Investigator: Dr. Alice Smith, Auburn University

Research Team:
Dr. Jeffrey Smith, Auburn University
Dr. Levent Yilmaz, Auburn University
Dr. Forrest Shull, Software Engineering Institute, Carnegie Mellon University

Student Researchers:
Donghuang Li, Saicharan Reddy Chada, Auburn University
Alexey Tregubov, University of Southern California

Sponsor: DASD(SE)
Copyright © 2015 Stevens Institute of Technology

The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Office of the Assistant Secretary of Defense for Research and Engineering (ASD(R&E)) under Contract HQ0034-13-D-0004.

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense nor ASD(R&E).

No Warranty.

This Stevens Institute of Technology and Systems Engineering Research Center Material is furnished on an “as-is” basis. Stevens Institute of Technology makes no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. Stevens Institute of Technology does not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

This material has been approved for public release and unlimited distribution.
Executive Summary

Goals of this research:
Develop the Demonstration and Analysis Tool for Agile SE Management (DATASEM)—a flexible, web-based modeling and simulation capability to

1. Enable realistic experiments to understand how governance models, organizational structures and work flows interact across a system of systems
   1) Investigate new approaches in a variety of acquisition and development environments
   2) Understand implications of mixed governance models across organizations
   3) A possible means of measuring the value of systems engineering
2. Provide a framework to calibrate assumptions of performance
   1) Build data for increasingly sophisticated experiments
   2) Resource capabilities, team maturity, turnover
   3) Overhead costs, complexity of work, rate of environmental change
3. Integrated experiment generation tools that provide the user with
   1) A selection of experimental patterns for organizations and governance models
   2) Sophisticated stochastic support for generating large work item networks
   3) Comparison to other similar experiments
   4) Graphic demonstration of benefit/cost to convince risk-averse decision makers

DATASEM is intended as an initial instantiation of an evolving and expanding set of integrated tools to support research and transition.

Results:

1. An architecture for DATASEM was developed and evolved
2. A Domain-Specific Language for formalizing the concepts represented in the simulation
3. Software was developed, packaged and delivered for standalone and web-hosted systems
4. Five SERC Technical Reports were published to support DATASEM use and transition:
5. Two conference papers were produced

Next Steps:

• Continue this research with the following goals:
  o Enhance the user interface for both standalone and web-based versions
  o Partner with industry to validate and evolve the DSL and the simulation capability
  o Extend the mechanisms to include social and communication aspects
  o Enhance the output products to better illustrate particular aspects

Once released, the software and all ancillary documentation will be available to anyone on the SERC website (www.sercuarc.org). Contact any of the authors if you have questions or would like to participate in the research.
Contents

Executive Summary ................................................................................................................. 3

Contents ................................................................................................................................. 4

Figures ................................................................................................................................. 5

Tables ................................................................................................................................. 5

1 Introduction .......................................................................................................................... 6
   1.1 Background ................................................................................................................... 6
   1.2 Objectives ................................................................................................................... 7
   1.3 Relationship to Previous SERC Research ................................................................. 7
   1.4 Research Goals .......................................................................................................... 7

2 Summary of Work Performed ............................................................................................ 9
   2.1 Concept Development ................................................................................................. 9
   2.2 DATASEM Domain-Specific Language Development (DDSL) ..................................... 10
   2.3 Simulation Architecture and Development .............................................................. 12
   2.4 User Interface and Results Presentation .................................................................... 14
   2.5 Documentation Development .................................................................................... 14

3 Products of the Research .................................................................................................. 15
   3.1 Contract Deliverables ................................................................................................ 15
   3.2 Domain-Specific Language ....................................................................................... 15
   3.3 Software ..................................................................................................................... 15
   3.4 Research, Transition Support, and Piloting Materials ................................................. 16
   3.5 Other Reports and Publications Developed Related to This Work ............................ 16

4 Conclusions and Next Steps .............................................................................................. 17
   4.1 Significant Lessons learned during the DATASEM Development ............................. 17
   4.2 The Impact of the DATASEM Project and Related Research .................................... 19
   4.3 Future research opportunities ..................................................................................... 20

5 References .......................................................................................................................... 21

Appendix A. An Overview of the DATASEM Domain Specific Language (DDSL) .................. 24
   User Library Components ............................................................................................... 24
   Experiment Components ................................................................................................. 29
   Miscellaneous ................................................................................................................... 35

Appendix B. An Example Experiment: Multiple Defense Products .................................... 36
   Description of the experiment .......................................................................................... 36
   The DSL Created for the Experiment ............................................................................... 39
   Experiment Results .......................................................................................................... 45

Appendix C. Framework for DATASEM Validation ............................................................... 47
Figures

Figure 1. Overall DATASEM Flow .................................................................9
Figure 2. Architecture 2.0 .............................................................................13
Figure 3. Multiple Defense Products Overview .............................................37
Figure 4. Workforce Simulated ....................................................................37
Figure 5. Multiple Defense Products Organization .......................................38
Figure 6. General Time and Efficiency Charts ............................................45
Figure 7. Main Effects Plots .........................................................................46
Figure 8. Interaction Plots ............................................................................46

Tables

Table 1. DSL Model Format..........................................................................11
1 INTRODUCTION

1.1 BACKGROUND

Developing, creating or evolving systems of systems (SOSs) present significant systems engineering and management problems. Dahmann and Baldwin characterize these problems as stakeholder involvement, governance, operational focus, acquisition, test and evaluation, boundaries and interfaces, and performance and behavior [1]. All systems face some of these problems, but the uniqueness of the dynamics and resulting communication issues in a SoS require a significant ability for adaptation within the system development community, as well as among the stakeholders. The principles for addressing these issues are no different from those required for any good systems engineering and development activity [2]. Implementation of those principles in SoS environments, however, is a much thornier problem.

Agile and lean philosophies have shown to be effective in supporting adaptation within development and evolution [3], [4], [5]. Complicated, large systems of systems in rapid or continuous deployment environments, where requirements are not precise and can change or emerge quickly, find traditional approaches inadequate.

In 2011, the Systems Engineering Research Center began to investigate alternative management and governance approaches for these complex environments, including a concept for an integrated multi-level network of pull scheduling systems based on explicit, transparent, and continuously updated value of work [6], [7], [8]. This Kanban-based Scheduling System Network (KSSN) concept was developed based on the following capabilities:

• Coordinate multiple levels of development activity across multiple system components with diverse and possibly disjoint or isolated development groups
• Support analysis and decision making at every level
• Flexibly schedule work considering value across the system of systems
• Balance work in progress (WIP) across resources with SoS organizational capacity to improve flow
• Make visible to all levels progress toward capability development and deployment
• Establish a basis for continuous improvement in a rapidly changing environment

Difficulties in validating this concept in vivo led to the decision to create a broad simulation environment that would allow in vitro experimentation with KSSN, but also be applicable to studying other mechanisms, singly and in concert, operating in a range of organizational structures (including all four types of systems of systems identified in [1]) and handling different kinds, durations, complexity, and volumes of work flow. We believe that establishing statistically significant evidence across various combinations of mechanisms, organizations and work flows, as well as providing a suitable simulation “sandbox” for adopters to perform their own experiments will provide a level of confidence that in vivo experimentation (piloting) is low risk and provides value to adopters.
1.2 Objectives

1.2.1 Develop the Demonstration and Analysis Tool for Agile SE Management (DATASEM)

The Demonstration and Analysis Tool for Agile SE Management (DATASEM) is a flexible, web-based modeling and simulation capability to advance the understanding of the KSSN value-based concepts, to investigate optional mechanisms for implementation, and provide support for organizations that are interested in piloting the concept. DATASEM will support broader, in-vitro experimentation required to provide comparative information across a broad set of implementation architectures and organizations as well as store information from in vivo pilots. Additionally, it will graphically demonstrate the key concepts of these adaptive management approaches to interested organizations.

1.3 Relationship to Previous SERC Research

This research builds upon previous findings from three earlier SERC research tasks:

- MPT, Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs, derived an initial methodology for evaluating software-related MPTs that might be applicable in systems engineering through surveys and literature searches.
- RT-35/35A, Agile-Lean Software Engineering (ALSE) Evaluating Kanban in SE, focused on using pull scheduling techniques to determine the applicability of Kanban scheduling to systems and software engineering in a rapid response environment. It also introduced the possibility of systems engineering as a service.
- RT-124, Agile Enablers and Quantification, identified and evaluated potential mechanisms that might be worthwhile to simulate with DATASEM.

1.4 Research Goals

The overall Agile SE Management Project research goals are to:

1. Identify agile, lean, and other adaptive processes and governance mechanisms to help systems engineers
   a. Identify, analyze and quickly react to issues in an environment of accelerating change
   b. Keep pace with evolving requirements, risks and opportunities throughout the extended development lifecycle
   c. Understand and manage the changing economic and political factors that undergird and enable system development
   d. Broaden SE influence and holistically approach complications from increasing
      i. Creation and evolution of systems of systems
      ii. Interoperability between legacy and new capabilities
      iii. Reductionism resulting in point solutions or locally optimized decisions
2. Provide a modeling environment to validate and experiment with adaptive mechanisms, their interactions with more traditional SE, and how they can balance adaptability with discipline in a broad variety of environments.

3. Inform organizations contemplating changes to their system development processes or working in system of systems environments where there are a number of different development approaches being applied concurrently.

Specific goals for this task were:

Develop the Demonstration and Analysis Tool for Agile SE Management (DATASEM)—a flexible, web-based modeling and simulation capability to

1. Enable realistic experiments to understand how governance models, organizational structures and work flows interact across a system of systems
   1) Investigate new approaches in a variety of acquisition and development environments
   2) Understand implications of mixed governance models across organizations
   3) A possible means of measuring the value of systems engineering

2. Provide a framework to calibrate assumptions of performance
   1) Build data for increasingly sophisticated experiments
   2) Resource capabilities, team maturity, turnover
   3) Overhead costs, complexity of work, rate of environmental change

3. Integrated experiment generation tools that provide the user with
   1) A selection of experimental patterns for organizations and governance models
   2) Sophisticated stochastic support for generating large work item networks
   3) Comparison to other similar experiments
   4) Graphic demonstration of benefit/cost to convince risk-averse decision makers

DATASEM is intended as an initial instantiation of an evolving and expanding set of integrated tools to support research and transition.
2 SUMMARY OF WORK PERFORMED

The work was performed over the 15-month period between 29 September 2014 and 22 December 2015.

2.1 CONCEPT DEVELOPMENT

The initial concept development grew out of the results of the RT-35/35a work. The idea was to create a means of modeling organizations, workflows and governance mechanisms and then simulating how they interacted with each other. The decision to use an agent-based simulation was made based on the desire for flexibility describing people responsible for doing managerial, technical and analysis work, making decisions about value, services, accepting or assigning work, negotiating, and other human activities. Although it required some additional overhead, and had significantly more power and functionality than we initially needed in the simulation, RePast was the open source product chosen because one of the researchers had a successful experience with using it. Somewhat later, it was discovered that RePast lacked a web-based interface, which led to unexpected work and limited the capabilities of the initial web-based version delivered in this task. Figure 1 provides an overview of the general flow of the DATASEM suite.
2.2 DATASEM DOMAIN-SPECIFIC LANGUAGE DEVELOPMENT (DDSL)

Early in this tool development task, there was a desire to provide a more formal description of the models that we wished to simulate. This would provide both transparency in implementation and an effective framework for expansion.

A DSL is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain. [9]. Examples of well-known DSLs include HTML, Verilog, Mathematica and SQL.

By using a DSL, a domain expert can focus attention on creating and experimenting with a model. The modeler is not distracted by the essentially algorithmic problems of realizing a model in a low level programming language, and does not risk being misled by errors in the implementation. In agent-based models, interaction and coordination mechanisms among the agents are subtle, but nonetheless difficult, parts of a simulation program where intended model and actual simulation often diverge. Model developers are likely to realize varying interpretations of intuitively appealing mechanisms, and consequently follow diverse development guidelines. This is a serious difficulty for end-users that need to modify the simulation to address new, but related, questions, and leads to high costs, unforeseen and possibly undetected errors, and other substantial problems [10].

Our DSL-based solution mitigates these problems by automating the process of synthesizing a simulation from DSL-specified components. This ensures that the simulation is based on well-defined (in terms of syntax and semantics) concepts. DDSL is implemented through software—compiled into XML and executed by java code using RePast libraries as an agent-based framework. This provides extensibility of the simulation by adding different mechanisms within the DSLs without requiring coding on the part of the user. These formal representations will support graphical interfaces that allow general users to easily construct sophisticated experiments.

Originally planned as three separate DSLs, one for each aspect – work flow, organization and governance – DDSL evolved into a single language that also captured the intentions and mechanisms for the experiment and so is a complete representation in a single artifact. The DSL took much more time to develop than initially thought, and is still evolving. However, it has had a significant impact in how the team approached the software development. A summary of the current DSL appears in Appendix A.

The DSL describes the entire experiment in one model. The basic format of a complete DSL experiment is shown in Table 1.
Table 1. DSL Model Format

<table>
<thead>
<tr>
<th>Section</th>
<th>Subsections</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model Builder</td>
<td>None</td>
<td>Name of the model (not necessarily the filename). This is reserved for later use.</td>
</tr>
<tr>
<td>User Libraries</td>
<td>This section defines basic model components that are referenced later in creating the experiment.</td>
<td>Service Provider Types This describes the general Service Providers that can be defined in the model. These are generally simply to identify the type of organization (for example a prime developer team, a development contractor, or a specialty service organization. These can be organized in some form of hierarchy, but all providers of the same type would be assigned the same hierarchical level. Service Providers are essentially the organizational building blocks. They can be complex or simple. In some cases, such as an expert that you want to model as a service, an a service provider could be a single individual. Work Item Types This describes the kinds of work addressed by the various services. For example, a development task, an analysis task or a problem resolution task could all be types of work items. These definitions also describe hierarchical relationships such as decomposition and associated value inheritance. For example, a ‘Capability’ work item defined as the highest level, could be decomposed into ‘Requirements’ work items (each requirement might inherit some percentage of the Capability’s value), which might be decomposed into Component or Feature work items (again perhaps inheriting some value), and so forth down to the smallest task modeled. The level, distribution and weight of such relationships are described in attributes associated with each Work Item Type.</td>
</tr>
<tr>
<td>Services</td>
<td>This section defines in detail the specific pieces of the model and how they interact with each other. It also provides information for how the experiment is to be run and what outputs are desired.</td>
<td>Each service, for example a development activity, required for completing the work items or governance mechanisms are named and described. These will be used to characterize skills for organizational resources. Governance Strategies Each strategy is identified by type (currently pull or push), the specific mechanisms implemented, and the attributes associated with each of the mechanisms.</td>
</tr>
<tr>
<td>Experiment Model</td>
<td></td>
<td>These are used as references within the Organizational and Work Item models to provide constants or multiple types of stochastic values/distributions.</td>
</tr>
<tr>
<td>Variables</td>
<td></td>
<td>The Organizational Model defines each of the specific Service Providers in terms of a name, Service Provider Type, Governance Strategy (and any governance attributes for this organizational component that are different from the default definition), internal resources (staff or other assets) and other service providers that may provide services to them. Staff/asset resources are characterized by a number of attributes,</td>
</tr>
</tbody>
</table>
### 2.3 Simulation Architecture and Development

Two versions of the DATSEM Software have been developed. One is a web-based implementation and the second is a standalone version. Both are represented in the overall architecture of the initial DATASEM Suite, is shown in Figure 1.

The Web application of the DATASEM consists of three major components (groups of modules):

1. DES Framework web application, which provides UI and connects all other external components. This web application was originally developed as part of the KSS simulator (later DES Framework). This component includes the following components:
   a. Web front end (UI) and corresponding back end modules for UI
   b. Database of DSL models
   c. Database of experiments
   d. Database of simulation results
   e. Experiment builders – these include adaptors for DSL code compiler (Extext Modeler)
   f. Repast adapters – modules that execute Repast simulation engine and collect/convert simulation results (orange modules on the diagram).
   g. DES simulation engine – not used in DATASEM
2. Repast simulation engine – a simulation engine that implements governance mechanisms in Repast Simphony simulation framework. This module is an independent application, which could be used as a standalone application.

3. Exttext Modeler – a compiler for DSL code.

The standalone version of the DATASEM application consists of Repast simulation engine and Exttext Modeler packed together in one package. The standalone version of the DATASEM does not have embedded DSL editor, so to edit DSL files user can either use any text editor or set up Eclipse IDE with Exttext/Extend plugins.

Additionally, the standalone version of the DATASEM can be installed in developers’ mode in Eclipse IDE. This installation includes two projects (Repast simulation engine and Exttext/Extend Modeler) in Eclipse IDE. From Eclipse IDE users can

1. Edit and compile DSL as well as change its grammar
2. Change Java implementation of the agents and governance mechanisms

Figure 2. Architecture 2.0
2.4 User Interface and Results Presentation

The user interface is currently fairly limited. Several designs were developed, but the continuing evolution of the models and the implementation of the DSL tool precedence over the user interface. Outputs are also minimal, although full .xls and .csv files of the simulation results provide for external tools to be added. The user interface and output formats are also at the top of the list for next steps.

2.5 Documentation Development

The last weeks of the task were spent developing the user and software documentation, the DSL documentation, implementation instructions for both the standalone and web-based version, and this technical report.
3 PRODUCTS OF THE RESEARCH

The RT-126 products are based on the specific goals of the research and include contractual deliverables, technical reports, and software in several forms.

3.1 CONTRACT DELIVERABLES

Per the statement of work, we delivered:

A005, Funds & Labor Expenditure Report
A009 Technical and Management Work Plan (July 7, 2014)
A010, Contractor Roster
A013, Final technical report summarizing the research findings (This Document)

Additional SERC Technical Reports developed under this task are listed in Sections 3.4 and 3.5.

3.2 DOMAIN-SPECIFIC LANGUAGE

The DATASEM DSL was developed to provide rigor to the models created with DATASEM, and is the fundamental medium for specifying the organizational, work flow, governance strategy, governance mechanisms, and experimental parameters for simulation. An overview of the current DSL can be found in Appendix A of this document. The full description of the language is captured in the evolving software and in the DATASEM Domain Specific Language Reference. Additional background and rationale can be found in the SERC Technical Report Taxonomy of Adaptive SE Governance Mechanisms.

3.3 SOFTWARE

A significant amount of software was developed or integrated into the delivered DATASEM Suite. The software and falls into 6 component categories:

- DSL Editor
- DSL Compiler
- DSL Instantiation
- Simulation
- User Interface
- Results display

The software is available in source code and two executable formats: Via web browser and as a standalone downloadable package. The package contains all of the custom and open source software in executable formats with instructions on installation. Web access to DATASEM in its initial form is currently available for use (without any warranties) on two servers:

http://datasem.eng.auburn.edu/
http://csse.usc.edu:9000/
3.4 Research, Transition Support, and Piloting Materials

The ancillary information for the DATASEM Suite is provided in Four SERC Technical Reports:

- **DATASEM Installation Guide**: Instructions on how to install the DATASEM Suite on your personal computer or a Server. SERC 2015-TR-111A
- **DATASEM Domain Specific Language Reference**: A complete description of the DSL and its use. SERC 2015-TR-111B
- **DATASEM Users Guide**: Initial manual for using DATASEM that covers both web-enabled and standalone operation. Includes an overview of the DATASEM and concepts, instructions on how to create a new or edit an existing experiment, information on the architecture and a brief discussion on creating new mechanisms. SERC 2015-TR-111C
- **DATASEM Introduction and Transition Briefing**: A Microsoft PowerPoint presentation that introduces the DATASEM concept. This will be the basis of conference presentations, information briefings to interested organizations, and the establishment of a DATASEM user and researcher community. SERC 2015-TR-111D

Once released (probably January 2016), the software and all ancillary documentation will be available to anyone on the SERC website (www.sercuarc.org). Contact any of the authors if you have questions or would like to participate in the research.

3.5 Other Reports and Publications Developed Related to This Work

3.5.1 Other SERC Technical Report


3.5.2 Non-SERC Publications

4 CONCLUSIONS AND NEXT STEPS

This section describes lessons learned, evaluation of status, and potential for DATASEM-related work in the future.

4.1 SIGNIFICANT LESSONS LEARNED DURING THE DATASEM DEVELOPMENT

Given the work accomplished in RT-35/35A, there was considerable overconfidence and a general underestimation of this task’s difficulty. While some was due to the normal ramp up of new researchers and considering new concepts (such as creating a DSL) that would make the tool more easily extensible, there were some humbling but enlightening lessons learned.

4.1.1 IMPACT OF THE INTERACTION OF GOVERNANCE STRATEGIES, ORGANIZATIONAL MODELS AND WORK FLOW DEFINITION IN BUILDING EXPERIMENTS

When initially considering the simulation suite, we believed that the three activities we were interested in modeling were essentially independent and that we could easily mix and match different types. As we actually began to develop the DSL and the implementation of the various mechanisms, it became clear this was not the case.

For example, the technical process generally defines how work is partitioned from higher levels of abstraction to lower implementation levels. This drives both the batch size in terms of scheduling, and the number of resources applied to work items. In the case of a Scrum-like technical management process, there is a significant effort to fit work into the sprint length. In kanban systems, some attempt is usually made to normalize the size of work to some extent. On the other hand, in a traditional development with significant sub-contracting or more specialty teams, work may packaged in larger batches for ease of contracting. The variation in batch sizes and abstractions makes value determination much more difficult and comparisons less convincing.

The technical process also impacts how a work item’s status towards completion is determined. Many technical processes are iterative and so may cycle through a number of intermediate versions of a task. The indeterminate number of cycles are more difficult to represent in a work item network. They may also complicate value determination.

The same governance strategy may require very different mechanisms depending on the organizational definition. Contractual boundaries require more ceremony, and in the case of organizations who adopt some form of bidding process for work, the idea of pull scheduling takes on a completely different flavor than that associated with a software development team or of an implementation of systems engineering as a service.

The experimental design and output specification also had interactions with the models and the way they needed to be specified in order to gather comparable data. This was particularly true if there were significant differences in the resulting work item network.

We discovered that the order in which decisions on organization and governance structure are made can affect the ability to create a reasonable work item network. This changed our original
approach to user interaction with the system and exacerbated a number of design activities, including those in the following sections.

### 4.1.2 DIFFICULTY IN RIGOROUS DEFINITION OF GOVERNANCE STRATEGIES AND MECHANISMS

One of the most frustrating realizations was the difficulty in concretely describing the various governance strategies and mechanisms. The diversity of researcher backgrounds, the unexpected vagueness of the concepts, and the distribution of the team in three areas of the country added to an already challenging situation. The team believed that the material developed in the previous research was sufficiently well-defined to create the simulation. Unfortunately, as the scope of DATASEM expanded from a specific governance strategy to a more general sandbox for experimentation, we realized that our internal definitions and understanding were not easily captured in more general ways.

It was not until the task’s last few months that it became obvious the team was in many ways reenacting the story of the blind men and the elephant\(^1\). Each of us had established a mental model to work with, but we were all approaching different parts of the problem and proceeding from different points of view. These disconnects in understanding each other’s mental models became much clearer as we attempted to build DSL examples of the KSSN concept. Flow control mechanisms such as pull scheduling, classes of service and limits on work in progress were not implemented in a way that represented the earlier work. What had been implemented, however, represented significant mechanisms that had not been explored earlier. One example is the use of the Contract Net Protocol as a basis for pull scheduling. While it does not capture some of the flow control concepts of kanban, it does provide a terrific foundation for modeling negotiation.

Because the discussions around interpretations happened late in the development process, we were not able to adjust the DSL or implementations to do a full model of KSSN for this release. We have, however, gained a great deal of insight into the issues around these mechanisms, and are seeking ways to unify our differing visions to create a broader, more useful DATASEM implementation.

### 4.1.3 TENSION BETWEEN ELEGANCE, EXTENDIBILITY, AND PRACTICABILITY OF THE DSL

The initial DSL was very elegant, but provided little concrete concepts to work with. This was thought to be reasonable given that we wanted an extensible model and elegance generally allows evolution to be relatively painless. However, as we learned more about the intricacies of the ideas we were going to investigate, more and more of the detail was ending up in the implementing java code than in the DSL. We felt that the need for systems engineers to be able to understand what they were modeling needed more explicit expression in the DSL.

---

\(^1\) In this ancient Indian parable, a group of blind men encounter an elephant. They wish to describe it, but each only touches one part of the animal. The man who grasped the tail said, “It is much like a rope.” The man who touched the elephant’s side said, “No, it is more like a wall!” The third man, having grasped the elephant’s trunk replied, “You are both wrong. It is obviously much more like a snake.” “No, not at all,” said the fourth, who had found the elephant’s leg. “It is quite surely like a tree.” And so, unable to resolve their differences, they went off—none the wiser about the elephant, but each thinking they understood it better than the others.
We learned fairly quickly, though, that implementing more detail in the DSL could not be done without looking back at the whole language. As we complete this initial development task, we are aware of a number of shortcomings in the DSL in terms of consistency across definitions and in the sense of structure and design of the language. Reviewing and revising the DSL is light of this first round of development is a high priority for the next phase.

4.1.4 Complexity of Interactions between Organizations

One of the uses that has been identified for DATASEM is investigating the human and managerial aspects of systems of systems. The use of an agent-based simulation engine was driven primarily by this desire to model the human components of the system. Developing the DSL proved hard for just the mechanical complexity of the governance, organizational and work flow representations. We readily identified places where behavior modeling would be valuable, but we were not able to address it and still deliver something useful within the period of performance. Areas where we believe that behavior modeling can be added include:

- Negotiation of services and contracts
- Interface of pull and push governance mechanisms
- Value determination and agreement
- Conflicts due to differing value systems and stakeholders between the system of systems level and the constituent system level
- Stakeholder/constituent behavior

4.1.5 Limitations and Complications of Agent-Based Simulation

In retrospect, RePast provided much more sophisticated simulation capabilities at the cost of not providing more useful capabilities (like a web interface) for this stage in the modeling and simulation work. The availability of all the agent capabilities was important to make sure we could include the human aspect. However, it forced us to think more in terms of agents than of mechanisms, which simplified the concept into service providers and service requestors, but complicated some of the design work by needing to translate everything into that paradigm. It also probably exacerbated the problem with defining the mechanisms. The good news is that there is now a baseline system available and it is always easier to make something better once you can actually use it.

4.2 The Impact of the DATASEM Project and Related Research

Initially motivated by the ineffectiveness of integrated master schedules in rapidly changing operational environments and the success of Kanban approaches for the knowledge work of software development, the project has expanded to investigating kanban as well as other adaptive governance mechanisms applicability. The initial Kanban scheduling system (KSS) networks described in SERC RT-35 sought a way to prioritize engineering tasks based on SoS or complex system capability priorities and task interdependencies by selecting value-adding features first, reducing wait time for scarce engineering specialties, and minimizing time wasted on context switching by overloaded resources. The KSS network concept provides two valuable side effects. First, the implementation of the network supports critical conversations about
schedule and value decisions by the appropriate people at the right time and nearest the actual implementation. Second, the network significantly improves executive and systems engineering visibility into the status of multiple independent development organizations.

The development of DATASEM and the DDSL provide a flexible, available, and inexpensive way to research how governance, organization, and work characteristics impact flow through an organization. New concepts can be mixed and matched with traditional concepts. Various types, sources, and cadences of work flow can be combined or assessed independently. Research can be conducted on the predictive nature of more subtle measurements for identifying trends, on improving the accuracy of status, and on understanding the impact of assigning, monitoring and using value, work load and other characteristics.

### 4.3 Future Research Opportunities

As discussed in the findings, DATASEM and the DDSL are still quite immature. Use, feedback, and change cycles are necessary for them to evolve and reach their potential.

#### 4.3.1 DATASEM Evolution and Transition

The first generation DATASEM Suite will provide a great deal of functionality and information. However, only a subset of possible models and mechanisms have been developed, and the DSL and infrastructure can be extended to address a variety of additional investigations. The following are opportunities to extend the capabilities and enhance the usefulness of DATASEM.

- **INCOSE** is conducting a broad analysis of the agile systems engineering life cycle during 2015 and 2016. This should identify additional mechanisms and approaches that are not directly supported by the first generation. It also provides a way of validating some of the first generation mechanism simulations.
- There needs to be a focused empirical effort to build a substantial database of different kinds of organizational structures, workflows and governance models to establish a baseline against which pilot information can be compared and the simulations calibrated accordingly.
- Industry has suggested we consider the concept of technical debt in the DATASEM models and as a significant attribute for characterizing particular configurations of work flow, organization, and governance.
- DATASEM has attracted significant attention from the system of systems community as a way to not only investigate, but also demonstrate the impact of change on both existing and adaptive governance mechanisms, and to measure the effectiveness of various combinations. Adding to the DSL and infrastructure support for modeling organizational negotiation and communication activities related to SoS governance can improve management understanding of the critical nature of such activities to effective governance.
- DATASEM can be applied to acquisition governance processes to develop and test alternative governance approaches and to identify and create ways to mitigate incompatibilities between adaptive and traditional governance mechanisms.
- All of these require the ability to identify, evaluate, and implement new mechanisms and approaches that emerge from the surveillance of adaptive management processes.
4.3.2 DATASEM-SEEA INTEGRATION

The Systems Engineering Experience Accelerator (SEEA) is a SERC research program aimed at creating automated learning environments that simulate real world experiences of systems engineers. Those experiences will be vivid and realistic enough to significantly accelerate the learning and maturation of those systems engineers. The project has just completed its 4th year and has successfully deployed the SEEA concept at the Defense Acquisition University in an experience tied directly to defense acquisition of an Unmanned Air Vehicle. The project is completing a set of experience development tools and is actively looking for organizations interested in building experiences.

DoD, OSD, and industry have expressed interest in adapting DATASEM for use with the SE Experience Accelerator project to provide a broader variety of management, governance, domain, and organizational environments within which to build experiences. We believe there is considerable value in a collaborative effort to integrate the DATASEM simulations into the Experience Accelerator architecture, develop an exemplar experience to validate its effectiveness, and continue to enhance the DATASEM role as a part of the SEAA portfolio of capabilities. It will also be useful to support significant learning experiences with adaptive management and governance mechanisms to support transition and the necessary cultural and infrastructural changes that are required for the mechanisms to fulfill their promise.

5 REFERENCES


APPENDICES
APPENDIX A.
AN OVERVIEW OF THE DATASEM DOMAIN SPECIFIC LANGUAGE (DDSL)

This section provides a semi-formal overview of the DDSL components. DDSL is continuously evolving, so this is a current snapshot of the language. The most up-to-date description can be found in the DDSL Reference Guide, which is a living document. The guide can be found on the SERC Website [URL to be specified]

Appendix B presents a complete experiment including the DSL describing it. It provides an example of how the components work together.

USER LIBRARY COMPONENTS

User Libraries describe the common characteristics for the building blocks of the Experiment Models. The User Library section is bracketed by the reserved words

• UserLibraries
• End UserLibraries

The User Libraries consist of four sections: ServiceProviderTypes, WorkItemTypes, Services, and GovernanceStrategies.

SERVICE PROVIDER TYPE

The Service Provider Type section is bracketed by the reserved words

• ServiceProviderTypes
• End ServiceProviderTypes

Each Service Provider Type is constructed in the following manner.

ServiceProviderType <name> { <SPTOption> }

where <name> is a required alphanumeric identifier for the ServiceProviderType

<SPTOption> can be one or more of the following:

hierarchy = <N>

An option that specifies the hierarchical level of this particular Service Provider Type, where <N> is the hierarchical level, an integer from 0 to ..., with 0 representing the lowest possible level].

WORK ITEM TYPE

The Work Item Type Section is bracketed by the reserved words

• WorkItemTypes
• End WorkItemTypes
Each Service Provider Type is constructed in the following manner.

**WorkItemType** `<name>` { `<WITOption>` `<WITOption>` ... }

where `<name>` is a required alphanumerical identifier for the `WorkItemType` `<WITOption>` can be one or more of the following:

- **hierarchy** = `<N>`
  
  An option that specifies the hierarchical level of this particular Work Item Type where `<N>` is the hierarchical level, an integer from 0 to ..., with 0 representing the lowest possible level. Only the reserved types (DevTask, Analysis, Resolution) are of hierarchy = 0 (“Task”), and user only specify types of Aggregation Nodes at all levels, which should always have hierarchies larger than 0.

- **ValueFunction** = `<functionType>` { `<VFAttribute>`, `<VFattribute>` ,... }
  
  A function that calculates the current value of a work item of this type. There are several types of ValueFunctions, each with their own attributes. The DSL can be extended to include others.

  **if** `<functionType>` is Derive then `<VFattribute>`’s are:

  - **HierarchyFactor** = `<R>` where `<R>` is a number between 0.0 and 1.0 that represents a weighting factor for the inherited value a work item of this type inherits from its parent.

  - **DependencyFactor** = `<R>` where `<R>` is a number between 0.0 and 1.0 that represents a weighting factor for the dependency-based value calculated from the values of work items that are dependent on this work item

  **else if** `<functionType>` is Fiat then `<VFattribute>` is

  - **Value** = `<R>` Where `<R>` is a number defines the value of the “value” attribute of a WI in Experiment Model.

**AutoGenerateWIN** = `<autoGenerate>` { `<Autogen Parameter>` }

This function automatically creates work items of type “DevTask” for an aggregation node of this work item type

**if** `<autoGenerate>` is set True then `{ `<Autogen Parameter>` }` should be set as:

- **MinTaskSize**=`<N1>` , **MaxTaskSize**=`<N2>` , **Complexity**=`<R>`

  - **MinTaskSize**=`<N1>` , where `<N1>` is an integer number larger than 0 that specifies the minimum efforts of each task to be generated from abstract descriptions.

  - **MaxTaskSize**=`<N2>` , where `<N2>` is an integer number larger or equal to `<N1>` that specifies the maximum efforts of each task to be generated from abstract descriptions.
Complexity = <R>, where <R> is a real number larger than 0 (suggested to be less than 1.0) that specifies the average number of direct prerequisites of each task to be generated from abstract descriptions.

**SERVICES**

The Services Section is bracketed by the reserved words

- Services
- End Services

Each Service Provider Type is constructed in the following manner.

**Service = <name>**

These names will be referenced in building Organizational Model and Work Items Network Model. The Services component is bracketed by the reserved words

**Services**

**End Services**

**GOVERNANCE STRATEGY**

A Governance Strategy (GS) is a Service Provider’s approach to accepting, scheduling, and performing work. Every SP has a GS that is either inherited from a parent or specifically declared for its own activities. The Governance strategies component is bracketed by the reserved words

- GovernanceStrategies
- End GovernanceStrategies

Each Governance Strategy defined is constructed in the following manner.

<GSName> = <Type> {<Mechanism>}

where <GSName> is an alphanumeric identifier and <Type> is one of the following:

- **pull**
  
  Requester SP adds the needed WI to a ready queue available to one or more candidate SPs. When a candidate SP has a resource availability, it will select from this or other requester SP’s ready queues according to its own external work selection process. The selected WI is then added to the candidate SPs BackLog(internal) queue. The use of a bidding system is also available, where candidate SPs may or may not respond with a Bid based on its own decision making. Then, the requester SP selects a Bidder and establishes a contract with it.

- **push**
  
  Requestor SP pushes a WI to a specific SP. In some cases, the SP candidate may decide to decline it.

Currently pull and push strategies share same format for Mechanism definition. This is
expected to change in later versions of DSL.

{<Mechanism>} is defined as <Name>: <value> {<mechanism-specific attributes>}

**WI Request Handling Cadence**

<Name> = *Cadence*
<Value> = <N> an integer number larger than 0 that specifies the frequency (in simulation steps?) a Service Provider performs the Requests Handling activity.

**WI Acceptance Mechanism**

<Name> = *Acceptance*
<Value> =
unlimited or limited {<mechanism-specific attributes>}

If limited, then the following Attributes need to be specified:
- BacklogLimit = <N> an integer number larger than 0 that specifies the upper limit of a SP’s Backlog Queue (WIs accepted but not started)
- WIPLimit = <N> an integer number larger than 0 that specifies the upper limit of a Service Provider’s Active Queue (WIs currently in progress)
- MaxMultiTasking = <N> an integer number larger than 0 that specifies the upper limit of number of Tasks each Resource can work on at the same time. Multi-tasking may incur a penalty. The default penalty coefficient is calculated as:
  \[ p = \frac{(n^2)}{n} \] where \( n \) = Number of Tasks a Resource is working on at the same time.

**WI Prioritization Mechanism**

<Name> = *Prioritization*
<Value> =
- FIFO (first in first out), or
- LIFO (last in first out), or
- EDD (earliest due date), or
- SPT (smallest processing time), or
- Neutral (random), or
- ValueBased {<mechanism-specific attributes>} (calculate a “current value” based on multiple attributes of a task)

For ValueBased, the following Attributes need to be specified:
- WeightCompleteness = <R> a real number larger than 0 that specifies the weights credited to the progress of target Task
**WeightPrecedency** = <R> a real number larger than 0 that specifies the weights that accounts for the level of severity that the target Task may block other Tasks

**WeightImpact** = <R> a real number larger than 0 that specifies the weights that accounts for the level of severity that the target Task may propagate changes to other Tasks

**WeightHierarchy** = <R> a real number larger than 0 that specifies the weights that accounts for the hierarchy of the target Task, as this Task might be of type “Analysis” associated with an AggregationNode.

**WI Assignment Mechanism**

<Name> = Assignment  
<Value> = 

*Neutral*, or  
*LeastLoad* (assign Task to connections with least current workload first), or  
*ExtendedCapacity* (assign WI to connections with largest “ExtendedCapacity” first).  
A Service Provider’s “ExtendedCapacity” on a given Service is computed as its own “Capacity” plus weighted sum of “Capacity” of all connected Service Providers.

**Resources Allocation Mechanism**

<Name> = Allocation  
<Value> = 

*Neutral* (randomly allocate capable resources to Tasks), or  
*MostEfficient* (allocate Resource with highest Skill Efficiency on Service required by Task)
EXPERIMENT COMPONENTS

This section defines the specific information for the organizational model and the workflow model and provides the necessary information about how the experiment is to be executed and what information and outputs produced. The Experiment Model component is bracketed by the reserved words

- **ExperimentModel <Name>**
- **End ExperimentModel**

MODEL VARIABLES

Users may create Model Variables and use them to assign attributes values of objects in Organizational Model and Work Items Network Model. The purpose of this feature is to

1) Allow users to change attribute values (e.g. efforts of certain type of WIs) in one step, and
2) Allow users to assign Random Distributions to objects to be created repetitively (e.g. skill efficiency of certain type of resources)

The Model Variables component is bracketed by the reserved words

- **Variables**
- **End Variables**

`Variable <name> : <type> <value>`

where `<name>` is a required alphanumeric identifier

If `<type>` is “num”, then `<value>` should be a real number

If `<type>` is “Random”, then `<value>` should be:

1) `.Uniform[<p1>,<p2>]`, where `<p1>` and `<p2>` are real numbers representing the lower bound and upper bound of a Uniform Distribution
2) `.Normal[<p1>,<p2>]`, where `<p1>` and `<p2>` are real numbers representing the mean and standard deviation of a Normal Distribution

ORGANIZATION MODEL

Users build the Organizational Model by describing the Service Providers (teams), the allowed workflow among Service Providers, the Strategy of each Service Provider, as well as the Resources (people) and their Skills.

The Organization Model component is bracketed by the reserved words

- **OrganizationModel**
- **End Organization Model**
Service Provider

In the Organizational Model, users specify a number of Service Providers. The Service Providers component is bracketed by the reserved words

- ServiceProviders
- End ServiceProviders

<name> type: [ServiceProviderType] {
    ( assignTo { [ServiceProvider]+ } )?
    strategy: [GovernanceStrategy] {<Mechanism> +}
    resources: { <Resource> +}
}

Where <name> is a required alphanumeric identifier for the ServiceProvider

The “type” attribute is required, and must be the name of one of ServiceProviderTypes as defined in User Libraries.

The “assignTo” attribute is optional. If specified, then the [ServiceProvider]+ contents in the braces should be names of Service Providers, excluding the current Service Provider itself.

The “strategy” attribute is required, should be the name of one of Governance Strategies as defined in User Libraries. <Mechanism> is optional but allows adding mechanisms or changing values from the GS defined in User Libraries. See Governance Strategy section above.

The “resources” attribute is required, should be one or multiple new Resource descriptions as explained in the next section.

Resource

<name> * <amount> skills: { [Skill]+ } 

Where <name> is a required alphanumeric identifier for the Resource

<amount> = <N1> where <N1> is an integer larger than 0 that specifies the number of this type of resource to be created in the model. <N1> can either be a:

1) Integer Number, or
2) [ModelVariable], which is the name of one of the Model Variables as defined before

The “skills” attribute is required, should be one or multiple new Skill descriptions defined in the next section.

Skill

[ [Service], <efficiency> ] 

Where <$[Service]$> is the name of one of the Services as defined in the User Library.

<efficiency> = <R> where <R> is a real number larger than 0 that specifies the efficiency level of this resource on a Service. <R> can also be a [ModelVariable] as predefined.
The standard value of Efficiency is 1.0 (100%). Theoretically, this resource will take $X$ time units to complete a task which requires $X$ units of efforts on corresponding service. However, there are situations which may cause change to this theoretical time duration, including:

1) **Multi-tasking** (one resource on multiple tasks),
2) **Collaboration** (multiple resources on one task),
3) **Rework** (due to uncertain nature of itself or due to change propagation from other WIs),
4) Suspended due to technical difficulties and request for **Resolution** from WI’s requester.

**WORK ITEMS NETWORK MODEL**

In each Work Item Network, users specify a number of Work Items. The Work Item Network Model component is bracketed by the reserved words

- `WorkItemNetworkModel`
- `End WorkItemNetworkModel`

**WORK ITEM NETWORK**

In each Work Item Network, users specify a number of Work Items. The Work Item Network component is bracketed by the reserved words

- `WorkItemNetworks`
- `End WorkItemNetworks`

`WorkItemNetwork <name> { WorkItem+ }`

**Work Item**

Users may define a Work Item, either as

1) A simple Task of `WorkItemType=DevTask`, or
2) A complex AggregationNode of any `WorkItemTypes` with `hierarchy=>0`.

1) Define a Dev Task

`<name> type: <[WorkItemType]> {`

- `( prerequisites { [WorkItem]+ } )?`
- `efforts { <Effort> }`
- `(impacts: { <Impact> +} )?`
- `(value=<stakeholderValue>)?`
- `(arrival=<arrivalTime>)?`
- `(duration=<dueDuration>)?`

`}`
Where `<name>` is a required alphanumeric identifier for the *Work Item* 

`[WorkItemType]` = DevTask in this case.

“*prerequisites*” attribute is optional. If specified, then the `[WorkItem]+` contents in the braces should be names of Work Items in the same Work Item Network, excluding the current Work Item itself. It is suggested that “prerequisites” of a WI to have the same hierarchy level as this WI.

“*efforts*” attribute is required, should be a single new `<Effort>` description.

“*impacts*” attribute is optional, should be a single new `<Impact>` description.

“*value*” attribute is optional. If specified, `<stakeholderValue> = <R1>` where `<R1>` is a real number larger than 0 that specifies the value delivered when this WI is completed. `<R1>` can also be a `[ModelVariable]` as predefined.

“*arrival*” attribute is optional. If specified, `<arrivalTime> = <N1>` where `<N1>` is an integer larger than 0 that specifies the time this WI to be activated in the simulation. `<N1>` can also be a `[ModelVariable]` as predefined.

“*duration*” attribute is optional. If specified, `<dueDuration> = <N2>` where `<N2>` is an integer larger than 0 that specifies the time this WI to be activated in the simulation. `<N2>` can also be a `[ModelVariable]` as predefined.

Note that if a WI is a subtask of another WI (upper WI), then the first WI would be activated due to arrival of, and analyzing of, the upper WI. Also, subtasks will inherit due date specifications of their upper WI(s). Moreover, subtasks will inherit stakeholder value from their upper WI(s) according to the upper WI(s)’ value function. So, it is not necessary to specify a WI’s value, arrival or duration unless it is of highest hierarchy in the model.

2) Define an Aggregation Node
   2.1) Aggregation Node with specified subtasks

   `<name> type: <[WorkItemType]> { 
   ( prerequisites { [WorkItem]+ })?
   decomposesTo { [WorkItem]+ }
   (analysisPhases: { `<AnalysisPhase>` + })?
   (impacts: { `<Impact>` + })?
   (value=<stakeholderValue>)?
   (arrival=<arrivalTime>)?
   (duration=<dueDuration>)?
   }
   Where `<name>` is a required alphanumeric identifier for the *Work Item*
[WorkItemType] is the name of one of WorkItemTypes as defined in User Libraries.

“prerequisites” “impacts” “value” “arrival” “duration” attributes are optional. See previous section.

“decomposesTo” (subtasks) attribute is required, the [WorkItem]+ contents in the braces should be names of Work Items in the same Work Item Network, excluding the current Work Item itself. It is suggested that “decomposesTo” (subtasks) of a WI to be exactly one level lower in hierarchy.

“analysisPhases” attribute is required, should be one or more new <AnalysisPhases> description.

2.2) Aggregation Node with abstract description on decomposing subtasks

<name> type: <[WorkItemType]> {
    ( prerequisites {[WorkItem]+})?
    (analysisPhases: {<AnalysisPhase>+})?
    efforts {<Effort>}+ 
    (impacts: {<Impact>+})?
    (value=<stakeholderValue>)?
    (arrival=<arrivalTime>)?
    (duration=<dueDuration>)?
}

Where <name> is a required alphanumeric identifier for the Work Item

[WorkItemType] is the name of one of WorkItemTypes as defined in User Libraries. In this case, this WorkItemType needs to have “AutoGenerateWIN” mechanism specified (see section 3.1)

“prerequisites” “impacts” “value” “arrival” “duration” attributes are optional. See previous section.

“analysisPhases” attribute is required, should be one or more new <AnalysisPhases> description.

“efforts” attribute is required, should be one or more new <Effort> descriptions.

Effort

<Effort> is described as [ [<Service>], <serviceEffort> ],

[Service] is the name of one of the Services as defined in the User Library.

<serviceEffort> = <R1> where <R1> is an real number larger than 0 that specifies the nominal efforts required for a type of Service. <R1> can also be a [ModelVariable] as predefined.

Analysis Phase

<AnalysisPhase> is described as [ [<Service>], <serviceEffort> ],
[Service] is the name of one of the Services as defined in the User Library.

<serviceEffort> = <R1> where <R1> is an real number larger than 0 that specifies the nominal efforts required for a type of Service. <R1> can also be a [ModelVariable] as predefined.

**Impact**

<WorkItem> [ <probability> , <severity> ]

[WorkItem] should be name of a Work Item in the same Work Item Network, excluding the current Work Item itself. It is suggested that the impacted WI(s) to have the same hierarchy level as the source WI.

<probability> = <R1> where <R1> is an real number between 0 and 1.0 that specifies the likelihood in general for the source WI would propagate changes to the impacted WI. <R1> can also be a [ModelVariable] as predefined.

<severity> = <R2> where <R2> is an real number between 0 and 1.0 that specifies the severity in general when the source WI propagates changes to the impacted WI. <R2> can also be a [ModelVariable] as predefined.

---

**EXPERIMENT SETTINGS**

This component provides a number of parameters that can control the simulation. The Experiment Settings component is bracketed by the reserved words

- **ExperimentSettings**
- **End ExperimentSettings**

**WIN Replications**

The WIN Replications component is bracketed by the reserved words

- **WINReplications**
- **End WINReplications**

**create** [WorkItemNetwork] * <numRep> at [ServiceProvider], where:
  - [WorkItemNetwork] is the name of one of the Work Item Networks as created in Work Items Network Model.
  - [ServiceProvider] is the name of one of the Service Providers as created in Organizational Model.
  - <numRep>=<N1> where <N1> is an integer larger than 0 that specifies how many times this WIN will be repeated in the simulation.

---

**5.1.1 EXPERIMENT PARAMETERS**

**Num Replications** = <N>
A positive integer. Number of Monte-Carlo Simulation replications to be conducted

**Task Maturity Levels = <N>**

A positive integer. Number of Total Maturity Levels for all WIs whose type=DevTask. In simulation, nominal efforts for each increment in Maturity Levels is calculated as:

\[
\frac{Total\ Nominal\ Efforts}{Total\ Maturity\ Levels}
\]

When a DevTask increases its maturity level, it may trigger Rework / Change Propagation / Resolution Request.

**Task Uncertainty = <R>**

A real number between 0 and 1. Represents probability for a DevTask to Request Resolution when increases its maturity level. A WI of type=Resolution will be created, requesting a skill from the requester Service Provider of this source DevTask, with nominal efforts =

\[
\frac{0.5 \times N}{Total\ Maturity\ Levels}
\]

Where \(N\) = nominal efforts of the source DevTask.

**Change Propagation Factor = <R>**

A non-negative real number, to be multiplied on severity of all change propagation occurrences. Default = 1.0 (thus no change)

**Learning Factor = <R>**

A real number between 0 and 1. When a Rework / Change Propagation / Resolution Request occurred, then the probability of triggering events of same type is reduced to:

\[
Pr_0 \times (1 - lf)
\]

Where \(lf\) is Learning Factor and \(Pr_0\) is the original probability for the corresponding event to occur.

**Rate Of Return = <R>**

A real number between 0 and 1, representing interest rate in computing NPV of Value Delivery over time.

Suggested value: 0.12 (12%)

---

**Miscellaneous**

Comments may be added in the form /* <comment> */
APPENDIX B.
AN EXAMPLE EXPERIMENT: MULTIPLE DEFENSE PRODUCTS

This Appendix provides a complete description of a DATASEM experiment.

DESCRIPTION OF THE EXPERIMENT

This study investigates a composite of 2 real-world cases of System-of-Systems Engineering projects. It considers the work break-down structure, requirements, complexity and value delivery management, the organizational structure and services, as well as system engineering activities and governance strategies involved in a command and control system component (AOC-WS) and an aircraft development (X49 Helicopter).

This study is based on various project documents from the stakeholder, the project contractor and other sources. Attempts of conceptual modeling and simulation of this case in DATASEM framework is made. The purpose of this study is to better understand the SoSE topic as basis for programs development.

The AOC-WS (AN/USQ-163 Falconer) is the senior command and control element of the U.S. Air Force’s Theater Air Control System and provides operational-level command and control of air, space, and cyberspace operations, as well as joint and combined air, space, and cyberspace operations. Capabilities include command and control of joint theater air and missile defense; time-sensitive targeting; and Intelligence, Surveillance, and Reconnaissance management.

The Piasecki X-49 "SpeedHawk" is a four-bladed, experimental high-speed compound helicopter under development [Wikipedia].

Organizational and WIN structure were developed referencing from real industry data: 1) New Model Helicopter Design 2) AOCWS / Air-force Operations Center Weapons System 3) DoD-SE Guide

The experiment compares agents’ performance by changing two aspects of their strategy configuration:

- Work Assignment (Neutral random vs. Balanced Assignment)
- Work Prioritization (FIFO (no prioritization) vs. RPW (rated precedency weighting

It compares the aspects under 3 different levels of:

- Rework Risk
- Technical Bottleneck Probability
- Impacts Severity

Figure 3 Describes the overall context of the Experiment and Figure 4 describes the dual-purpose organizational structure.
Figure 3. Multiple Defense Products Overview

Figure 4. Workforce Simulated
Figure 5. Multiple Defense Products Organization

The Experiment specifically tracked the following:

Total Workload - Sum of (nominal efforts * progress) of all WIs on the agent at each time step

Active Workload - Sum of (nominal efforts * progress) of in-progress WIs on the agent at each time step

Resource Utilization Rate - (Number of Busy Resources / Number of Total Resources) of the agent at each time step

Workload Imbalance - The coefficient of variation of the agents’ resource utilization rates at each time step.
THE DSL CREATED FOR THE EXPERIMENT

ModelBuilder: AerospaceAndDefenceProjects

UserLibraries

ServiceProviderTypes
  SETeam { [hierarchy=2] // Lead System Engineering Team
  DomainTeam { [hierarchy=1] // Domain Teams
  DevTeam { [hierarchy=0] // Developers
  CtrTeam { [hierarchy=0] // Contractors
end ServiceProviderTypes

WorkItemType
  SoSCap { [hierarchy=3]
    ValueFunction=Derive{[HierarchyFactor=0.2,DependencyFactor=0.2]}
  }
  SysFuncCap { [hierarchy=2]
    ValueFunction=Derive{[HierarchyFactor=0.2,DependencyFactor=0.5]}
  }
  SysOperCap { [hierarchy=2]
    ValueFunction=Derive{[HierarchyFactor=0.2,DependencyFactor=0.5]}
  }
  SubSysReq { [hierarchy=1]
    ValueFunction=Derive{[HierarchyFactor=0.5,DependencyFactor=0.5]}
    AutoGenerateMin= True,MinTaskSize=5,MaxTaskSize=15,Complexity=0.5
  }
/*Please keep the following 3 types unchanged*/
  DevTask {} // General Task
  Analysis {} // Decomposition Analysis
  Resolution {} // Bottleneck Resolution
end WorkItemType

Services
  SoSE // System-of-Systems Engineering
  SE_Elec // System Engineering: Domains
  SE_Aero SE_Power SE_Infra SE_MD SE_SW
  // Engineering
  Eng_Elec Eng_Mech Eng_Aero Eng_Ctrl
  // ITs
  IT.IS IT_SW IT_DB IT_SS
Testing
end Services

GovernanceStrategies

MyStrategy1 type:pull[
  mechanisms {
    Cadence = 10
    Acceptance = Limited{
      BacklogLimit=6,
      WIPLimit=6,
      MaxMultiTasking=1
      /* other options: Unlimited */
    }
    Prioritization = ValueBased{
      WeightCompleteness=1, WeightPrecedency=1,
      WeightImpact=5, WeightHierarchy=3
      /* other options: FIFO/EDF/EDD/SPT */
    }
    Assignment = ExtendedCapacity /*(considering both my resources
    * and SPs which I can assign WI to)
    /* other options: Neutral / LeastLoad */
    Allocation = MostEfficient /*(already considering multitasking effects) */
    /* other options: Neutral */
  }
]

MyStrategy2 type:push[
  mechanisms {
    Cadence = 10
    Acceptance = Unlimited
    Prioritization = FIFO
    Assignment = LeastLoad
    Allocation = Neutral
  }
]
end GovernanceStrategies
end UserLibraries
ExperimentModel
AerospaceDefence
Variables
end Variables
OrganizationalModel
ServiceProviders

LeadTeam type: SETeam[
  assginTo [SE_Elec, SE_Aero, SE_Power, SE_Infra, SE_ND, SE_SW, TestingBase]
  strategy: MyStrategy1 {BacklogLimit=3, WIPLimit=3, MaxMultiTasking=2}
  resources:
    - SysEng*3 {skills:[SoSE, 1.0]}
]

SE_Elec type: DomainTeam[
  assginTo [Contractor_1, Contractor_2, Contractor_3]
  strategy: MyStrategy1 {BacklogLimit=3, WIPLimit=3}
  resources:
    - SysEng*3 {skills:[SE_Elec, 1.0]}
]

SE_Aero type: DomainTeam[
  assginTo [Contractor_2, Contractor_4]
  strategy: MyStrategy1 {BacklogLimit=3, WIPLimit=3}
  resources:
    - SysEng*3 {skills:[SE_Aero, 1.0]}
]

SE_Power type: DomainTeam[
  assginTo [Contractor_1, Contractor_3]
  strategy: MyStrategy1 {BacklogLimit=3, WIPLimit=3}
  resources:
    - SysEng*3 {skills:[SE_Power, 1.0]}
]

SE_Infra type: DomainTeam[
  assginTo [Contractor_1, Contractor_4, Contractor_5]
  strategy: MyStrategy1 {BacklogLimit=3, WIPLimit=3}
  resources:
    - SysEng*3 {skills:[SE_Infra, 1.0]}
]

SE_ND type: DomainTeam[
  assginTo [Contractor_2, Contractor_3, Contractor_5]
  strategy: MyStrategy1 {BacklogLimit=3, WIPLimit=3}
  resources:
    - SysEng*3 {skills:[SE_ND, 1.0]}
]

SE_SW type: DomainTeam[
  assginTo [IT_Team_1, IT_Team_2, IT_Team_3]
  strategy: MyStrategy1 {BacklogLimit=3, WIPLimit=5}
  resources:
    - SysEng*5 {skills:[SE_SW, 1.0]}
]

Contractor_1 type: CntrTeam[
  strategy: MyStrategy1
  resources:
    - Elec_Eng*2 {skills:[Eng_Elec, 1.2]}
    - Mech_Eng*2 {skills:[Eng_Mech, 1.0]}
    - Aero_Eng*1 {skills:[Eng_Aero, 1.0]}
    - Ctrl_Eng*1 {skills:[Eng_Ctrl, 0.8]}
]

Contractor_2 type: CntrTeam[
  strategy: MyStrategy1
  resources:
    - Elec_Eng*1 {skills:[Eng_Elec, 1.0]}
    - Mech_Eng*3 {skills:[Eng_Mech, 1.2]}
    - Aero_Eng*1 {skills:[Eng_Aero, 1.0]}
    - Ctrl_Eng*1 {skills:[Eng_Ctrl, 0.8]}
]

Contractor_3 type: CntrTeam[
  strategy: MyStrategy1
  resources:
    - Elec_Eng*1 {skills:[Eng_Elec, 1.0]}
    - Mech_Eng*1 {skills:[Eng_Mech, 0.8]}
    - Aero_Eng*2 {skills:[Eng_Aero, 1.2]}
    - Ctrl_Eng*1 {skills:[Eng_Ctrl, 1.0]}
]
Contractor_4 type: CntrTeam
  strategy: MyStrategy1
  resources:
    Elec_Eng*1 {skills:[Eng_Elec, 1.0]}
    Mech_Eng*1 {skills:[Eng_Mech, 0.8]}
    Aero_Eng*1 {skills:[Eng_Aero, 1.0]}
    Ctrl_Eng*2 {skills:[Eng_Ctrl, 1.2]}
}

Contractor_5 type: CntrTeam
  strategy: MyStrategy1
  resources:
    Elec_Eng*1 {skills:[Eng_Elec, 1.0]}
    Mech_Eng*2 {skills:[Eng_Mech, 1.2]}
    Aero_Eng*1 {skills:[Eng_Aero, 1.0]}
    Ctrl_Eng*1 {skills:[Eng_Ctrl, 0.8]}
}

IT_Team_1 type: DevTeam
  strategy: MyStrategy1
  resources:
    IT_Mng*1 {skills:[SE_SW, 1.0]}
    IT_Dev*6 {skills:[IT_SG, 1.0], [IT_SW, 1.0], [IT_DB, 1.0], [IT_SS, 1.0]}
}

IT_Team_2 type: DevTeam
  strategy: MyStrategy1
  resources:
    IT_Mng*1 {skills:[SE_SW, 1.0]}
    IT_Dev*4 {skills:[IT_SG, 1.2], [IT_SW, 0.8], [IT_DB, 1.2], [IT_SS, 1.0]}
}

IT_Team_3 type: DevTeam
  strategy: MyStrategy1
  resources:
    IT_Mng*1 {skills:[SE_SW, 1.0]}
    IT_Dev*4 {skills:[IT_SG, 1.0], [IT_SW, 1.2], [IT_DB, 0.8], [IT_SS, 0.8]}
}

TestingBase type: DomainTeam
  strategy: MyStrategy1
  resources:
    Tester*3 {skills:[Testing, 1.0]}
}

end ServiceProvider
end OrganizationalModel

WorkItemNetworkModel

WorkItemNetwork MyProjects {

  WorkItems {
    PX49 type: SoSCap {
      // PX49 Helicopter Development Project
      decomposes {Sys, ASys, PSys, Infra, MDSys}
      analysisPhases {[SoSE, 30]}
      value = 4000
      arrival = 1
      duration = 600
    }
    AOCWS type: SoSCap {
      // AOC Weapon Sys Integration Project
      decomposes {Infra, MDSys, SI}
      analysisPhases {[SoSE, 30]}
      value = 6000
      arrival = 201
      duration = 600
    }
    ESys type: SysFuncCap {
      // "Electronics System"
      decomposes {R1, R2, R3, R5, R8, R9}
      analysisPhases {[SE_Elec, 30], [SoSE, 30]}
    }
    ASys type: SysFuncCap {
      // "Aerospace System"
      decomposes {R4, R8, R13, R10, R14, R12, R15}
      analysisPhases {[SE_Aero, 30], [SoSE, 30]}
    }
  }
}
PSys type: SysFuncCap {
  // "Power System"
  decomposes {R6,R7,R10,17,R11,R18}
  analysisPhases {{SE_Power,30,},{SoSE,30}}
}

Infra type: SysFuncCap {
  // "Infrastructure"
  decomposes {R8,R18,R20,R22}
  analysisPhases {{SE_Infra,30,},{SoSE,30}}
}

WDSys type: SysFuncCap {
  // "Weapons and Defense System"
  prerequisites [Infra]
  decomposes {R19,R21,R23}
  analysisPhases {{SE_WD,30,},{SoSE,30}}
}

SI type: SysFuncCap {
  // "Software Integration"
  prerequisites {ESys,ASys,PSys}
  decomposes {R23,R24,R25}
  analysisPhases {{SE_SW,30,},{SoSE,30}}
}

R1 type: SubSysReq {
  // "Air Conditioning"
  analysisPhases {{SE_Elec,10,},{SE_Aero,10,},{SE_SW,10}}
  effects {{Eng_Elec,15},{IT,IT,15},{IT,IT,30},{IT,IT,50},{Eng_Ctr,140}}
  impacts {R1[0,2,0,0],R4[0,2,0,0],R6[0,2,0,0],R7[0,2,0,0],
           R8[0,2,0,0],R12[0,2,0,0]}
}

R2 type: SubSysReq {
  // "Auxiliary Electrics"
  analysisPhases {{SE_Elec,10,},{SE_Aero,10,},{SE_SW,10}}
  effects {{Eng_Elec,50},{IT,IT,15},{IT,IT,30},{IT,IT,50}}
  impacts {R1[0,2,0,0],R3[0,2,0,0],R4[0,2,0,0],R14[0,2,0,0],
           R16[0,2,0,0],R18[0,2,0,0],R12[0,2,0,0]}
}

R3 type: SubSysReq {
  // "Avionics"
  analysisPhases {{SE_Elec,10,},{SE_Aero,10,},{SE_SW,10}}
  effects {{Eng_Aero,70},{IT,IT,30},{IT,IT,50},{Eng_Elec,30},{Eng_Ctr,20}}
  impacts {R1[0,2,0,0],R5[0,2,0,0],R11[0,2,0,0],R11[0,2,0,0],
           R18[0,2,0,0],R6[0,5,0,0],R10[0,5,0,0],R12[0,5,0,0],
           R15[0,5,0,0],R7[0,7,0,0],R19[0,5,0,0]}
}

R4 type: SubSysReq {
  // "Bare Fuselage"
  analysisPhases {{SE_Elec,15,},{SE_Aero,15}}
  effects {{Eng_Aero,70},{Eng_Ctr,50}}
  impacts {R2[0,2,0,0],R10[0,2,0,0],R13[0,2,0,0],R14[0,2,0,0],
           R15[0,2,0,0],R17[0,2,0,0],R18[0,2,0,0],R1[0,7,0,0],
           R3[0,7,0,0],R5[0,7,0,0],R7[0,7,0,0],R8[0,7,0,0],
           R9[0,7,0,0],R12[0,7,0,0],R11[0,5,0,0],R19[0,5,0,0]}
}

R5 type: SubSysReq {
  // "Cabling and Piping"
  analysisPhases {{SE_Elec,30,},{SE_Elec,30}}
  effects {{Eng_Elec,60}}
  impacts {R6[0,2,0,0],R1[0,5,0,0],R4[0,5,0,0],R10[0,5,0,0],
           R11[0,5,0,0],R12[0,5,0,0],R13[0,5,0,0],R14[0,5,0,0],
           R15[0,5,0,0],R19[0,5,0,0],R2[0,7,0,0],R3[0,7,0,0],
           R7[0,7,0,0],R9[0,7,0,0]}
}

R6 type: SubSysReq {
  // "Engine Auxiliaries"
  analysisPhases {{SE_Elec,15,},{SE_Aero,15,},{SE_Power,10}}
  effects {{Eng_Elec,15},{Eng_Aero,15},{Eng_Mech,30},{Eng_Ctr,40}}
  impacts {R7[0,7,0,0],R10[0,2,0,0]}
}
R18 type: SubSysReq 
  // "Transmission"
  analysisPhases [SE_Elec,10], [SE_Aero,10], [SE_SW,10]
  efforts [Eng.Elec,10], [IT_SYS,10], [IT_SYS,10], [IT_SYS,10], [Eng.Mech,75], [Eng.Ctrl,40]
  impacts[R18[0.0,0.0], R31[0.0,0.0], R18[0.0,0.0], R17[0.0,0.0], R19[0.0,0.0]]
} 
R19 type: SubSysReq 
  // "Operational Platform"
  analysisPhases [SE_Elec,10], [SE_Aero,10], [SE_SW,10]
  efforts [Eng.Elec,10], [Eng.Mech,25], [IT_SYS,85], [IT_SYS,100], [IT_SYS,100], [Eng.Aero,25], [Eng.Ctrl,25]
  impacts[R19[0.0,0.0], R19[0.0,0.0]]
} 
R20 type: SubSysReq 
  // "Data Processing"
  analysisPhases [SE_SW,30], [SE.Infra,30]
  efforts [Eng.Elec,10], [IT_SYS,40], [IT_SYS,40], [IT_SYS,40], [IT_SYS,40]
  impacts[R20[0.0,0.0], R20[0.0,0.0], R20[0.0,0.0], R20[0.0,0.0]]
} 
R21 type: SubSysReq 
  // "Weapons Testing"
  analysisPhases [SE_SW,30], [SE.Infra,30], [SE.W0,00]
  efforts [Testing,100]
} 
R22 type: SubSysReq 
  // "Interfacing"
  analysisPhases [SE.Elec,10], [SE.Aero,10], [SE_SW,10], [SE.Infra,30]
  efforts [Eng.Elec,25], [IT_SYS,40], [IT_SYS,40], [IT_SYS,40], [IT_SYS,40]
  impacts[R22[0.0,0.0], R22[0.0,0.0]]
} 
R23 type: SubSysReq 
  // "SOI"
  analysisPhases [SE.Elec,10], [SE.Infra,30]
  efforts [IT_SYS,40], [IT_SYS,40], [IT_SYS,40]
  impacts[R23[0.0,0.0], R23[0.0,0.0], R23[0.0,0.0]]
} 
R24 type: SubSysReq 
  // "Pilot Testing"
  analysisPhases [SE.Elec,10], [SE.Aero,10], [SE_PW,10], [SE.W0,30]
  efforts [Testing,100]
  impacts[R24[0.0,0.0], R24[0.0,0.0], R24[0.0,0.0], R24[0.0,0.0], R24[0.0,0.0]]
} 
R25 type: SubSysReq 
  // "Integrated Testing Z"
  prequisites [R24]
  analysisPhases [SE.Elec,10], [SE_SW,10], [SE_PW,10]
  efforts [Eng.Elec,25], [IT_SYS,50], [IT_SYS,50], [IT_SYS,50], [IT_SYS,50], [Testing,60]
  impacts[R25[0.0,0.0], R25[0.0,0.0], R25[0.0,0.0], R25[0.0,0.0], R25[0.0,0.0], R25[0.0,0.0]]
} 
}
end WorkItemNetworks
end WorkItemNetworkModel

ExperimentSettings
  RTMApplicators
  create MyProjects*1 at LeadTeam
end RTMApplicators

ExperimentParameters
  ExperimentMode = Batch {
    NumRealizations=000,
    TaskMaturityLevels=45,
    TaskUncertainty=0.2,
    ChangePropogationFactor=1.0,
    ErrorRisk=0.2,
    Volatility=1.0,
    LearningFactor=0.2,
    MultiTaskingPenalty=0.33,
    RateOfReturn=0.12
  }
end ExperimentParameters
end ExperimentSettings
end ExperimentModel
EXPERIMENT RESULTS

The results are presented in graphical form. Graphs were generated using the data results imported to Minitab. Other graphical data is available in both

GENERAL TIME AND EFFICIENCY

Figure 6 shows the general influence of “Work Uncertainty” factor (level 1 ~ level 3). There is an increase in Total Time for completion and a decrease in Average Work Efficiency, where Work Efficiency is a WI’s (Nominal Efforts / Cycle Time). Cycle Time is computed as actual time spent on a WI, including rework, but not including waiting or suspension time.

![Graphs showing Total Time and Average Work Efficiency](image)

Figure 6. General Time and Efficiency Charts

MAIN EFFECTS PLOTS

Figure 7 shows Main Effects plots for the experiment and the effect of RPW. The Main Effect Plots indicate that RPW prioritization reduced Total Time and increased Work Efficiency. Balanced assignment also reduces the Total Time significantly, but does not help improving Work Efficiency.

Assignment rule and Prioritization rule are two aspects of the Strategy. The data suggests having the right Assignment rule may be more important than having the right Prioritization rule, but there is little impact on Work Efficiency.

The influence of different Assignment rules in the larger and complex model (Aerospace) is very small, suggesting that a more complex Prioritization rule needed.
INTERACTION PLOTS

The Interaction Plots shown in Figure 8 indicate RPW prioritization has a slight advantage over none-prioritization (FIFO) when Work Uncertainty increases. Combining RPW Prioritization and Balanced Assignment shows no benefits.

Figure 8. Interaction Plots
APPENDIX C.
FRAMEWORK FOR DATASEM VALIDATION

Dr. Forrest Shull, Software Engineering Institute, Carnegie Mellon University
A Framework of Empirical Studies of Kanban-Based Scheduling for Software-Intensive Systems

Forrest Shull

**December 2015**

**Technical Note**

CMU/SEI-2015-TN-010

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

http://www.sei.cmu.edu
Copyright 2015 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

This report was prepared for the
SEI Administrative Agent
AFLCMC/PZM
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and “No Warranty” statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

DM-0003160
Executive Summary

The traditional systems engineering discipline was developed over half a century ago, and has demonstrated its effectiveness in many projects and programs. However, in many domains, today’s programs are finding themselves challenged by a need to keep pace with ever-faster changes in user needs, and in the technologies that constitute the solution space. System requirements are more likely to be less tangible, more complex, emergent, and adaptive. New system engineering methods are needed to address these challenges effectively [Turner13].

One approach that has been proposed to do exactly that is the use of Kanban-based Scheduling Systems [KSS] in systems engineering. Kanban represents a fundamental shift in workflow management that relies on “pull scheduling” to maximize the value to users that can be delivered by the system engineering program. Kanban visualizes the work in progress (WIP) on a program and only allows new work to be started when some resource is available to do it – usually when another piece of work has been completed. Kanban relies on the application of WIP limits (any given resource can only have so many work items in progress at a given time), algorithms for selecting the next piece of work to be started when resources are available (usually based on the work item with the highest value to end users), and applies Classes of
Service to manage flow. A KSS network builds on these principles by creating a tiered set of Kanban levels, each of which has its own information flows but all of which communicate with one another. For example, a set of KSS levels could include

1) Executive / Stakeholder Management (ESM), which decides which capabilities will be scheduled for development.

2) Capability Engineering (CE), which handles all Systems Engineering activities that provide support to product teams.

3) Product / Domain Engineering (PDE), which implements the product requirements assigned to a given team.

A review of the literature (Section 2 of this report) indicates that evidence does exist concerning the utility of Kanban and KSS principles in relevant environments, but more work needs to be done to quantify the expected effects and the contexts in which they can be expected to hold. The vast majority of the empirical evidence found relies on anecdotal reports or on qualitative data drawn from interviews and surveys with developers. While important, such qualitative evidence may represent the subjective views of the developers more than the underlying reality. The minority of studies that were identified that do collect quantitative data from real environments do corroborate the qualitative evidence, but represent only a few development contexts with no claim of wider applicability.

The DATASEM tool has been designed to fill this gap and allow the simulation of KSS principles in realistic project contexts, so that the likely effects can be understand and the impact of this new approach can be studied.

This report outlines a framework of empirical studies, building one upon the other, which can be executed to validate the tool itself and the results of its simulations. It also analyzes which metrics need to be collected and monitored by the tool in order to support those studies, providing confidence that the tool would be designed in such a way to provide confidence in its outputs.

The framework of empirical studies includes:

1. Validation that the simulation capabilities and the Domain Specific Languages that describe given scenarios are capable of capturing a given system engineering environment with sufficient accuracy.

2. Once the simulator has been validated for at least some environment and scenarios, analyze the effects of changing key scheduling processes from “traditional” approaches to ones that capture key KSS principles.

3. Once a comparison between traditional and KSS principles in the same development context has been executed, repeat the study with alterations in specific context variables (e.g. number of resources available, expertise available, cost of resources) to understand the parameters under which KSS adoption does and does not provide value.
4. Finally, for a given scenario that might represent a team intending to adopt KSS principles, conduct a deeper analysis of specific values representing resource efficiency (e.g. resource usage, Work Item delay) to understand where bottlenecks exist and where the system could be fine-tuned to deliver more value. (E.g. where investment in additional resources of a given type can provide benefit.)

To support these studies, a number of specific metrics are documented in Section 3 along with the rationales for why their analysis would provide insight. Across all of the studies, however, it is important to keep in mind two major dependent variables that describe the effectiveness of any Kanban-based scheduling:

- *Time to deliver all of the work items.* This represents the amount of calendar time needed for a user or stakeholder of the system to get the full delivery of capability.

- *The area under the curve when graphing value delivered over time.* This represents how much functionality could be delivered, and how early, to stakeholders. Using this value as a way to compare different scenarios rewards strategies that get more incomplete functionality into the hands of users early, on the assumption that even incomplete capabilities can satisfy some user needs and elicit important feedback for later system engineering.

Within these constraints, a number of different scenarios can be specified and evaluated in the tool, leveraging the benefit of modeling and simulation to get insightful information into the hands of decision makers.

6 Motivation and Background

The work described in this Technical Note is part of an ongoing research effort aimed at addressing the efficacy of Kanban scheduling principles in Systems Engineering. The motivation for this research has been laid out in a prior publication [Turner13]:

Traditional systems engineering (SE) developed over half a century ago, primarily driven by the challenges faced in the aerospace and defense industries. The environment was fairly uniform – hardware-driven, long lived, single mission. The result of this uniformity was that practices that worked well in that specific context were seen as best practices, and came to define the discipline of systems engineering. In the last few decades, system contexts have multiplied, and the speed of change in both needs and solution technologies has accelerated. This has led to an inherent loss of determinism—requirements are less tangible, more evolving, and sometimes emergent and systems are both complex and constantly adapting. The practice of systems engineering, with its roots in long-term, primarily hardware projects, has not kept pace.

Engineering principles involving agility and leaness have been adopted to address nondeterminism 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. 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.

This ongoing research examines one of those approaches, kanban (pull) scheduling techniques, to determine its applicability to systems and software engineering in a rapid response environment. Derived from lean manufacturing and modified significantly to apply to knowledge work such as software engineering, pull scheduling aims to provide the highest value flow through a system by visualizing the work in progress (WIP) and only starting additional work when some resource is available to do it – usually when another piece of work has been completed. It applies WIP limits, selection based on next highest value, and Classes of Service to manage flow.

As part of this overall effort, researchers are constructing a modeling and simulation tool, called DATASEM. DATASEM is capable of representing Kanban as well as “traditional” systems engineering environments and allowing users to compare and contrast the effectiveness of different approaches to systems engineering projects.

This report focuses on the additional research needed to determine the extent to which Kanban methods can be successfully applied in the initial definition of highly complex systems, or in the creation, operation, and management of systems of large, independently evolving systems. It lays out a framework for empirical studies using the simulation tool. The goal of this framework is to ensure that the tool is instrumented for the necessary data collection, and that studies are designed appropriately to enable the evaluation and calibration of DATASEM.

Prior research and application has shown that the Kanban approach can be successful in the development and evolution of individual systems. A number of benefits have been posited: By prioritizing engineering tasks based on System-of-System (SoS) or complex system capability priorities and task interdependencies, it is hypothesized that application of Kanban leads to implementation of the most value-added features first, a reduction in the wait time for scarce engineering specialties, and minimizes time wasted on context switching by overloaded resources. The studies described in this document present a way to test and accumulate evidence for or against such benefits in particular contexts.

6.1 Key Terms

To maintain consistency in terminology and tie the metrics discussed as closely to DATASEM as possible, this document uses the following terms to refer to basic concepts modeled in the simulator. These are extracted from the DATASEM Domain Specific Language (DDSL) Reference Guide [Turner15].

- Work item – A basic representation of a task to be done as part of a system engineering project. Work items are associated with other work items in a
Hierarchical decomposition. That is, to achieve a high-level task, a number of more detailed tasks can be specified.

- Work items are currently described as to whether they represent an Incremental Capability or a New Capability.
- Additional suggestions for ways to categorize Work items to support analyses are found in Section 3.
- When new work items become available, they may be “pushed” to resources with the appropriate skillset (although those resources can decline) or “pulled” by resources with spare capacity. (In the case that multiple resources attempt to pull the same work item, a bidding occurs.)

- **Resource** – A resource is an actor in a system engineering project who completes work items.
  - Resources belong to some organization.
  - Resources possess skills; for each skill they have an efficiency rating. That is, different resources can accept different sets of tasks, and perform differently on those tasks.
  - Resources have a limited number of items that can be Work In Progress (WIP). These are activities which have been started but not yet completed. When the WIP limit is reached, no new items can be started until one is completed.
  - Resources have a backlog, a queue that represents the work accepted to be done but not yet started. Each resource has a limit to how many items can be on the backlog.

### 6.2 Roadmap of this Document

Section 2 reviews other published empirical studies of Kanban principles to 1) understand what metrics have been feasible and useful to collect in prior studies; and 2) review the benefits of Kanban principles (both proposed and demonstrated) to understand how to collect measures from DATASEM that can be used to detect such benefits in its simulations.

Section 3 begin by laying out the goals of the empirical study for this research task, in order to understand what metrics have to be collected to satisfy those goals. Section 3.3 and later sections present a framework of empirical studies which rely on the identified metrics to achieve those research goals.

### 7 Related Work

There has been much written about the Kanban approach. One of the most widely used references remains *Principles of Product Development Flow*, a book by Donald Reinertsen, one of the first to articulate the key principles that became known as Lean Produce Development [Reinertsen09]. However, the scientific literature that examines the effect of these principles on software-intensive systems through
measurement and empirical study, as we intend to do, is much sparser. In this section we review the available studies in this domain with an eye on not only their documented results, but also their approach to measurement and analysis of Kanban-like processes.

Most comprehensively, a 2013 study ("Kanban in software development: A systematic literature review" [Ahmad13]) systematically reviewed the available published literature concerning Kanban’s effectiveness in software-intensive systems development. The authors reviewed 492 papers which resulted from their keyword search and selected 19 as primary studies containing relevant data. Because of the comprehensive nature of this systematic review we rely heavily on this study’s results rather than re-analyze the original studies themselves. It is important to note that only two of these studies included quantitative measures; the remainder presented evidence relying on a mix of experience reports, qualitative data (e.g. interviews, surveys), and simulations.

The study authors cite several expected benefits of Kanban in this type of context: “By limiting work in progress a sustainable pace of development is achieved, yielding higher quality products and greater team performance. The combination of improved flow and higher quality software helps to shorten lead time, leading to regular releases that help in building trust among the customers.” The evidence compiled from the 19 papers surveyed by the authors included a more granular breakdown of benefits, as shown below. The number in parentheses represents the number of studies, out of 19 possible, providing evidence on that point; more studies may indicate that the effect is more robust, i.e. liable to be experienced in more development contexts.

- Better understanding of whole processes (7)
- Improved software quality (6)
- Improved meeting of customer needs and customer satisfaction (6)
- Increased motivation of engineers (6)
- Improved communication/coordination between stakeholders/ in team (5)
- Bugs were fixed more quickly, WIP made it easier to handle blocking in work (5)
- Increased software productivity (4)
- Problem solving (easy detection and removal of bugs) (3)
- Reduced batch size (3)
- Decreased time to delivery (2)
- Increased release frequency (2)
- Efficiently controlled software projects (2)
- Changes to requirements made welcome (2)
• Early feedback on features, without delays (1)
• No massive documents (limited to customer request) (1)

The authors do acknowledge that, “in most of the studies, these benefits were reported at the general level,” indicating a dearth of hard data demonstrating these effects.

As indicated above, studies on Kanban are much more likely to report qualitative findings than quantitative impact. A representative study is “On the Impact of Kanban on Software Project Work: An Empirical Case Study Investigation” [Ikonen11], a case study of Kanban teams in an academic software development environment. The study did not collect quantitative metrics from the development process but relied instead on qualitative interviews with the developers, which focused on 9 dimensions in which Kanban is expected to impact development that were extracted from the literature: amount of documentation produced; approaches toward problem solving; visualization; ability to understand the whole project; communication; degree to which the team embraced the method; feedback; approval process used; strategies for selecting work assignments. Based on the interview results, the researchers concluded that all of these dimensions were supported within the team by the Kanban approach, with the exception of “understanding the whole” and “improved feedback” (specifically, from the customer). In these last two areas, no evidence was found that improvements were supported. [Ikonen11] (It is interesting to note that the lack of evidence for “understanding the whole” contradicts the results of the systematic literature review [Ahmad13], where that was the benefit most commonly reported.)

Other authors (“Studying Lean-Kanban Approach Using Software Process Simulation,” [Anderson11]) have constructed simulators and examined the results of using Kanban on relatively small sample systems (in this case, 4 activities and 100 work items). As their primary outcome value they use cost, which includes both the time to deliver the system and the queue limits themselves. Another attempt to apply simulation compared the results of Kanban-lean to waterfall and scrum lifecycles (“Simulating Kanban and Scrum vs. Waterfall with System Dynamics,” [Cocco11]). In this simulation, the authors used “time to deliver implemented requirements” as the outcome on which to differentiate the different approaches.

More relevant to our purposes here, “Quantifying the Effect of Using Kanban vs. Scrum: A Case Study” [Sjoberg12] is one of the very few studies using metrics from an industry project to assess the impact of Kanban. This was not included in the systematic literature review by Ahmad et al., described above. In this study the authors conducted an in-depth analysis of metrics from a single development project in a medium-sized software company that switched from Scrum to Kanban. Because the development paradigm changed while important other factors stayed the same, the authors were able to analyze key metrics before and after the switch to defensibly quantify the impact of Kanban, at least compared to Scrum. By analyzing data associated with more than 12,000 work items collected over the
years 2009-2011, the authors concluded that by using Kanban instead of Scrum the company:

- Almost halved the lead time (the time to get a version of the software ready for release);
- Reduced the number of weighted bugs by 10% (in this analysis, bugs were weighted by criticality, with a “blocking” bug weighted 8 times as much as a “minimal” bug);
- Improved productivity by 21% for Project Backlog Items; and
- Reduced productivity by 11% for bug fixes.

In summary, from this survey of evidence regarding Kanban’s use on software-intensive systems in the scientific literature, we find several studies detailing benefits for Kanban. We note that most of the evidence is qualitative, based on interviews and surveys of developers. Many of the dimensions examined in these studies (e.g. customer satisfaction, motivation of developers) are outside the scope of our own work. However, several of these dimensions can be abstracted up to values that can be tracked in DATASEM: *Productivity, software quality, time to delivery, amount of time activities spend blocked.*

It is also important to note that the one study where we were able to find hard and defensible numeric data (drawn from the same environment before and after a switch to Kanban) corroborated many of these benefits. This study contributes evidence that it is feasible to collect several key metrics from Kanban teams: *Delivery of value over (calendar) time, bug-proneness of the software, and productivity for various classes of work (new functionality versus rework, in particular).*

In the next section, we lay out experimental designs by which these and additional metrics can be collected to address key research questions.

8 Empirical Study Framework

8.1 Research Goals and Their Implications for Measurement

The proper design of measurement and empirical study needs to be tied to the larger goals for the research project in general, in order to ensure that the measures collected and analyzed have relevance and provide insight. As a description of the goals of this research project in general and DATASEM in particular, we rely on a prior slideset developed by Dr. Richard Turner, the SERC PI, which listed the following goals:

- “Better visibility and coordination managing multiple concurrent development projects”

Implication for measurement: Any set of measures deployed on this project must be able to deal with the DoD’s emphasis on Systems of Systems, meaning that metrics need to be rolled up to provide a comprehensive view over multiple systems and
multiple organizations. There are likely to be different measures of progress at the enterprise level (the Executive / Stakeholder Management tier) versus the sub-organization level.

This is accomplished by aggregating the measures defined for the system as a whole versus the resources owned by any given sub-organization; thus, resources need to be tagged according to which organization they belong to. Metrics representing value at the enterprise level versus value to the sub-organization will likely need to be defined for each specific scenario, but as a default position we suggest that the value to the enterprise could be measured by *value delivered over time* (indicating whether the sponsor of the work is satisfied with the pace of delivery), while the value to the sub-organization could be measured by the *degree of utilization of its associated resources* (i.e. how close to capacity the WIP and backlog queues are, on average, over the life of the simulated project). When it comes to resource utilization, over- or under-utilization are both challenging.

- **“More effective integration and use of scarce SE resources”**
  
  Implication for measurement: There must be a way to measure the utility provided by resources in the network and the value produced by System Engineering.

  This is accomplished by assigning each work item a *work item value* that represents its utility to stakeholders in the system. It is important to note that such values are able to change depending on the effectiveness of the System Engineering process employed. For example, the utility of late work may decrease sharply over time representing that its value to the end user diminishes if not available in a timely fashion.

- **“Increased project and enterprise value delivered earlier “**
  
  Implication for measurement: This is a central goal of the measurement plan: Every Work Item must have a value assigned as well as a date of delivery. It must be possible to aggregate value delivered by time, so that the outcomes of different approaches can be compared in the simulation.

  As described earlier, this is accomplished by assigning each work item a *work item value* that represents its utility to stakeholders in the system. Each work item likewise has a *date of completion* associated with it.

  From these two metrics, we can plot outcomes of the simulations as a two-dimensional graph, with “cumulative product value delivered” on the y-axis and time on the x-axis. The metric we want is the *area under the curve* for each scenario: Higher values represent more value delivered and earlier delivery of the value.

- **“More flexibility while retaining predictability”**
  
  Implication for measurement: The measurement plan must be able to measure the difference in performance between governance strategies that are locally optimized versus enterprise optimized.
This is accomplished by tagging each requirement which generates work items as to whether it adds value at the enterprise or local level.

- “Less blocking of product team tasks waiting for System Engineering response”

Implication for measurement: The measurement plan needs to account for two types of resource contention: How much extra calendar time do work items require for completion, because the necessary resources are unable to get to them? Secondly, how much calendar time do resources sit idle, when their associated work items cannot be generated because another resource has to complete first?

This is accomplished by measuring the work item delay (i.e. the time spent on the backlog queue or as WIP) for each WI as it goes through the system. Also, queue length must be monitored so that the amount of time that queues sit empty while a relevant work item might have been generated is measured.

- “Lower governance overhead”

Implication for measurement: When applicable, work items must be able to be associated with governance overhead so that their contribution to the overall effort and calendar time can be measured.

This is accomplished by tagging each WI as “technical” work or a “governance mechanism.” It is assumed that different governance strategies will generate governance activities of different types and at different rates, and that this will be specified in the governance model. At the end of the simulation total calendar time and effort must be aggregated for each type of work item.

8.2 Goals for Measurement and Empirical Study Activities

Based on the above project goals, we formulated a set of specific goals for the measurement and empirical study part of this project. These goals were presented and discussed at a meeting with the project sponsors and the technical stakeholders in early November, 2015.

Goal 1. Validate the DATASEM modeling implementation. That is, show that the DATASEM implementation is a reasonable simulation of the phenomena under study.

Goal 2. Evaluate potential improvements to System Engineering programs due to adoption of KSS network concepts (as described in the final report of a predecessor Systems Engineering research task, RT-35 [Turner13]). Essentially, are the project goals listed in Section 3.1 being realized?

Goal 3. Understand the context under which any improvements due to KSS network concepts can be expected. DATASEM should be useful to stakeholders as a mechanism for trying out different governance structures (including mixed approaches to governance) and understanding their effects.
Goal 4. Identify potential areas of improvement in a given scenario. Show that running “what if” scenarios can result in practical insights that can make system performance better.

Because the DATASEM tool was still under development during the timeframe of this study, the SEI was asked to provide designs for empirical studies, and the associated measurement requirements, to be run in the future to satisfy those goals. Each study in the following sections is capable of satisfying one goal.

Each of the following subsections presents one such empirical study in turn, preceded by a general discussion of some underlying measurement considerations common to all of the studies.

8.3 General Measurement Considerations for Kanban

A key metric in any empirical study is the cost of delay, which will be scenario-dependent. This value will have a substantial impact on how the tool estimates the total value provided by the system engineering process, since it will be used to weight the value obtained for successfully completing each WI.

Any study will need a defensible way of estimating this value. The “cost of delay” represents one of the key tenets of KSS, namely that the cost of resources (and the cost of leaving resources under-utilized) needs to be balanced against the increased value of delivering value to the customer earlier.

Kanban experts such as Reinertsen state that there would be a global value for each scenario, based on the degree to which customer satisfaction is impacted by being late to market [Reinertsen09]. However, it is unlikely that this approach translates well to the system engineering context of our sponsors. For this reason we advocate assigning a normalized value to each WI based on the number of downstream tasks affected by each WI (so that tasks with a bigger “ripple effect” are penalized more for being late).

8.4 Validating DATASEM

Perhaps the most important goal of future measurement work on this project is to address Goal 1: Validate the DATASEM modeling implementation. Validation studies will provide confidence that the implementation of DATASEM developed by the research team appropriately models the system engineering phenomena of interest.

This study design is based on the following assumptions:

- There is no historical data from a real environment that can be used for comparison. Talks to date with the sponsors and external parties have shown that it is unlikely that we would have access to historical data from a real program, at the appropriate level of granularity, to use for comparison purposes against the model outputs. For that reason, the experimental design below
focuses on the elicitation of estimates from Subject Matter Experts (SMEs) via approaches such as the Delphi process (described below). Elicitation will be done for a specific system engineering scenario in a specific context, for key values in the model. If it were found that historical project data did exist, the research team should model that scenario in the tool, run the simulator, and compare results to recorded data.

- The research team can identify an ongoing program that we can shadow (i.e. we are not making any changes to the program, but just want to be able to talk to some of the stakeholders to get information).
- The research team can get access to 3-4 SMEs from that program.

Because of the assumed lack of usable historical detail, this study relies on the elicitation of important parameter values from SMEs as inputs to the model. A proven method for doing so is the Wideband Delphi Process (abbreviated in this report as “Delphi process”), which was popularized for use in the context of software-intensive systems by Barry Boehm’s book on *Software Engineering Economics* [Boehm81]. The Delphi process uses expert judgment as an input but presents a structured way of converging to a value that is usable for modeling and preserves the aggregated experience of the group. The steps of the Delphi process, tailored for the task of eliciting expert judgment on key attributes that are necessary for the systems dynamic model, include the following. Generally, the Delphi method is applicable for groups of 3-7 experts.

1. A facilitator presents each expert with a scenario (in this case, the given system context, and an explanation of the parameter(s) to be estimated).
2. The facilitator leads a meeting of all participants where the experts discuss issues related to the requested estimation task.
3. The experts document their estimates anonymously.
4. The facilitator summarizes the estimates and shares that information (still anonymously) back to the group of experts.
5. The facilitator leads another meeting to further discuss issues, especially related to areas where the estimates varied widely.
6. The experts document revised estimates. Steps 4-6 are iterated until sufficient convergence is achieved.

**Empirical study procedure 1:**

- Model the ongoing program (or some aspect thereof) in DATASEM. The objective is to model as accurately as possible at this time.
- Define a scenario – some snapshot in the lifetime of the program where it is possible to enumerate which work products are in which state of completion, and which Work Items are on the queues. The Work Items themselves may be
completely hypothetical although the resources, queues, and governance model should be as accurate as possible.

- Hold a workshop with 3-4 SMEs from the program. In any event, there should be no more than 7 SMEs participating.
  - Share the scenario with the SMEs.
  - Use a Delphi process to have SMEs converge on an estimate for cost of delay in this program. (Record the amount of disparity among SMEs as well as how many rounds are required to converge on a solution.)
  - For the given scenario, again use Delphi to have SMEs converge on predictions for:
    - Time required to deliver 25%, 50%, 75%, 100% of the product value (as measured by the value of the work items in the scenario).
    - Total amount of time work items spend blocked.
    - Again, record the degree of disparity and number of rounds required for convergence.
  - Run the simulation and present the results regarding time to deliver product value and the total time items spend blocked for discussion to the SMEs. It is important to note the time required to deliver product value and the total amount of time work items spend blocked, and how these compare to the SME estimates. It is also important to allow a free discussion (which would be recorded and analyzed) on factors that might account for any observed discrepancies.

The key outcome of the study will be the determination of whether the experts and the simulator agree sufficiently 1) at first, and 2) after the expert discussion of the simulation results. It is expected that the simulator and the experts are unlikely to agree at first, which is why it is important to allow the experts to discuss the results (essentially, this forms an additional iteration of the Delphi process with the simulator added as a new expert). This discussion session allows the research team to see whether the simulation results add anything new to the experts’ understanding of the scenario, e.g. whether it puts the spotlight on complex interactions among some resources that produce unexpected results. If the simulation results are ultimately unconvincing to the experts, the chance to visualize intermediate results produced by the simulator may help the experts “debug” where the modeled behaviors departed from the expected.

8.5 Evaluating the Impact of Kanban on Systems Engineering Programs

The intent of the DATASEM tool is to understand whether and how KSS concepts can improve the practice of Systems Engineering. To this end, once confidence in the tool has been demonstrated via the type of empirical study described in Section 8.4,
additional studies can be undertaken using the tool to simulate and quantify the impact of KSS concepts.

To this end we can identify several specific questions to be answered:

Question 2.1: Would moving to Kanban-based scheduling improve a given system engineering project / program?

“Improve” implies we want to compare Kanban scheduling to “traditional” scheduling, ideally for the same system or similar systems.

A common measure is needed to compare the output of different runs of the simulator. As discussed in Section 3.1 that should be the area under the curve of “product value delivered over time.”

Clearly, a key evaluation metric is the total amount of time required to deliver all of the completed work items. However, by evaluating product value over time, it is possible to prioritize the approach that delivers more value (i.e. more capabilities) into the hands of users earlier.

To get data useful to System Engineering decision makers we will want to measure cost as well as benefit. This data can better address the question of whether switching to Kanban-based scheduling for a given environment is worth the cost.

The cost of the process should be measured as:

\[(\text{Cost of capacity}) + (\text{cost of delay})\]

Which allows tradeoffs between the cost of adding additional resources and the cost of having fewer resources and not delivering as quickly.

If reasonable estimates can be obtained, the cost should also incorporate additional factors:

- The cost of using different types of resources. Cost may vary both by resource type (e.g. architects may cost less than managers but more than developers) as well as by other properties (e.g. more flexible resources may cost more; scarcer resources may cost more).

- Total amount of time tasks spent blocked

Empirical study procedure 2: For a modeled scenario (say, the context modeled for the validation study in Section 3.4), generate an additional scenario in which the governance model and other aspects of the simulation are perturbed to incorporate key KSS network concepts. Run the simulation and compare to the results from the “traditional” scheduling approach. If possible, it may be useful to reuse the experimental design from Section 3.3 to elicit expert opinion regarding how the Kanban principles will affect the results, using Delphi to drive the SMEs to converge on an estimate, and then allowing a free discussion of how and why those estimates diverge from the simulation outputs.
Question 2.2: Can we find evidence for key tenets of the Kanban philosophy (the key aspects upon which claims of Kanban’s value rest)?

There are several key assumptions that underlay Kanban; one advantage of a simulation-based approach is that they can be tested using various what-if scenarios. One such tenet which is both critical and seemingly counter-intuitive is:

**Leaving capacity under-utilized leads to better overall performance.** [Reinertsen, p. 3]

Testing this hypothesis can make use of the same overall success measures defined for Question 2.1 (total time to deliver all work items, area under the curve of value delivered over time). It also requires measuring:

- Percentage of capacity which is underutilized over the simulation run (i.e. the average amount of unused space on the WIP queue of resources)

Tweaks can be made to the appropriate models to result in larger or smaller amounts of capacity underutilized, and correlated with the success measures.

Empirical study procedure 3: Start from the same Kanban scenario used in empirical study 2, above. Alter the algorithms for pulling functionality (in DATASEM, part of the Governance model) so that resources do not bid on work items if their queues are above a certain percent full; although this would be scenario dependent it is suggested to run this for 100% and 80% capacity. Compare results to the original algorithm. This will test whether having more availability on the queue minimizes problems due to time-slicing and multitasking, and leaves room available for high-priority tasks that must be pushed down to resources.

Another tenet of Kanban that can be treated as a hypothesis and tested using simulation results is:

**More flexible resources (versus more specialized resources) lead to better performance.** [Reinertsen, p. 13]

Since the simulator assumes that different resources have different effectiveness rates for different tasks, we can test this by varying two parameters:

- The number of different skills that a given resource is “good at” (i.e. has high measures of efficiency for);

- The degree of improvement in efficiency that specialized resources have when performing their specialized task, versus other tasks.

Empirical study procedure 4: As above, start from an existing scenario and alter the skillsets and efficiency ratings of scenarios. Compare the results for several different parameter settings to understand how resource specialization affects the overall success of the work.
8.6 Understanding the Context for Likely Improvements

This study addresses Goal 3, understanding the context where improvements due to KSS will be seen.

To address this goal we want to simulate different products in different contexts, carefully varying important attributes so that we can understand likely outcomes in different situations. We don’t expect Kanban-style scheduling to be beneficial in every context, but we should be able to understand the parameter values where the benefit is more likely to be observed.

The key dimensions to vary include the following:

- **Cost of delay**
- **Scheduling algorithms – how resources get assigned as new tasks come up**
- **When scheduling decisions get made (E.g. having everything decided upfront in the lifecycle, versus decisions made later in the lifecycle based on more up to date info)**
- **Queue size**
- **Batch size**
- **WIP limits (E.g., Are resources able to be 120% booked, with accompanying reduction in effectiveness?)**
- **Feedback speed**
- **Flexibility of resources**

Empirical study procedure 5: Each of these dimensions should be varied in a separate study and analyzed against the success measures described in the prior section, in order to understand how that particular factor affects the outcome of the scenario. In this way the boundaries under which improvements due to Kanban are likely to be seen can be better understood.

8.7 Identifying Improvements in a Particular Scenario

The final goal for empirical study using DATASEM is to identify potential improvements in a given scenario. That is, supposing that prior evidence has been collected (perhaps through the empirical studies described above) to provide confidence in the benefits of KSS for a given scenario and in the results of the simulation tool, could the results then be used to optimize the application of Kanban in a given scenario?

To answer these questions we assume that the best way to improve a given scenario is to understand which queues are the bottlenecks that are preventing more
efficient utilization of resources. To address this, the following set of metrics should be monitored and the final values reported:

- *Which queues are most under- and over-utilized? (These indicate areas where work may be better distributed to other resources or where investments in new resources may be necessary.)*

- *Maximum and average queue size*

- *Aging distribution of items on each queue*
9 Summary

This report brings together a number of different technical areas in order to recommend a consistent and thorough measurement and empirical study plan for use on DATASEM. Specifically,

- It reviews the stated goals of the DATASEM research project in order to ensure that those goals can be addressed by the measurement instrumentation in the simulation tool, specifically with a focus on the anticipated benefits of a Kanban-based scheduling approach.

- It presents a review of the relevant scientific literature related to empirical study of Kanban and its use on software-intensive systems. The studies found are analyzed to understand what metrics they found feasible and useful to collect for demonstrating the practical utility of Kanban. It also reviews what has been empirically demonstrated regarding the effectiveness of Kanban in practice.

Based on the above information, and adding additional metrics that are supported by the simulation tool, this report then articulates four concrete goals of empirical study in this domain and proposes a specific study design capable of satisfying each goal. The goals build upon each other in a progression to show what could be accomplished using the constructed simulation tool:

1. Firstly, to validate that the simulation capabilities and the Domain Specific Languages that describe given scenarios are capable of capturing a given system engineering environment with sufficient accuracy.

2. Once the simulator has been validated for at least some environment and scenarios, it could then be used to study any changes involved in moving from a more traditional system engineering approach to one based on KSS. In particular, the metric of product value delivered over time was identified as a key way to capture the effect of Kanban.

3. Further refinements in the understanding of KSS principles can be achieved by altering some of the key parameters in the scenario while keeping the vast majority the same, and watching the impact on product value delivered over time and other key metrics. In short, this would allow an understanding of the contexts under which Kanban principles do or do not provide value.

4. For a given scenario, an analyst could do a deeper dive into very specific values produced by the simulation (e.g. resource usage, Work Item delay) to understand where bottlenecks exist and where the system could be fine-tuned to deliver more value. (E.g. where investment in additional resources of a given type can provide benefit.)

Obviously, much of the specifics of any empirical study will depend on the context in which the study is run: For example, which system engineering projects will be willing to experiment with the tool, provide SMEs, and allow their context to be
captured in models that can be run. However, the general outline provided in this report can be adapted to almost any specific study to provide insight.

Appendix: Glossary of Acronyms
CE    Capability Engineering, one level in a KSS hierarchy
ESM   Executive / Stakeholder Management, one level in a KSS hierarchy
KSS   Kanban-based Scheduling System
PDE   Product / Domain Engineering, one level in a KSS hierarchy
SME   Subject Matter Expert
WI    Work Item
WIP   Work in Progress

References/Bibliography
URLs are valid as of the publication date of this document.

[Ahmad13]
Muhammad Ovais Ahmad, Jouni Markkula, Markku Oivo: Kanban in Software Development: A Systematic Literature Review. EUROMICRO-SEAA 2013: 9-16

[Anderson11]

[Boehm81]

[Cocco11]
[Ikonen11]

[Reinertsen09]

[Sjoberg12]

[Turner13]

[Turner15]