DISSERTATION

A MODEL-BASED SYSTEM FOR ON-PREMISES SOFTWARE DEFINED INFRASTRUCTURE

Submitted by

Eric Enos

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2025

Doctoral Committee:

Advisor: Dr. Daniel Herber

Dr. Jediddiah McClurg
Dr. Erika Gallegos
Dr. Kamran Eftekhari Shahroudi
Dr. Steven Conrad

ABSTRACT

A MODEL-BASED SYSTEM FOR ON-PREMISES SOFTWARE DEFINED INFRASTRUCTURE

This paper is based on a case study of an IT organization in a large, US-based healthcare provider, and attempts to identify the applicability of on-premises, software-defined infrastructure to such organizations. These organizations are often grouped into departments by technical skill and support both operational work (tickets) and project tasks of various priorities that is regularly viewed as queued and assigned based on the priorities of the day. The key question at hand is whether the investment in the underlying technologies and processes would enable the gains in efficiency, quality, and scalability enjoyed by organizations leveraging DevOps methods in public cloud IT environments.

Using project and operational metrics from the case study organization, a hybrid simulation model using both system dynamics and discrete event simulation developed through this research depicts the flow of work through a skill-based team as well as many of the key factors that influence that workflow, both positive and negative. Experience indicates that the interaction between project and operational work – as well as between teams with differing skills – entangles work queues and wait times within those queues in a way that rapidly scales in complexity as the number of interacting individuals and teams increases. Results from model simulation bears out this intuition, and help answer the question as to whether – and where – automation of an on-premises software-defined infrastructure can be of benefit.

Following this, Model-Based Systems Engineering (MBSE) tools and methods are used to develop an initial, high-level reference architecture of a Software Defined Infrastructure (SDI). In particular, this section focuses on the capabilities required for a SDI management system that leverages the available programming interfaces of underlying IT infrastructure subsystems. These

IT infrastructure subsystems represent the installed base to be managed, yet at the same time are evolving over time with components being constantly upgraded or replaced in a large organization. The SDI management system must be designed in a way that provides stability for the automation yet also adapts to this evolution of the interfaced subsystems.

.

TABLE OF CONTENTS

ABSTRACT		ii
LIST OF TABLES		vii
LIST OF FIGURES		viii
Chapter 1	Introduction	1
1.1	Introduction to the Study	1
1.2	Background of the Problem	1
1.3	Statement of the Problem	3
1.4	Questions to be Answered	3
1.5	Design of the Study	4
1.6	Importance of the Study	4
1.7	Assumptions and Limitations	6
1.8	Organization of the Remainder of the Study	9
Chapter 2	Literature Review	10
2.1	Introduction and Organization	10
2.2	Significant Prior Research	10
2.2.1	Relevant Processes	11
2.2.2	Dynamics of Project Management	14
2.2.3	Dynamics of IT Service Management	17
2.2.4	Modeling and Simulation	19
2.2.5	IT Infrastructure	21
2.2.6	Automation Technologies	26
2.2.7	SysML and MBSE	28
2.2.8	Architecture Descriptions and Reference Architectures	30
Chapter 3	Simulation Modeling of the Work Environment	33
3.1	The Case Study Organization	33
3.2	Modeling the Work Environment of a Single Team	35
3.2.1	The Single Team System Dynamics Model	35
3.2.2	Full Single-Team Model	35
3.2.3	The Single Team Discrete Events Simulation Model	37
3.2.4	Interaction Points Between Models	39
3.3	Single-Team Simulation Results and Discussion – Long Iterations	40
3.4	Single-Team Simulation Results and Discussion – Short Iterations	41
3.5	Uncertainties in the Models	45
3.5.1	Data Gaps	45
3.5.2	Data Estimates and Simplifications	46
3.5.3	Validation of Estimates	47
3.6	Modeling the Work Environment of Two Teams	50
3.7	Improvement Focus Areas	51

Chapter 4	Developing an SDI Architectural Framework with MBSE	56
4.1	System of Interest	57
4.2	Solution Requirements	57
4.2.1	High-level Requirements	57
4.2.2	Example System Provisioning Requirements	58
4.3	Operational Viewpoint	59
4.3.1	Structural Perspective	61
4.3.2	Behavioral Perspective	64
4.3.3	Data Perspective	66
4.3.4	Services Perspective	67
4.3.5	Contextual Perspective	71
4.4	Logical / Functional Viewpoint	71
4.4.1	Structural Perspective	72
4.4.2	Behavioral Perspective	73
4.4.3	Data Perspective	77
4.4.4	Services Perspective	77
4.4.5	Contextual Perspective	77
4.5	Organization Specific Elaboration	77
Chapter 5	Presentation of Results	82
5.1	Introduction	82
5.2	Example Business Case for SDI	83
5.2.1	Objectives and Scope	83
5.2.2	Proposed Solution	84
5.2.3	Cost Analysis	84
5.2.4	Benefits Analysis	85
5.2.5	Assumptions and Constraints	86
5.2.6	Risks and Mitigation	86
5.3	Roadmap for SDI Implementation	87
5.4	Findings	88
5.4.1	Research Question 1: What are the basic components and capabilities of an on-premises SDI system?	88
5.4.2	Research Question 2: Should healthcare providers implement on-premises SDI?	89
5.4.3	Research Question 3: How should provider organizations proceed to implement on-premises SDI, if at all?	89
5.4.4	Research Products	90
5.5	Conclusions	90
5.5.1	Software Engineering Integration	91
5.5.2	Barriers to MBSE Adoption in IT	92
Chapter 6	Summary, Implications and Future Work	94
6.1	Summary	94
6.2	Implications	94
6.3	Future Work	94

Appendix A Appendix A: Simulation Scripts . . . 97
A.1 Simulink Script . . . 97
A.2 Vensim Script . . . 132

Appendix B Appendix B: Regression Scripts . . . 133
B.1 Regression Script . . . 133

Acronyms . . . 168

Bibliography . . . 171

LIST OF TABLES

3.1 Managerial Estimates for DES Inputs 47
3.2 Regression Targets 48
3.3 Optimal Variable Results from Direct Search 49
3.4 Regression Analysis 50

LIST OF FIGURES

1.1 "New School" technologies and processes. 3
1.2 Healthcare provider IT technologies and processes. 4
1.3 Overall flow of research. 5

2.1 Document-based or traditional approach to SE compared to MBSE. 25

3.1 Vensim system dynamics model of a single team reflecting both project and operational workflows. 36
3.2 Simevents queuing model depicting entity generators for each work type feeding four engineers with individual queues. 38
3.3 Scripting, parameters, and logic flow of iterations. 44
3.4 Schematic of regression analysis. 48

4.1 SDI System of Interest. 57
4.2 High-level functional requirements. 58
4.3 High-level non-functional requirements. 58
4.4 Automated provisioning requirements. 59
4.5 Mapping of provisioning requirements to high-level SDI requirements. 60
4.6 SDI Management System in the context of existing technical infrastructure and management tools. 61
4.7 Equivalent Visio Diagram of SDI Management System in the context of existing environment. 61
4.8 High level domain diagram of the SDI. 62
4.9 Allocation of high-level requirements to SDI domains. 65
4.10 Use case depicting the response of the SDI to a request or event. 66
4.11 Activity diagram outlining the SDI response to a generic event. 67
4.12 Sequence of activities between system modules in the provisioning process. 68
4.13 Conceptual Data Model for the SDI. 69
4.14 Block Definition Diagram of the integration engine. 73
4.15 IBD showing interfaces between the SDI domains. 74
4.16 Use case for requesting server and storage capacity. 75
4.17 Use case for provisioning server and storage capacity. 79
4.18 Possible states of a provisioning request. 80
4.19 SDI architectural layers. 80
4.20 Product-specific design example for the SDI Management System. 81

Chapter 1

Introduction

1.1 Introduction to the Study

The motivation for this study started with what seemed like a simple question: How could my organization, heavily dependent on physical, on-premises infrastructure and applications, leverage the automation techniques commonly used (and widely extolled) by other organizations in the public cloud to improve the quality and reduce the costs of IT, and, by extension, the organization as a whole? Many of the building blocks are readily available, but there is little guidance available to IT leaders who haven't already adopted them in the cloud as to whether the investments make sense, how to identify opportunities for improvement through automation, and how to go about justifying those investments.

1.2 Background of the Problem

The healthcare provider industry is highly dependent on purchased, Commercial Off-The-Shelf (COTS) applications with minimal custom development. This results in isolated pockets of critical data that must be merged through transactional integration and scheduled data Extract, Transform and Load (ETL) processes to allow consolidated decision making. For a variety of reasons, these systems remain largely deployed on-premises, with shifts of production workloads to public cloud service providers still limited. However, there are extremely potent and growing drivers for data sharing between systems and stakeholders, including support for internal Big Data and AI initiatives. In addition, the accelerating deployment of large numbers of networked biomedical Internet of Things (IoT) devices within clinical settings (and increasingly in patient homes) greatly increases the volume of consolidated real-time telemetry. The technology infrastructure should provide a deterministic platform to enable these initiatives concurrently with ”normal” clinical usage, but emerging behavior often leads to unpredictable performance and reliability.

At the same time, sustained high levels of merger, acquisition, and divestiture activity continue to increase these legacy footprints and their technical variability, while security threats demand more complex tool and process overlays. Finally, financial pressures force these highly variable technical environments to support business-shared services and centers of excellence amid cost controls and constrained headcount and skill sets. As a result, healthcare technology organizations are highly complex systems-of-systems with many conflicting demands.

There is an increasingly wide gap between highly promoted IT best practices such as cloud services adoption, DevOps methodologies, agile development life cycles, and infrastructure automation on the one hand (which I will refer to as ”new school” technology and processes, Figure 1.1) and the current reality of managing traditional enterprise COTS systems based on the concepts of the IT Information Library (ITIL) (esp. Versions 2 and 3) and driven by extensive and long-term capital investments made by providers in on-premises systems and infrastructure (traditional ”old school” technology and processes shown in Figure 1.2 below). Note the centrality of topics such as custom development in the public cloud, leveraging IaC and DevOps in the former model and contrast that with the importance of COTS systems running in private clouds (virtual environments on premises) managed through . The leverage of Software Defined Infrastructure (SDI) by an enterprise IT organization in fact results in the creation of a new system for the management of the infrastructure and enables transformation of certain use cases of IT management from manual to automated processing.

Anecdotes of successes and failures of these systems are easily found - surrounded by the claims of vendors of related technologies and tool sets and the opinions of pundits and analysts – but currently there are few rigorous data or objective guidance available to IT leaders in terms of systemic and high-leverage success factors, tools, processes to adopt, and consequences to address during and after any proposed change to .

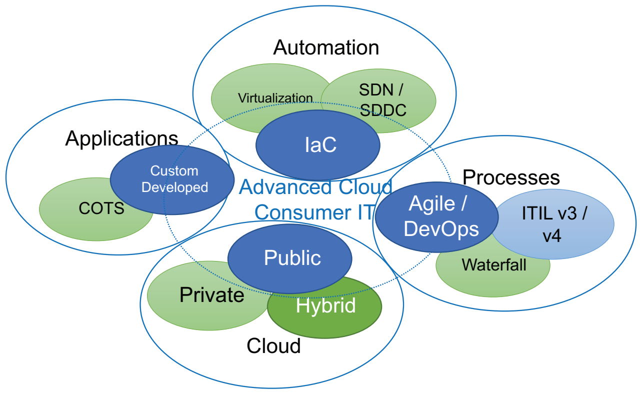

Figure 1.1: "New School" technologies and processes.

1.3 Statement of the Problem

Although the benefits of SDI are widely understood in the context of the public cloud, healthcare providers have lagged in the adoption of these services for a variety of reasons. At a time when the provider business units are actively investigating digital transformation to increase efficiency and quality, it remains unclear to healthcare provider IT leaders whether and to what extent they should adopt on-premises SDI. In addition, there is no clear guidance on how to assess your readiness for adoption, where to target these investments, and what organizational changes are recommended to realize the value of SDI.

1.4 Questions to be Answered

Research Question 1: What are the basic components and capabilities of an on-premises SDI system?

Research Question 2: Should healthcare providers implement in-house ?

Research Question 3: How should provider organizations proceed to implement on-premises SDI, if at all?

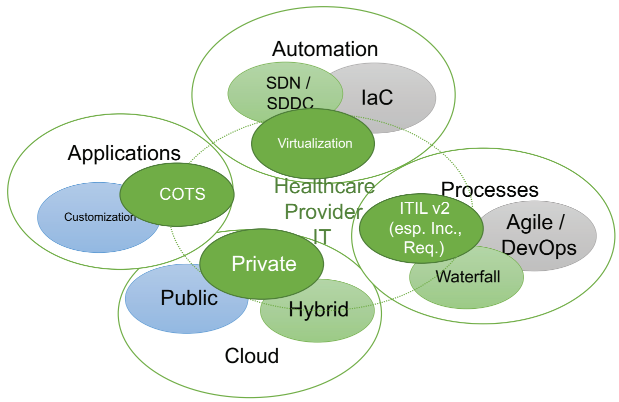

Figure 1.2: Healthcare provider IT technologies and processes.

1.5 Design of the Study

To conduct this research, I propose an explanatory sequential mixed-method approach to data gathering and analysis. This will include the development of simulation models of the work performed by a large representative healthcare IT organization, intended to identify potential areas of impact of a system. This will be followed by an organizational case study to highlight quantitative results of interest, proposing an architecture for a functioning SDI system and a decision model for leaders to use in determining readiness for and guiding implementation of SDI technologies and practices. The general flow of the research is shown in Figure 1.3.

1.6 Importance of the Study

There continue to be many industry articles published stating that the shift to SDI technologies and DevOps practices is necessary and inevitable (especially in the context of public cloud adoption), but precious few indicating how to build a new management system with a high probability of success, as these are provided by the public cloud providers inherently as part of their services –

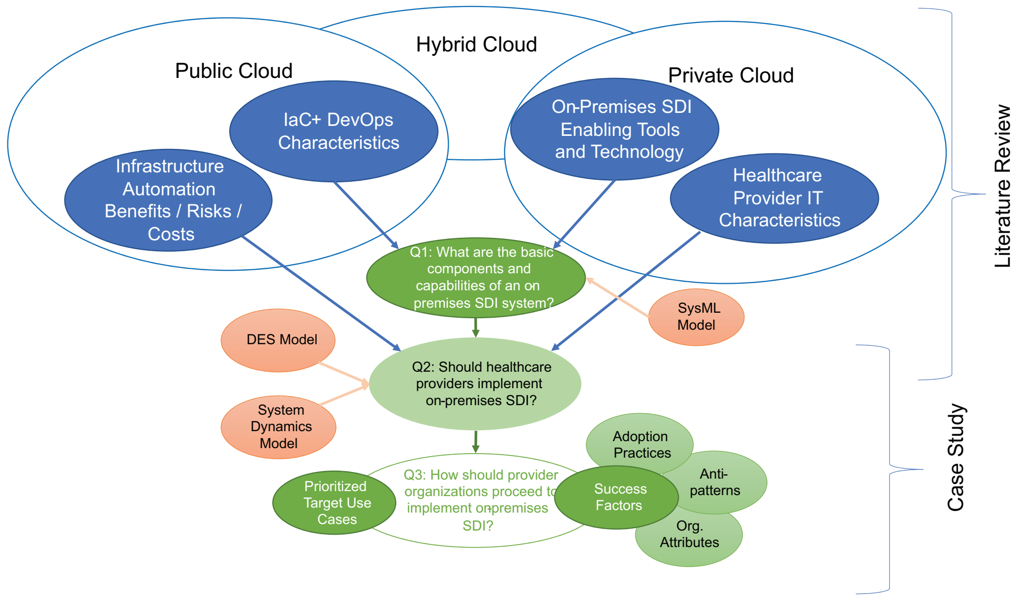

Figure 1.3: Overall flow of research.

and virtually none addressing the applicability of these technologies to on-premises infrastructure because:

- Public cloud infrastructure is often not appropriate (in terms of performance or cost, among other factors) for many healthcare provider applications. To the extent that it is appropriate, applications are generally deployed and supported in the same way as they would be on premises, as long-lived systems deployed, configured, and updated on virtual servers and storage.

- Significant on-premises infrastructures (especially network communications, but also key clinical solutions) must remain in place even when cloud-based applications are appropriate, resulting in hybrid cloud infrastructures.

This leaves healthcare IT decision-makers with a lack of proven recommendations on how to adopt the concepts of ”digital transformation” to their own operations, and major questions remain unanswered:

- Does a traditional approach to infrastructure management (e.g., on-premises systems managed manually or via 3rd party tool sets, through ITIL processes and waterfall projects) remain the best choice?
- Is an organization-wide shift to a modern application infrastructure and management system (e.g., cloud infrastructure and DevOps) relevant and realistic?
- Is a hybrid approach that builds a semiautomated management system the most appropriate option and, if so, is focused on which use cases?

According to National Institute of Standards and Technology (NIST) cloud services have the following characteristics and are available from public cloud service providers, but can also be built on premises using SDI technologies [1]:

- On-demand, self-service
- Broad network access
- Resource pooling
- Rapid elasticity or expansion
- Measured service

Note that while these characteristics do not explicitly require software-defined capabilities, the ability to programmatically provision and deprovision infrastructure capacity significantly enhances the first and fourth characteristics above and are critical to the development of agility and scalability common in DevOps environments.

1.7 Assumptions and Limitations

This study will focus on the applicability of SDI technologies and practices to large healthcare providers in the United States, which are generally assumed to share the following characteristics:

- Relatively low margins with increasing erosion due to evolving industry dynamics driving high rates of mergers, acquisitions, and divestitures and the resulting high levels of IT infrastructure variability.

- Earnings-driven incentive systems which discourage operational costs in favor of capital investments and drive:

  - Significant dependence on COTS applications and commensurately on long-lived, "mutable" systems (regularly changed, for example through infrastructure patching and application upgrades).

  - Significant deployment of on-premises IT infrastructures and toolsets (which increasingly support automation via APIs).

  - Prevalence of waterfall project methods over agile approaches (due to the above factors).

  - Limited direct adoption of the public cloud (PaaS and IaaS) and associated skills, tools and techniques. Cloud adoption is primarily focused on Software as a Service (SaaS) applications.

  - Limited IT staff depth and breadth generally and especially of software development skillsets.

  - Organization in traditional technology skills silos of mixed skill level that complete both scheduled (project) and unscheduled (event- and request-driven) work.

- High and increasing operational and clinical dependence on IT system and data performance and availability as direct contributors to clinical quality and safety, which has driven:

  - Ubiquitous adoption of ITIL 2/3 (especially for help desk, incident, and request management processes).

  - Increasing deployments of networked biomedical IoT and associated generation of large volumes of telemetry data (a form of "Big Data").

– Increasing need to leverage data – especially clinical – to improve clinical quality and operational efficiency.

Although many of the building blocks for SDI are already available in most organizations, some will inevitably need to be purchased. These building blocks generally come from a variety of vendors in any reasonably large-scale environment and together represent only a potential platform without substantial effort by internal IT to build a fully integrated solution with these blocks that can enable process automation. The overarching problem with platforms is that the user has to decide what to do with them (some assembly *is* required), and here the existing guidance for IT leaders in the literature remains weak. The second major goal of this research is to improve that guidance by providing a product-agnostic road map for implementation of on-premises SDI.

Finally, with the questions as to *whether* to build an on-premises SDI environment and *how* answered, the final goal of this research is to provide some guidance to healthcare IT leaders on how to select *which processes* should be automated.

The conclusions drawn from this research may have a broader applicability to organizations or industry segments with similar characteristics.

It is possible that the significant adoption of SDI technologies and practices on premises (private cloud, or even hybrid cloud) is not appropriate for some healthcare providers, or that it is only appropriate for certain limited activities (e.g. server provisioning), under certain conditions (in support of a specific development effort or project), or for certain applications (e.g. data center network micro-segmentation). In addition, it is possible that some technologies and practices (such as SDN or Scaled Agile) are more applicable on-premises than others.

This study assumes that full-scale migration to modern application architectures (i.e. ”cloud native”) in public cloud services is unrealistic for many key applications used by large healthcare providers and that, as a result, significant infrastructure must either remain and be managed on-premises or be managed in a similar manner in the cloud. The study also assumes that SDI technologies are generally available to healthcare care providers today, either currently or within reach of planned infrastructure re-update activities that would upgrade to hardware and software platforms that incorporate the appropriate application programming interfaces (APIs). Furthermore, the study assumes that healthcare provider systems do not support the widespread use of 'immutable' infrastructure (unchanging and therefore highly predictable) that underlies their applications, but instead require significant configuration and customization that preclude rapid re-creation of them. Regarding staff and skill levels within IT, the study assumes that provider organizations have minimal internal software development capability, including skills such as business analysis and software quality control.

These assumptions will be validated where possible through subsequent phases of the research.

1.8 Organization of the Remainder of the Study

The next major section of this research consists of a qualitative review of the literature of the related areas that underlie and the tools to be used throughout the investigation. Following this, an analysis is performed on whether and in what areas SDI is most suited to healthcare provider organizations, using modeling and simulation tools and a case study organization. With the motivation for and targets of SDI established, an architectural framework for an on-premises SDI is developed. The results of these two sections are combined in the next section to answer the research questions. Finally, the last section offers a summary and implications of this research and suggests areas for future research.

Chapter 2

Literature Review

2.1 Introduction and Organization

The modeling and simulation of relevant processes (both project and operational) using the techniques DES and SD techniques; on-premises SDI and research in related areas such as public cloud-based automation (e.g., Infrastructure as Code (IaC)); and support technologies (such as Software-Defined Data Centers (SDDC) or Software-Defined Networking (SDN)) and related processes such as DevOps. The review of the relevant technologies is reasonably complete; however, it remains to be fully elaborated in the final dissertation.

2.2 Significant Prior Research

There is significant research in the modeling of project and operational work management systems and processes, including the leverage of Discrete Event Simulation (DES) and System Dynamics (SD) in the understanding of complex processes as well as the prediction of benefits from improvement of them. Substantial prior research is available on the topics of software-defined networking, DevOps, infrastructure automation, and IaC. There is extensive coverage of these concepts in the context of off-premises cloud service providers, separately and in combination, in both the industry literature and peer-reviewed research. There is also a significant body of knowledge available on the topic of cloud adoption, both generally and within specific industries and regions. However, there is limited application of all of these concepts in combination to the problems of managing on-premises IT infrastructure (e.g., private and hybrid clouds), and there is no research available addressing how to best identify and prioritize opportunities for on-premises IT process automation leveraging such infrastructures.

2.2.1 Relevant Processes

IT in the healthcare provider context, and the organization used as a case study through this research, is generally organized around specific technology skills, with teams supporting both project and operational work. As a result, individuals are routinely assigned multiple project tasks, as well as operational tickets. This intensifies the need for team members to stop and start work on any given task or ticket based on changing work priorities. Of course, individuals vary in skill level and only the highest-skilled team members can complete every task or ticket assigned to the group quickly and with high quality. These complicate the queueing within the team as assignments are juggled between team members. In addition, many tasks and tickets require multiple skills to complete (e.g., a network engineer, a server administrator, and a security analyst), which results in queuing of work as it passes between teams. All of this contributes to a high percentage of queue time relative to the work being performed, which is deadly to timeliness and customer satisfaction.

Leaders of all types find it challenging to complete work in a timely and high-quality manner in a skills-based organizational model, although this has been treated most explicitly in the context of call centers [2] it is common across IT and in other functions such as R&D [3]. It is not unusual for a large IT organization to have many dozens of projects active at any given time, representing a large number of active tasks to be completed within a schedule. Similarly, most organizations have a similar (or larger) number of active tickets representing incidents and requests for service. Each of these tasks and tickets can have a variable — and sometimes changing — priority for completion. The time frame for this analysis is deliberately assumed to be short to prevent the addition of personnel. In this short time frame, leaders are limited to adjusting individual work priorities and shifting individual team members between different work items. See Mitchell [4] and the CPHIMS Review Guide [5] for additional information.

The work areas that each IT team must support include unscheduled work (incidents and requests, which arrive at random and unpredictable times and rates) and scheduled work (project tasks and scheduled maintenance). This is especially true for teams that are consolidated in larger organizations and offered as shared service functions (such as information security or networking).

In addition, these teams are organized around specific domain skills associated with each shared service function, resulting in a significant amount of collaboration in many common work processes. In addition to the complexity of work management, the units of work assigned to teams have variable initial priority levels which can change over time due to changes in urgency and other sources of managerial pressure – a key source of organizational conflict, according to Payne [6]. Franco et al. [7] discuss the importance of considering the dynamics of not just the product development / project management activities, but also the post-implementation phases of product lifecycles, as well as the complexity of technical and organizational interactions.

SDLC is a generic term commonly used in information technology to refer to the processes for planning, creating, testing, and deploying an information system. There are many different methodologies used in practice, including waterfall, iterative, and Agile (described below), which are IT-specific equivalents to common systems engineering methods such as linear, waterfall, "V" and spiral outlined in *Systems Engineering Principles and Practices* [8]. Despite the name, SDLC typically address only the project phases of an overall system life cycle, with production operations and eventual decommissioning generally left poorly addressed, if at all. The waterfall model is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous and corresponds to a specialization of tasks. The waterfall approach to project management is used almost exclusively in enterprise IT infrastructure as it remains well suited to the sequential tasks of hardware procurement, installation, configuration, testing, and transitioning to operations that defy incorporation into Agile methodologies. Waterfall methods are also often used in large projects with stable requirements, but suffer from poor outcomes when requirements change rapidly throughout the life of the project [9]. Iterative (or incremental) development is a method 'to develop a system through iterations (repeated cycles) and incrementally (in small portions of time).' A well-known example of an iterative development process is the Rational Unified Process (RUP) [10], which was developed by Rational Software as a productized software development process and is heavily associated with the Unified Modeling Language (UML). RUP was in fact partially created using UML notation.

Agile development is an “extreme” version of iterative methods, characterized by short and tightly restricted iterations (commonly two weeks), close coordination with the system customer, and an evolutionary approach to functionality. Scrum and Kanban are two popular variations of Agile software development, although there are other variations. Agile methods have been found to provide an alternative to traditional project management methods in situations where the business context and requirements can change rapidly [11]. Agile methods have grown significantly in recent years and are now the dominant methodology for pure software product development projects. Base Agile development methods often do not scale well as commonly articulated due to their focus on small cross-functional teams (what Jeff Bezos at Amazon refers to as the ”two pizza” rule) and relatively short time frames (2-4 weeks), as well as a common shortfall in systems architecture planning [12]. In addition, the viability of agile methods in general suffers when physical systems and logistics are included in the effort. Scaled Agile (SAFe) and Large-Scale Scrum (LeSS) are two attempts to apply Agile practices to much larger projects. SAFe incorporates structures at a program and enterprise level and addresses issues such as system architecture, product and implementation road maps, and portfolio management. SAFe also addresses the concept of an “architectural runway” to incorporate non-Agile activities such as hardware design, procurement, and deployment. In particular, the use of Kanban methods in Agile projects directly bridges the concepts of scheduled project tasks and the management of work through queues, and research has been conducted using queueing networks to model the dynamics of Kanban-based systems [13, 14]. Furthermore, the various levels of "backlog" used in Agile methods can be easily modeled as prioritized work queues. From the perspective of the skill-based teams modeled in this research, work generated through Agile management processes can be viewed as scheduled or unscheduled work: it is technically known and therefore scheduleable, however, due to the short planning time frames commonly associated with the two-week ’sprint’ tasks could also effectively be treated as unscheduled requests.

Project management practices are largely defined through two competing de facto standards organizations: the mutually supporting set of practices and standards based on () published by

Project Management Institute (PMI), and PRojects IN Controlled Environments (PRINCE2) published by Axelos (and formerly by Central Computing and Telecommunications Agency (CCTA). For the purpose of this research the focus will remain on project management, and not expand into the related by distinct areas of program and portfolio management. () was developed by the CCTA in Great Britain in the 1980s to provide a framework of best IT practices to obtain better quality at a lower cost. ITIL has served as the de facto standard for IT infrastructure and operations since the publication of version 2.0 in 2000. This was the first comprehensive methodology that attempted to address all aspects of the operational support of IT systems. The responsibility for the ITIL publications was transferred to Office of Government Commerce (OGC) in 2001. Version 3.0 was released in 2007 with a focus on end-to-end services and expanded the practices to encompass all aspects of IT, including service design and transition, areas traditionally covered by various SDLC, product and project management methodologies. Version 4.0 was released in 2019 and added coverage for Agile, DevOps and Lean concepts through the *ITIL4: High Velocity IT* [15] publication.

The models referenced in the sections below each address separate aspects of the work dynamic within a healthcare IT organizational model, but none fully explore the leverage points which automation can potentially address. As such, they will be used as the basis for a model that better highlights these areas. Each of these are expanded below, along with their contribution to the model for this study.

2.2.2 Dynamics of Project Management

Scheduled work in the form of projects has received substantial attention from researchers with an interest in system dynamics. Projects, especially large projects, have long been recognized as highly complex internally, as well as in relation to the rest of the organization [16, 17]. There is a rich body of research on the applicability of system dynamics as applied to the project management of single projects [18–21]. Lyneis and Ford, in particular, developed this model depicting the management of scheduled work through several iterations [22, 23]. Based on Lyneis' model, automation would be expected to reduce 'Effort applied' and increase 'Productivity,' which would increase the rate of 'Progress.' At the same time, automation would be expected to reduce the 'error fraction' and therefore the rate of 'error generation'. The combined effect is to increase the 'Work done' and decrease the 'Undiscovered Rework.' Other aspects of the model that affect morale occur over a substantially longer period than is normally considered for the management of day-to-day operational work. The same is true for models that explicitly focus on recruitment, training, and staff turnover, such as the work of Abdel-Hamid [24]. Ford and Sterman recognize another source of rework in their modeling of a product development process in addition to accidental errors: deliberate changes to the scope or requirements [25]. Their model further allows the representation of multiple interconnected project phases that require active coordination in a long-running project; however, these occur on a longer time horizon than that under consideration in this research. Rodriguez and Williams assess the implications of customer satisfaction in the context of projects, especially with regard to intolerance to milestone delay and the impact on management pressure and productivity [20]

Ordonez et al. [26] elaborated on the characteristics of a multi-project environment that apply to project managers, functional managers, and staff, including the need for staff to multitask between projects. Platje and Seidel [27] emphasize the complexity of balancing costs, resource allocations, and completion times in these scenarios, while Van Der Merwe [28] explores the interplay between functional and project managers in managing work. Payne estimates that up to 90% of all projects are run in this context and often lead to complex matrixed organizational structures [6]. These characteristics are seen in the case study organization in Chapter 3.

Kang and Hong [29] explain the competition for limited resources between projects and the resulting increases in queue time as each project waits for resource availability, even with close attention to resource allocation. This dynamic highlights the importance of reducing queue time to accelerate project delivery. In particular, they explain how this creates competition for limited resources between projects and increases queue time in each project waiting for those resources to become available, even with close attention to resource allocation. This dynamic makes reduction of queue time important to accelerate projects. Important to note in the switch from discussion of work management in single projects vs. that of multiple simultaneous projects is the shift to thinking of even scheduled work as existing in queues awaiting scarce resources. Jensen et al. developed a model depicting the interactions between "work stacks", which could be between individuals focused on incidents (repair / reactive work) and project tasks (maintenance / proactive work), between teams with different skills, or both [30]. This is a critical management function to model, as queue time is often directly related to the amount of 'ticket-passing' between individuals within a team and even more so between teams. Antoniol et al. also discuss the treatment of project work tasks by queueing [31].

Patanakul and Milosevic [32] discuss the unique demands on project managers who manage multiple efforts simultaneously, which often have unrelated goals and stakeholder needs. They highlight the need to manage the interdependencies between the projects, which if nothing else can include demand for the same staff resources, and the need for strong multitasking skills. They explicitly recognize the complexity inherent in managing efforts of differing levels of importance, complexity, and novelty. Finally, they recognize the effect of shifting costs to managers of multiple projects. In practice, these characteristics also apply to functional managers responsible for resources in a skill-based organizational structure that balances project and operational work, as discussed by Fricke and Shenbar [33]. Diao and Hecheng acknowledge similar management overhead in the context of coordinating operational tickets between teams [34]. Platje and Seidel [27] discuss the need for operational managers to delegate more to subordinates under conditions of high operational uncertainty, such as that created by the need to support multiple types of work and priorities. Rahmandad and Weiss [35] emphasize the interactions between projects and the need to develop “slack” in resource capability to be able to absorb changes in priorities and demand, and warn that there are tipping points with sustained schedule pressure. Finally, Jensen et al. [30] developed a model depicting the interactions between “work stacks” – which could be between individuals focused on incidents (repair/reactive work) and on project tasks (maintenance

/ proactive work), between teams of different skill levels, or both. This bridges the gap between project and operational work outlined in the next section.

2.2.3 Dynamics of IT Service Management

The prevalent framework for managing work based on events that occur within the organization, generally classified as “incidents” and “requests”, is the IT Information Library (ITIL). Incident and request management in IT organizations is routinely managed through queue-based ticketing systems, with queues assigned to individuals and teams. These were successfully modeled as queueing systems by Bartolini et al. using their SYMIAN simulation [36] and treated by other researchers [37]. ITIL was developed in Great Britain in the 1980’s and has served as the de facto standard for IT infrastructure and operations since the publication of version 2.0 in 2000. Version 3.0 was released in 2007, and version 4.0 was released in 2019. All versions of ITIL since 2.0 have treated the management of incidents and requests as a queueing problem.

According to version 4 of the ITIL framework, ’the purpose of incident management practice is to minimize the negative impact of incidents by restoring normal service operations as quickly as possible’ [38]

Voyer et al. [39] developed a model of major incident management that can be used as a basis for one major type of unscheduled work. In cases where automation can be used, this model would predict improvements in ’response coordination’ and associated improvements to downstream work quality. This model does distinguish between temporary fixes (“workarounds”) and what ITIL refers to as “irreversible corrective action” (“resolutions”); however, for the purposes of the model constructed later in this research, a workaround will be considered a partially completed work effort, which will be returned to the queue until a full resolution can be completed. Voyer’s complete stock and flow model would predict the downstream improvement to ’time to correct errors’ based on improvement in ’efficiency of coordination.’ It would also indicate an opportunity to improve work quality by increasing accuracy and consistency of implementation through automation. Finally, there is an opportunity to improve the "Major Incident Resolution Rate" through increased response to events through automation.

Wiik and Kossakowski [40] developed a model of incident management that specifically incorporates the benefits of automation applied to information security response activities. This is a reasonably detailed incident response model that incorporates the impact of automation by shifting a percentage of work off human staff. In this model, automation simply reduces the fraction of incidents that require manual intervention, improving productivity, and reducing staff needs (and, by extension, the associated labor costs).

Neither of the models above reflects the differentiation of skills, in terms of skill *level* or skill *type*, and therefore do not address the routing of tickets between individuals or teams due to incorrect initial assignment or the need for multiple teams to collaborate to complete the ticket. Discussion of the dynamics of a multilevel (skill) service desk operation is discussed by Fenner et al. [41], and treatment of these issues resulting in ticket re-routing / reassignment is addressed by Li et al. [42].

Oliva developed a request management model that can be used as a basis for the second major type of unscheduled work [43]. Automation increases 'Service capacity', which in turn decreases 'Work pressure'. This should flow through to increase the "potential order fulfillment rate" and increase the rate of "orders processed", but as modeled it is not due to the structure of the "Time per order", as it does not consider the impact of context switching time when "work pressure" is high. Context switching is an area that could be positively impacted by automation. This model can be used with adaptation to ensure the expected downstream impact of an increase in "Service capacity" to the rate of "Orders processed" and "Labor effectiveness."

Automation would most directly impact this model by increasing "Service capacity" – at least for activities that can indeed be automated. By increasing "Service capacity", "Work pressure" is reduced, which in turn reduces 'work intensity' and counter intuitively reduces 'work effectiveness'. In addition, automation can increase the rate of "orders processed" for some of the requests received, with a corresponding reduction in the "Service Backlog" and downstream reduction in the "Work pressure." Although not explicitly treated in their model, note that the concept of "work pressure" as represented can be interpreted as impacting the relative *priority* of work items. This is an important consideration in the assignment of tickets in any operational model as discussed by Li et al. [44] and can also be fruitfully extended to project tasks.

2.2.4 Modeling and Simulation

Systems and System Dynamics

A system is defined as "a collection of elements and a collection of interrelationships among the elements such that they can be viewed as a bounded whole relative to the elements around them" [45]. Organizations are well researched as complex adaptive systems [46], and exhibit varying levels of complexity through feedback loops, which are often poorly understood and can lead to highly non-intuitive outcomes during their operation [47]. Systems thinking is a collection of methodologies that allow for consideration of the "whole" system, including its constituent parts and interactions [48]. Work management within the organization under study meets these criteria.

System Dynamics (SD) is a rigorous methodology that has been successfully used in various contexts to model the dynamic behavior of complex managerial and organizational systems such as those considered here [49–52]. The models referenced in the sections below each address separate aspects of the work dynamic within a healthcare IT organizational model, but none fully explore all types of work commonly serviced by these teams and only partially identify the leverage points which automation can potentially address. As such, they will be used as the basis for a consolidated model that better highlights these areas. Each of these are expanded below, along with their contribution to the model for this study.

Discrete Event Simulation

Discrete Event Simulation (DES) is a fundamentally different approach to process modeling based on queueing theory. Models essentially depend on several key concepts: *entities* (along with attributes that can be assigned to entities), *resources* (such as queues and servers that act on entities), and *activities* (including routing between resources based on attributes and action taken by servers). In addition, *attributes* track any changes in entity state that occur during specific *events* (such as entry into a queue or completion of service) [53].

A key distinction between DES and SD is in the word "discrete": Each entity is distinct and events occur at discrete points in time, while SD assumes a continuous flow through the model controlled by rates of change, which are determined by differential equations [54]. Another element that sets DES apart from SD is that it is inherently stochastic in assigning key elements of the model [55]. Key model settings such as interarrival times, service times, and, if needed, the values of entity attributes are determined through probability distributions. Finally, DES models are considered predictive, with complex queuing and routing systems displaying emergent behavior over many iterations, while SD models are considered descriptive of the effect of causal loops on the underlying queuing system.

The applicability of DES to operational work management is obvious, as operational tickets are commonly managed through explicit queueing systems, and from a practical point of view, project tasks can also be considered to be queued, as discussed below. DES allows the construction of highly valid and verifiable models of the management of work in environments such as the case study organization.

Hybrid Modeling

These models can be used together to retain the accuracy and predictive capabilities of statistical queueing within DES models while also adding the broader descriptive capability of the SD model [55, 56]. For the purposes of this analysis, the goal is to obtain a clear understanding of the effects of the dynamical influences on the queue time and total throughput time of entities in the DES model [57]. The conceptual "metamodel" for the integration of the two models closely follows the approach discussed by Viana et al. [58], with the exception of using Matlab instead of Simul8 for the DES model.

Hybrid modeling requires explicit modeling of key influences in the SD model as entity attributes in the DES model, so that these can be adjusted in subsequent iterations of the models based on the results of previous simulations. Chahal et al. [55] provide a conceptual framework for the integration of the modeling methodologies that are followed in this research. It is theoretically possible to combine the two methods by using a tool such as AnyLogic, which inherently enables both model types). However, in this research, MathWorks SimEvents is used for DES while Vensim is used for SD modeling, and data is passed between them manually (initially) and ultimately in an automated fashion following each tool's simulation run. This is referred to as *cyclic interaction* in [55] as opposed to *parallel interaction*. A third, viable modeling option also exists: once the system dynamics model is built and the influences clearly understood and the influence equations established in the Vensim model, these influences can be (re)built within the SimEvents model leveraging Simulink blocks.

Note that a trade-off develops as the cycles are shortened between the model iteration frequency and clarity in the representation of the dynamic interactions. As the frequency of iteration increases to allow more frequent interaction between the two models, certain " slow" dynamics (those that evolve over longer periods of time than the cycle lengths, as well as delayed effects) can no longer be simulated exclusively within the SD model. If both tools continue to be used, these dynamics may not be explicitly represented in *either* the SD or DES model, but instead are represented in the mechanism used for integration between the two models. Morgan et al. [59] discuss this hierarchy of model timing in their study of a radiology clinic, with the SD model providing the larger / longer-term framework for the clinic and the DES model addressing day-to-day operations. This trade-off appears to be inherent in the distinct ways the two modeling methods handle time (e.g., continuous vs. discrete). Borshchev [60] discusses this issue in some detail along with the methods used within AnyLogic to overcome it.

These issues are discussed in more detail in the context of the specific research problem in Section 3.4

2.2.5 IT Infrastructure

IT infrastructure (also referred to in the literature as "information infrastructure") refers to the hardware, software network, and other tools on which enterprise applications are deployed.

An infrastructure can generally be considered as a single complex adaptive system [61], and an enterprise IT infrastructure shares these characteristics. Common domains of IT infrastructure include servers, storage, databases, firewalls, networks, data centers, cloud services, and end-user computing (laptops, desktops, and mobile) [62, 63]. Each of these domains is composed of many individual complex systems that are both operationally and managerially independent, so it is appropriate to also consider the IT infrastructure a complex system of systems [64].

The IT infrastructure of an enterprise is initially established through an architecture process, which determines the overall design of the IT environment and the logical and physical integration between them. Since this is generally done when an organization is very early in its life cycle (and therefore relatively small), this process is often ad hoc and minimalist. The architecture also establishes the technical standards for the infrastructure as well as the vendors and products that will be used. At the next level of detail, specific systems and products must be designed (or engineered) and deployed according to architectural road maps. Unfortunately, IT systems engineers are rarely able to design *de novo* or ”green field” infrastructures with stable requirements in their careers.

As the organization evolves and technologies come and go over time, the infrastructure must also evolve, necessitating a continuous architecture process resulting in technology road maps that guide change in each domain. Systems and subsystems are upgraded, replaced, or retired, and new ones are added. It is not unreasonable to ask whether IT infrastructure is similar to the mythical Ship of Theseus: after all the components are replaced, while it still has the same purpose, is it still the same ship? It will likely not bear much resemblance to the original infrastructure when the organization first started. ”Top-down” design activities are completed by IT systems engineers using life cycle processes that are generally aligned with those of formal systems engineering as defined by International Council on Systems Engineering. However, there is also a high degree of ”bottom-up” evolution of the infrastructure that occurs with the introduction of new capabilities by the product providers, as well as the obsolescence of older components by those same vendors.

Architectural road maps must accommodate and allow both sources of change and adapt to the requirements imposed by the infrastructure that exists at that time [63,65–67].

It is important to note that infrastructure components / subsystems are deployed in *physical space* – whether within a physical rack in a data center or in different offices across the globe – and that a clear understanding of various locations where the infrastructure is deployed is of critical importance to IT engineers. In addition, these subsystems are predominately purchased from vendors, with design characteristics that differ between products and product configurations, as well as over time. These differences can be important to track as the infrastructure evolves as they can become constraints on the deployment of future capabilities.

Common IT Infrastructure Challenges

According to Hanseth and Lyytinen, there is a high degree of complexity inherent in managing the existing system of systems they refer to as the Information Infrastructure [63]. They point out that unlike traditional system design activities, where requirements are established through a life cycle that results in a *de novo* system, infrastructure is rarely a "green field" and is instead an example of managed evolution from an installed base. This evolution occurs through a series of cross-departmental, cross-skill activities that change subsystems within the infrastructure. It is also an area where various components of the infrastructure become obsolescent on different time scales, and certainly much more quickly than the overall infrastructure itself [68].

The implementation of changes is usually contained within a specific domain silo, so IT systems engineers in other teams are often unaware of the changes outside their domain, especially in a large organization. The effect of what may be episodic changes within individual domains can be a high velocity of overall infrastructure change, especially in a large organization. These are driven by large numbers of concurrent projects that drive changes to specific systems within the infrastructure, as well as execution of request- and incident-driven operational changes to various system configurations. Grisot et al. refer to these as innovation *in* the infrastructure (”replacing or modifying existing components”) and *on* the infrastructure (extending the infrastructure with new components) [66].

Published evaluations on current practices regarding the documentation of IT infrastructure are limited, although more work has been done in the context of enterprise architecture (EA) [69]. Anecdotally, in the absence of architecture-specific EA tools, documentation is primarily handled through a variety of management tool sets — often product- or technology-specific — in conjunction with static documents (such as Visio diagrams or Excel spreadsheets), which may or may not be version-controlled and can proliferate in multiple versions within an organization. In fact, architectural frameworks such as Zachman [70] and TOGAF [71] explicitly or implicitly rely on the production of document artifacts and viewpoints. These information repositories are not integrated and therefore, often not shared across domain specialties. Additionally, documentation creation remains labor intensive pending the maturation of generative AI for this use case, especially at scale [72].

IT Infrastructure Documentation

Generally, IT systems engineering exhibits the following documentation practices for the infrastructure:

- There are no standards for IT infrastructure design and support documentation in terms of how systems and structures should be represented. Various IT vendors (such as Google, Cisco, and others [73]) have popularized consistent representations through reference manuals and training. In addition, these vendors provide various stencils for use in representing their products in different diagramming tools.

- Any standards that may exist are often organization-specific and highly idiosyncratic, but can be distinct within each technical domain within an organization.

- Certain process frameworks (such as Scaled Agile's SaFE [74]) address the need for conceptual deliverables but do not dictate specific forms, formats, or tools. For example, the SaFE methodology discusses the use of UML-based artifacts (for example, domain diagrams), but often specifies text artifacts for epics, features, stories, and enablers. These text artifacts are often supported through various Agile-focused toolsets.

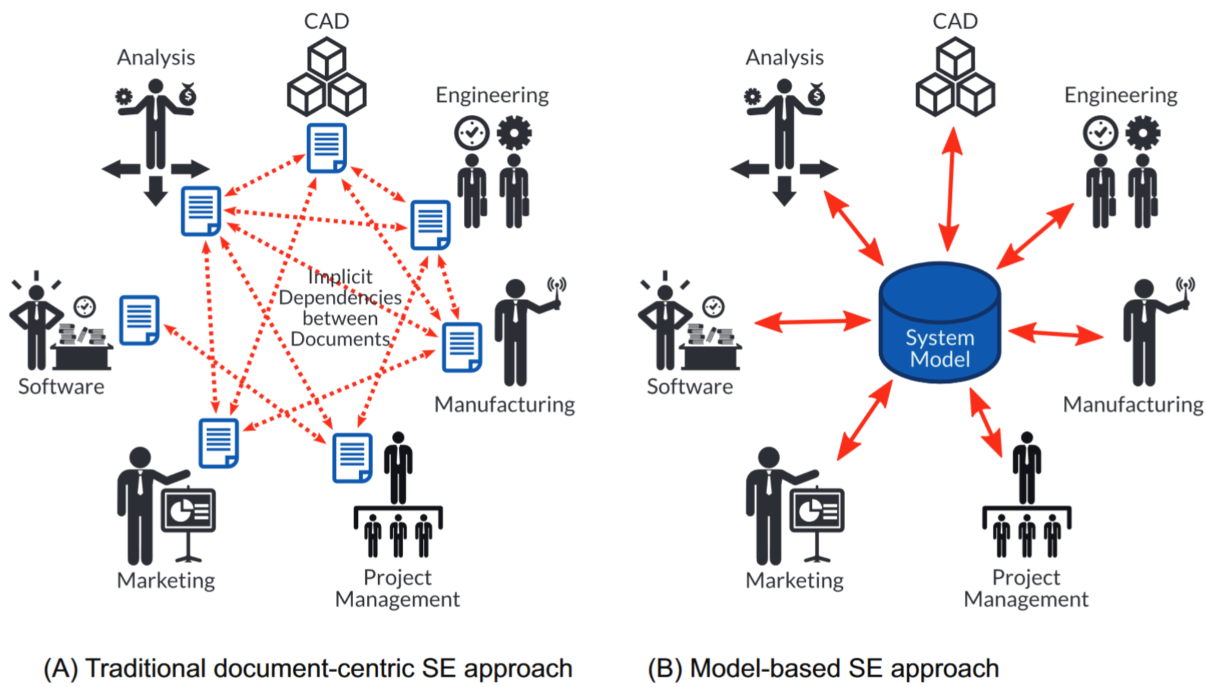

Figure 2.1: Document-based or traditional approach to SE compared to MBSE.

- Visio diagrams and those from other drawing-only programs are ubiquitous, but these are point-in-time documents [75]. Although Visio can support some level of data access, this is relatively uncommon in practice. Visio is commonly used by IT systems engineers to visually diagram locations within the infrastructure. In addition, many vendors provide comprehensive stencils for their products and services for use within Visio, making it easy to visually identify products in a diagram. Such diagrams vary in content and style by organization, department, and even by engineer.

Call and Herber [76] elaborate the practical differences between Document-Based Systems Engineering (DBSE) and MBSE, summarized clearly in Figure 2.1. Kotusev [77] cites Lohe and Legner in outlining the problems of document-heavy approaches within IT in the context of enterprise architecture. Although not specific to the IT infrastructure, these are consistent with other sources in highlighting the potential value of MBSE.

IT Infrastructure and "The Cloud"

As mentioned in Chapter 1, NIST states that cloud services have the following characteristics:

- On-demand, self-service
- Broad network access
- Resource pooling
- Rapid elasticity or expansion
- Measured service

Vaquero et al. discuss the differing perspectives on cloud services in more detail, and arrive at the following definition, which is worth including in full:

’Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms, and / or services). *These resources can be dynamically reconfigured to adjust to a variable load (scale), allowing also optimum resource utilization.* This pool of resources is typically used by a pay-per-use model in which the Infrastructure Provider by means of customized SLAs” [78]

In short, "the Cloud" and especially IaaS, can be usefully thought of as a method for providing IT infrastructure for consumption. When provided to customers within the enterprise, it is referred to as a "private cloud". Automation is a critical capability of any private cloud that underpins this method.

2.2.6 Automation Technologies

The relevant technology areas are highlighted below.

Software-Defined Networking

There is a significant body of literature that covers the design of SDN in terms of equipment and enabling systems and APIs (e.g., OpenStack) as well as how best to meet the requirements for performance, resilience, and security. However, there is limited information available on the decision criteria to adopt and deploy SDN or in the supporting changes required to maximize the value of the technology. The practical implementations of SDN have been in the implementation of two specific network 'overlays', data center networks and wide area networks, with the aim of increasing the security and resilience of these critical areas. The research in this area will be further elaborated in the final dissertation.

Infrastructure Automation and Infrastructure-as-Code

The research in this space to-date has focused on the development practices to effectively leverage the APIs exposed by infrastructure providers (either on-premises or in the public cloud), especially regarding quality and security. The literature is focused on the technical implementation and optimization of the technologies by developers – generally in the context of both DevOps and public cloud - not on the decision criteria regarding adoption or the organizational changes required to successfully leverage. Further, infrastructure automation is shown to rely on a high level of systems standardization, based on the common analogy of systems (especially servers) being managed as "cattle" (larger numbers but essentially indistinguishable) as opposed to "pets" (individually unique). The research in this area will be further elaborated in the final dissertation.

Cloud Adoption

There has been research done in this space that covers some of the criteria decision makers should use to assess whether cloud technologies are appropriate, and the most appropriate deployment model (e.g., public, private or hybrid). There does not appear to be specific coverage of the organizational changes required to successfully leverage the technologies, beyond an occasional nod to DevOps and Agile methodologies. This research does address the needs of certain industries, and in particular the adoption of the public cloud for electronic medical records and similar clinical applications. The research in this area will be further elaborated in the final dissertation.

DevOps and Variants

Extensive research has been done regarding the adoption and subsequent management of DevOps and its variants (DevSecOps, NetDevOps, etc.), especially in the context of public cloud services. The literature makes clear that the success of DevOps and concepts such as Continuous Integration and Continuous Deployment (CI/CD) absolutely requires cloud technologies that deliver capacity that is programmable, elastic, and on demand. These could be provided through any variant of cloud services deployments - public, private, or hybrid. Furthermore, essentially all the literature on DevOps assumes the existence of an internal software development organization with the tools and skills to enable the 'shift left' (e.g., the transition of operational functions to developers). The research in this area will be further elaborated in the final dissertation.

2.2.7 SysML and MBSE

Systems Modeling Language is a modeling language developed by the Object Management Group (OMG) as an open specification to enable "the specification, analysis, design, verification, and validation of a broad range of systems and systems-of-systems." [79] Of particular interest, SysML is designed as a UML 2 profile to be flexible enough to model both software *and* hardware components, such as servers and network equipment, which is a critical difference between the concerns of software and IT systems engineers as the physical world introduces many new constraints - especially when procurement logistics are involved. However, to date SysML has not been adopted by IT systems engineers despite the ability to design and document the hardware domains. There is limited discussion of the application of SysML and MBSE to civil infrastructure projects [80] and evolutionary systems-of-systems [81]. In addition, there are some academic papers that outline the use of SysML for the purpose of designing individual enterprise IT systems [82–85]. However, there is no existing literature that addresses the application of these tools and practices to the broader enterprise IT environment and data centers.

Now, MBSE is defined by INCOSE as a "formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases." [86] MBSE is tool-enabled, with a centralized database that enables collaboration by different specialty engineers against a unified view of a system, and hence provides a shared source of truth for "as-built" and multiple potential "to-be" system designs. MBSE is most commonly adopted in organizations building complex systems that require collaboration across engineering disciplines such as mechanical and electrical engineering, notably defense, aerospace and automotive [87]. Interestingly, the survey results indicate some initial (but decreasing) adoption since 2012 by organizations that identify themselves as being in the IT *industry*, but there is no indication of use by the IT *function* within an organization. However, there is no inherent reason MBSE could not be used to build and maintain IT systems built and maintained by server, storage, and network engineers, other than those common to all adoption efforts, including cost, complexity, and old-fashioned resistance to change [88]. Note that MBSE has been successfully adopted in traditional waterfall and Agile design environments.

SysML and MBSE are complex tools and processes in and of themselves - the costs and effort to adopt and maintain them over time as part of the IT management processes is only justified under certain conditions - when the perceived value outweighs the barriers to adoption listed by Clouthier, Chami et al. nderson and Salado reviewed the extant literature regarding the benefits of MBSE in 2021 [89], and find only two papers claiming robustly measured benefits and a larger number citing observed benefits which can be useful from a practitioner standpoint. Most articles on the subject of benefits are claimed without evidence, and those that provide it are primarily observational and highly subjective. More recent analysis by Campo et al. [90] supports these conclusions and goes further to discuss claims in the literature regarding barriers, notably increased cost, time, effort, and complexity during the adoption of tools and methods. Campo et al. also note that claimed costs are better justified through case studies than benefits. According to a survey conducted in 2018 by Huldt and Stenius, any benefits are realized most often during the early phases of the design process [91]. Constraining the discussion to only *measured* and *observed* benefits, these can be broadly summarized as improved quality (specifically in terms of error reduction and design consistency) and efficiency of the engineering process (esp. concerning traceability and collaboration), and these are only realized post-adoption of the tools and methods. In combination with the front-loading of costs and back-loading of benefits (relative to adoption), these characteristics predict that MBSE can benefit long-running architecture and design efforts more than short-lived initiatives initially, although all can eventually benefit once the tools and methods are adopted across the organization.

The combination of complex system-of-systems with a high degree of cross-departmental and interdependent involvement makes IT infrastructure a candidate for MBSE-enabled design and management, due to the reported benefits to collaboration and system quality. Furthermore, the characterization of IT infrastructure as an evolving system implies that early-phase system engineering activities (esp. requirements, architecture, and design) are performed regularly in some subset of the technical environment, where MBSE is reported to provide the most value. In particular, certain aspects of IT infrastructure are extremely cross-departmental but are regularly involved in both project- and operationally driven change and constantly evolving as a result:

- *Data Integration* includes the transactional interfaces between interconnected systems (leveraging standards such as EDI or HL7), as well as bulk data transfers (Extract, Transform, and Load, or ETL), replication, and aggregation in support of analytics and AI initiatives. There are almost no systems in a modern data center that are not connected to others in some way.
- *Information Security* includes several key subsets of particular interest:
  - The engineering and management of identities within internal and external systems.
  - 3rd party (vendor) risk engineering, in response to external systems integration as well as partner access to internal systems.
- *Data Center Engineering* includes the design, upgrade, migration, and recovery of both on-premises and cloud-based application hosting environments. The next section will discuss the deployment of a software-defined infrastructure within a data center as a case study.

2.2.8 Architecture Descriptions and Reference Architectures

Architecture Description

ISO/IEC/IEEE 42010:2022(E) somewhat unhelpfully defines an architecture description as a "work product used to express an architecture", with architecture defined as "fundamental concepts or properties of an entity in its environment." It then further elaborates that the architecture includes the entity's constituent elements, interactions between them and with other entities in the environment, it's behavior and structure, and principles governing it's design, use, operation, and evolution. [92]

The architecture should include the definition of the System of Interest (SoI) and environment; stakeholders with their concerns and perspectives; and architecture considerations, views, and viewpoints from the point of view of the various stakeholders. The Model-Based System Architecture Process (MBSAP) [93] applies the methods of MBSE to successively elaborate a system architecture and will be followed in Section 4.

Reference Architecture

Borky and Bradley define () as ”a logical / functional abstraction that defines the features and behaviors common to a domain or class of entities. ” Soares et al. define an RA as a "high-level design solution for a class of similar software systems belonging to a given domain," which is based on an architectural analysis of the target domain and solution requirements; consists of synthesis of these requirements, the domain concept, and organizational styles and patterns; and an evaluation of the quality attributes for the solution [94]. The MBSAP specified Operational and Logical / Functional Viewpoints are most appropriate for the representation of a reference architecture as defined above, while the Physical Viewpoint is better suited to a concrete instance of an architecture.

Data Center and Cloud Reference Architectures

Most common conceptual architectures focus on issues such as hierarchical service models ("x as a Service"), with for example IaaS being a foundational service, with PaaS built on top of that and ultimately SaaS at the highest level of abstraction [95, 96]. Tsai et al. propose a conceptual architecture that addresses the need for certain additional capabilities provided in layers, such as the ”Cloud Broker Layer” and the ”cloud Ontology Mapping Layer” intended to enable cross-vendor management [97].

Vendor-specific architectures are provided to enable IT engineers to consume their products or services and is focused on detailed implementations - either to build services within a cloud provider's environment or to build private clouds on premises using common enterprise vendor products (for example, VMware, Oracle, Cisco, IBM, etc.) [98–100]. All are designed to remove barriers to purchase and help IT properly implement the underpinning infrastructure, and in some cases nod in the direction of integration with existing enterprise tools (IBM under the umbrella of "Platform Services", or Cisco with "Service Orchestration") [101]. However, there is a general lack of research from the viewpoint of the enterprise IT leader on how to realize these subsystems in combination with commonly deployed management tool sets to provide automated services. For example, none of these addresses how an enterprise monitoring solution or ticket management system would be integrated. NIST Special Publication 800-146 comes closest to enumerating these with the discussion of the provisioning / configuration function within the "Cloud Management Service" [102].

Youseff et al. discuss this need at a high level in their ontology as the "cloud Software Environment Layer", from the standpoint of APIs provided to developers to enable these functions, but do not discuss where these APIs come from (other than the "cloud service provider") [103]. Torkashvan and Haghighi propose an "Intelligence as a Service" layer for cloud services, comprised of an Event Control Agent and a Service Execution Agent, to enable programmatic response to events that could occur in a cloud environment [104]. These are important elements of an on-premises SDI but still highly conceptual from the point of view of providing guidance to IT on how management tools must be integrated and coordinated to perform these functions.

Chapter 3

Simulation Modeling of the Work Environment

The elaboration of the characteristics of the organization under study, including modeling and simulation, is completed in this chapter. This includes details regarding the modeling and simulation of a single skill-based team, **although additional elaboration regarding justification of certain SD model parameters and more extensive verification and validation of results remains in process.**

3.1 The Case Study Organization

The case study organization is a national healthcare provider in the United States with a large number of acute and non-acute care hospitals, ambulatory clinics, and physician practices. The application environment is a combination of on-premises and SaaS-based systems with a substantial physical infrastructure. At the time of this study, central IT consisted of approximately 700 staff, organized by technical skill (network, security, etc.) and function (project management, application analyst, etc.). Operational requests and incidents are managed in ServiceNow (a leading ticketing system), and projects are managed through several different applications and tools. There were several hundred active projects of various sizes underway, with dozens of tickets of both types varying complexity arriving daily.

Large healthcare providers in the United States are generally assumed to share the following characteristics that affect the structure of their IT organizations.

- Relatively low margins with increasing erosion due to evolving industry dynamics driving high rates of mergers, acquisitions, and divestitures and resulting high levels of system variability and high pressure to reduce staff costs.

- Earnings-driven incentive systems that discourage operational costs in favor of capital investments and drive:

  - Significant dependence on capitalizable on-premises applications and commensurately on long-lived "mutable" systems.
  - Commensurately low reliance on cloud-based infrastructure, systems, and associated work management methods (e.g., DevOps).
  - Prevalence of waterfall project methods over agile approaches (due to the above factors).

- High and increasing operational and clinical dependence on IT system and data performance and availability as direct contributors to clinical quality and safety, which has driven:
  - Ubiquitous adoption of ITIL 2/3, especially for the help desk, incident, and request management processes.
  - The increasing number and complexity of the projects focused on maintaining regulatory compliance, gaining cost savings, improving quality of care, and supporting innovation.

These combine to create a complex technical environment and organizational structure, with a large volume of operational tickets and the need to simultaneously pursue a variety of projects ranging from routine to critical transformational efforts. The conclusions drawn from this research may have a broader applicability to organizations or industry segments with similar characteristics.

The models developed in this paper depict only interactions with a single team. The modeled team supports mixed work types (e.g., both unscheduled and scheduled). Individuals can be primarily assigned to one or the other work type, but can work on either if priorities necessitate at any given time. Although researchers have demonstrated the preference to protect scheduled work from unscheduled demands, this is not always economically feasible. Each team supports multiple concurrent projects / products at different stages of planning and execution, as well as multiple concurrent incidents and requests. Due to the limited number of resources with particular skills, work of both types is queued awaiting completion. The models are intended to reflect the flow of work over relatively short timelines and do not address the ability to flex staff through hiring (or contract outsourcing) within the time window under analysis. The validity period for the models is six months.

Resources may have specific tasks and/or tickets assigned to them in some cases and may, in other cases, pull work from a team queue based on perceived priorities. The assigned resource or a manager can make the decision to stop or reassign any work for the reasons outlined below. Active work in progress can be returned to the queue due to 1) requiring a higher skill level than the assigned resource; 2) being interrupted by higher priority/urgency work; or 3) being "reassigned" to another team's queue due to a lack of certain technical skill in the originating team. Note that this can happen multiple times with a given piece of work, even within a single team; when multiple teams are involved, a work item can spend significantly more time in queue than being actively worked.

3.2 Modeling the Work Environment of a Single Team

3.2.1 The Single Team System Dynamics Model

The model summarized in this work is built using Vensim PLE. It is intended to incorporate key elements of previous models that specifically highlight areas where automation may be of benefit. As discussed above, the time horizon for the analysis is too short to allow adjustments to resource availability through new hires or sourcing arrangements.

Beginning with project work, the base loops and flows are shown in green, where the boxes represent the primary flow of project work, the green flows represent the "happy path" of work through the system, and the red flow represents work that is returned to the queue (work stops in this diagram). With respect to arrows, red arrows represent negative influences on the performance of the process, green arrows represent positive influences, and blue arrows are neutral.

3.2.2 Full Single-Team Model

Subsequent elaborations add the following aspects until the complete single-team model is reached in Figure 3.1:

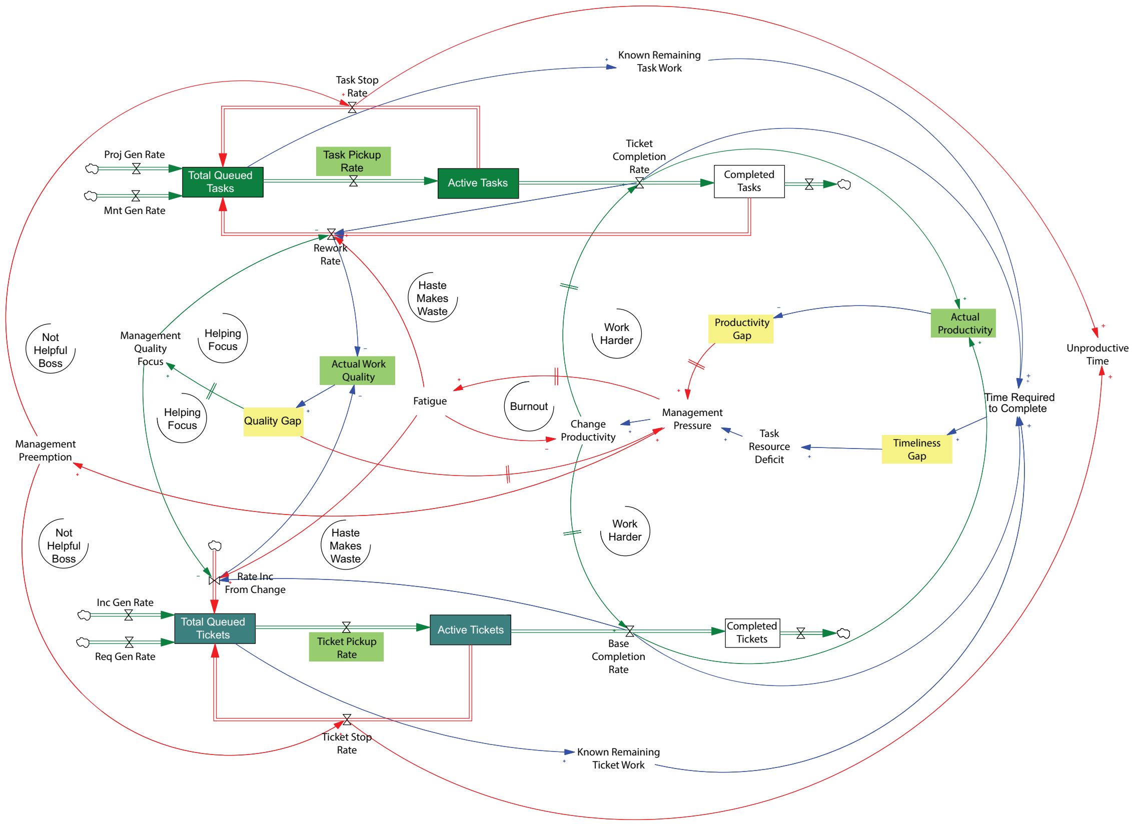

Figure 3.1: Vensim system dynamics model of a single team reflecting both project and operational workflows.

- Both operational work and rework (including incidents from change), including interactions between the different types of work. The operational work is shown below the project work in teal boxes.
- The effects of management response, the introduction of the possibility of a gap between desired and actual work quality, with a delayed response by management that can be positive (through constructive assistance) or negative (through pressure causing fatigue).
- The effects of resources having to stop work on a particular task / ticket and shift to another introduce the concept of switching costs which have a negative impact on overall productivity.
- The influence of “timeliness” corresponding to the on-time delivery of project tasks and the rapid fulfillment of operational tickets.

3.2.3 The Single Team Discrete Events Simulation Model

Compared to the system dynamics model, the DES model is superficially much simpler; however, the complexity is embedded in the attributes of the entities and the routing rules based on them.

Greasley recommends clearly defining the scope of a DES model, including assumptions, abstractions, and areas deliberately left out of scope [57]. In order to maintain a manageable level of complexity, no additional teams are included; however, this is an abstraction, as, in reality, several teams can be involved in even relatively simple and frequent tickets or tasks. Teams are modeled with accurate staffing in terms of numbers, skill level, and type of skill of team members. These team members are modeled as *servers* in SimEvents. Note also that this model is the more appropriate place to deal with the issues of (re)prioritization and the impact of skill level mismatches between the task / ticket and the assigned resource on error rates, as these issues are more complicated to model in Vensim. The model is shown in Figure 3.2, with work generators on the left, a team queue to consolidate the work types, and four individual parallel queues and engineers ("servers"). Work stoppages, where the task or ticket requires more time than the server has available to complete, are sent back to individual queues, and completed work is forwarded to the termination points on the right. Note that the probability of errors is captured through a signal from each termination point and generates incidents that are rerouted to the team queue.

The data to support determination of the inter-arrival, total duration, and completion data for operational tickets are derived from six months of actual service desk system data and then fit to specific Poisson distributions using Matlab fitting functions for each team and ticket type. These are modeled as negative exponential distributions for stochastic generation within the DES model.

Data related to tasks were estimated through interviews with department leaders and also modeled as Poisson distributions, and this decision requires some justification. **There is some discussion in the literature distinguishing the inter-arrival times and service times of project tasks as distinct from operational tickets**. In the case of arrival times, the distinction is made that while tickets arrive randomly, tasks are planned. With regard to service times, the argument is that project

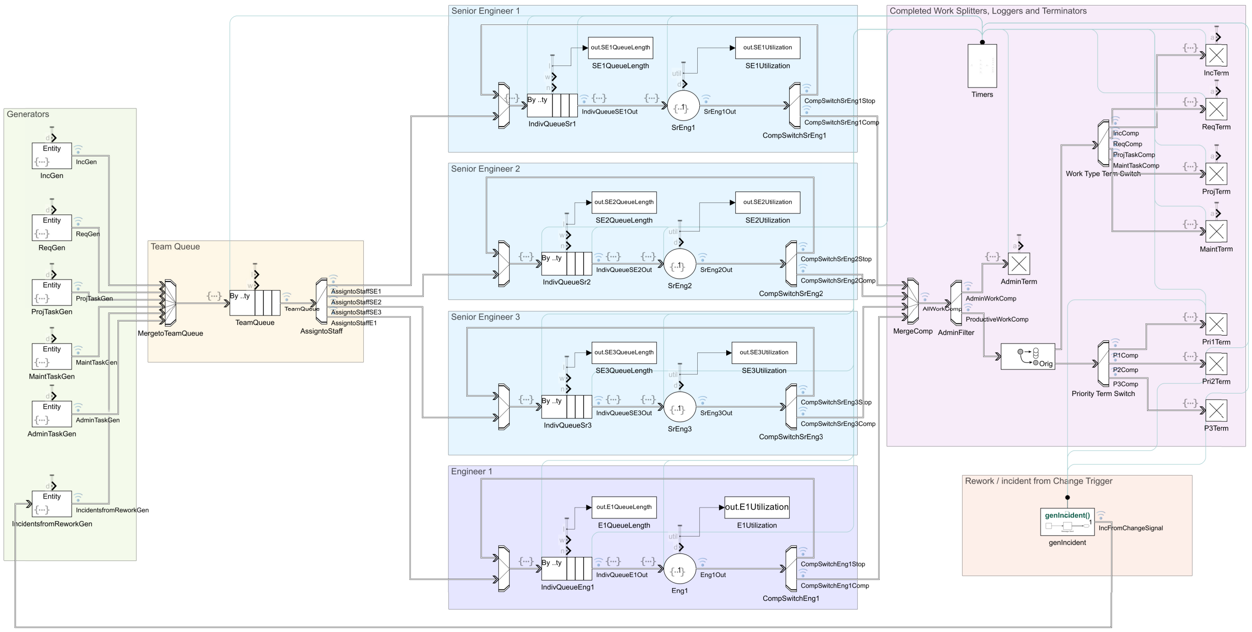

Figure 3.2: Simevents queuing model depicting entity generators for each work type feeding four engineers with individual queues.

tasks are generally more complex than operational tickets, and as a result, have a longer required service time. These claims are made in the specific context of software development teams and, unfortunately, are largely anecdotal. For the purposes of this research project and maintenance tasks are also modeled as also having Poisson distributions for arrival and service time with the following justification:

- The case study organization had over 200 concurrently active projects at the time of this study. Although task *identification* and *assignment* are often completed during specific waves (during initiation or as part of an iteration or agile sprint), this is completely different from their *due dates*, which is closer to the time the tasks will be complete. In combination, the arrival of tasks from the perspective of a shared-services team is perceived as essentially random within many teams in the organization.

- The specific team modeled for the case study is not a development team but a technical infrastructure team. Although there may be substantial differences in service time for new development tasks that require new and creative methods to solve versus maintenance tasks (or a "bug fixes"), leaders in the team do not see such a difference in the deployment, configuration, or reconfiguration of infrastructure technology.

Tickets and tasks of different types are modeled as entities in SimEvents, with independent generators driven by appropriate distribution settings. Baseline data related to distributions of skill type and level required to complete tickets / tasks, and other attributes that drive statistics of routing are based on estimates from interviews with department leaders.

In order to fully model the throughput of the team, the model also incorporates administrative / " nonproductive" time. A non-trivial amount of time (the literature suggests up to 1/6th of a resource's time) is regularly siphoned off into activities that don't contribute to the completion of either tickets or tasks such as filling out time sheets, attending town halls, or participating in team-building exercises. These activities are also estimated with Poisson distributions for both arrival and time spent on them.

3.2.4 Interaction Points Between Models

As recommended by [57], the following interaction points are defined between the DES and system dynamics models:

- The completion, rework, and preemption rates from the SimEvents model are fed into the Vensim model by adjusting the associated work generation rates in the stock-and-flow diagrams.

- The effect of fatigue driven by an increasing work intensity in the Vensim model results in an increasing probability of rework and incidents from changes over the simulation time, which is fed back into the SimEvents model directly.

- The effect of increasing management pressure in the Vensim model is fed back into the SimEvents model as an increasingly frequent interruption of in-process work (i.e., having to stop a task / ticket), modeled through a proportional decrease in the engineers' available service time.

The cycle is iterated to determine changes in the performance of the queueing process under the influence of changing dynamics. These changes are finally analyzed to determine the impact on the model (in terms of the completion rates) on the dynamic attributes driven by management interventions and resource responses to changes in pressure over time, as well as the sensitivity of the changes to changes in specific attributes.

3.3 Single-Team Simulation Results and Discussion – Long Iterations

The initial data runs coupling the two models were set to 260 working days - roughly a full year, less weekends. The purpose of this was simply to determine the relative direction and strength of the effects in each model.

The base SimEvents model demonstrates that the team can adequately and quickly handle high- and medium-priority tasks and tickets (within a day and two working weeks, respectively), but that low-priority work completion times continue to increase. These queues build monotonically throughout the simulation (ranging from 140–220 days by the end of the simulation, with an average of 88 days). This is consistent with observations from historical ServiceNow data during certain periods – the department analyzed is not necessarily in equilibrium. However, these results are dependent on the estimated arrival times for tasks (project, maintenance, and administrative) as well as the required and available service times for each event; this will be explored more fully in a later section on model uncertainty.

Although the Vensim model does not represent the differences in priority, the same steady increase in queuing was observed. The additional influences introduce different behaviors over time, including oscillations in quality, timeliness, and productivity that flow through to the observed behavior of the queues and flow rates.

The dynamic behavior observed in the error generation and stop rates has a strong impact on subsequent long iterations of the SimEvents model. After feeding back the changes from the Vensim results to the SimEvents model in the second iteration:

- 1/3 more work items were stopped and requeued due to management pressure, and there was a very large increase in Incidents from Rework.
- This resulted in a 25% increase in the work completed in reactive incident response (because there were so many more) as well as an increase in all completion times of 18% for P1 and a 125% increase in P2.
- For all intents and purposes, many P3s simply remained in queue with 75% less completed during the simulation.
- These differences are shown over time in the difference graphs in the top corners of Figure 3.2) – the graphs show the increase in days to complete work over time between the baseline and the next iteration.

In essence, these interactions create a new reinforcing loop between the model iterations that drives increasing queue times, especially for medium- and even high-priority work.

3.4 Single-Team Simulation Results and Discussion – Short Iterations

The long iteration times are unfortunately not realistic – as described above, this is the equivalent of generating a year's worth of queuing effects, using those to generate a year's worth of dynamic effects, then cycling those back into the DES model to generate another year of queuing effects. In reality, the two models should interact in real time: the short-term, process-level view in the DES model is influenced over longer time frames by the dynamics modeled in SD. The following section will explore the process of adapting the models to enable shorter cycles and the results obtained.

Automating the Models

The DES model can be easily automated using Simulink scripting, with inputs to and outputs from Vensim driven through Microsoft Excel leveraging built-in functions. The script does the following:

- Call the SimEvents simulation model.
- Input the parameters for the error rate (project task rework and incidents from change). These are initialized in the Matlab script during the first iteration and updated by the previous Vensim run for subsequent iterations.
- Assign the input parameters to the model and execute the simulation.
- Analyze the results of the model and generate statistics.
- Output appropriate parameters to an Excel file (to be used by the Vensim model as input).

The SD model can also be parameterized in a similar manner, with the input parameter file loaded during model initiation appropriate starting values loaded to specific variables. The Vensim model is called using a Windows command-line script. On completion, the model exports the result data from the Vensim.vdfx file to Microsoft Excel to be used as input for the next iteration of the DES model. The results from each iteration of both simulation models are maintained in Matlab throughout the exercise.

Integrating and Iterating the Models

Due to it's flexibility, Simulink is used as the base system for calling and running both the SimEvents and Vensim models, iterating between them and generating Microsoft Excel files with cumulative statistics. This is accomplished through the following procedure.

- Set the number of iterations as a variable.
- Set the initial parameters for the first iteration of the SimEvents model.
- Call the SimEvents script referenced above.
- Log results into a history table in Matlab, with each iteration appending to the table.
- Export the results from SimEvents to initialize the next Vensim iteration.
- Call the Vensim script referenced above, loading parameters generated from the results of the SimEvents model run.
- Export Vensim results to an Excel file for Matlab to ingest both historical analysis between runs as well as to initialize the next iteration of SimEvents.
- Embed the sequence of activities above into a "for" loop, to continue for the number of iterations specified above.

The duration of each model's run is specified in the individual model configurations, but this can also be parameterized through the Simulink script. Figure 3.3 illustrates the integration and iteration logic.

Reducing Iteration Length

As noted in Section 2.2.4, with an arbitrary decrease in the iteration length to 20 days (one working month) or shorter, certain dynamical processes in the system configured with specific time delays longer than the iteration length or with longer periods to effect the impact must be modified. At this point, there are three viable approaches to enabling shorter iteration cycles:

- Continue modeling the system using the two distinct methodologies and tools, adapting the integration process accordingly.
- Shift to a single tool (such as AnyLogic) that enables native representation of both methodologies within a single model.

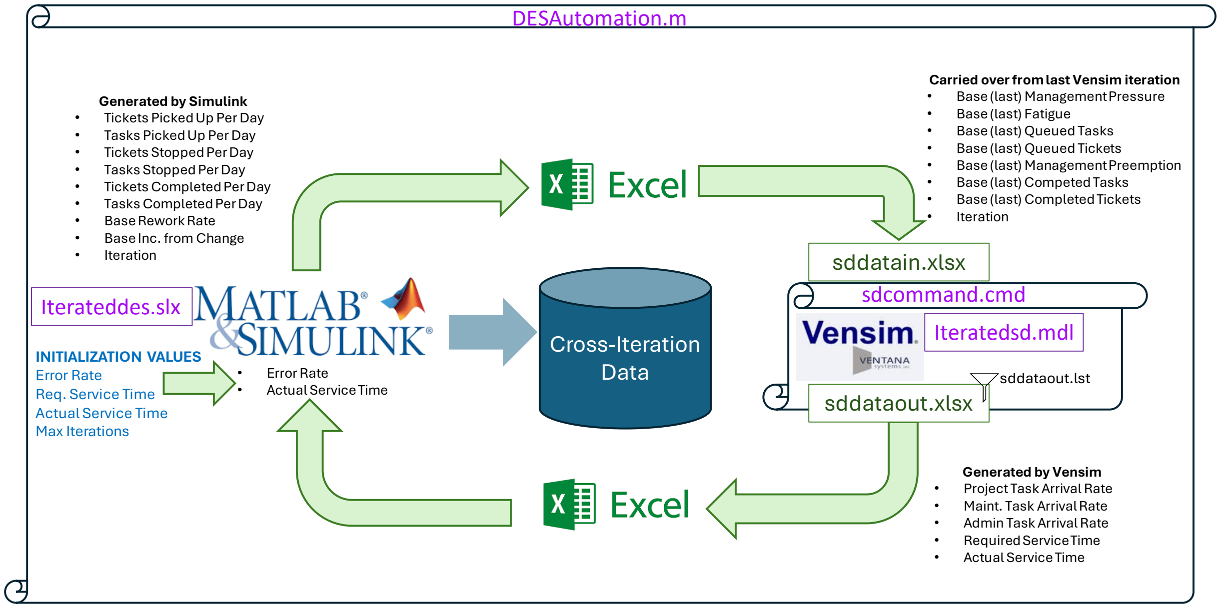

Figure 3.3: Scripting, parameters, and logic flow of iterations.

- Shift all queuing and dynamic functions into one of the original tools - for example, use Simulink blocks within the EventSim to model the system dynamics.

Each of these is technically viable with different trade-offs. For the purposes of this research, AnyLogic was eliminated because of the increased cost. In order to establish a pattern for leveraging Vensim and EventSim together, this research continues to push the limits of tool integration at the cost of increased complexity in that integration.

As iterations become shorter and more frequent, the parameter passing process via Excel files has to be expanded to allow for the passing of state between iterations of the SD model so that in practice the multiple iterations of the system dynamics model resemble a single, long-running iteration within which multiple iterations of the DES model run.

As an example, the Rework Rate is used to capture the occurrence of errors in project tasks that lead to additional unplanned work that must be done within the project to complete the original tasks correctly. This is initialized at the start of the simulation iteration through the Base Rework Rate variable and is then dynamically modified through the end of the iteration. This ending value of the Rework Rate must be carried through to the next iteration's Base Rework Rate variable, rather than being initialized repeatedly to the starting state of the first iteration. This greatly increases the number of parameters that must be configured for integration between tools, initialization within each tool, and tracking over time to enable analysis and true understanding of dynamics.

Additionally, the ability to model delays within the Vensim model (e.g., variables that don't take effect until after specific periods of time) must be reconsidered. In this research, Managerial Pressure is an example of a dynamic variable originally configured with a (arbitrary) delay. The intent of the delay setting is to recognize that managers do not immediately start intervening in the conduct work, but generally only do so after a period of time. The shortening of the iteration cycle forces reconsideration of this dynamic: in reality, managers step in as either quality erodes or timelines for work completion stretch beyond those deemed acceptable. The result can be considered a "phase transition" or step function in management intervention, with essentially none before certain thresholds are passed and increasing amounts afterward. The conclusion is that these variables are better modeled with a dynamic dependency on reaching specific thresholds in other modeled variables, such as the Rework Rate or the Average Queue Time.

The scripts used can be found in Appendix A.

3.5 Uncertainties in the Models

3.5.1 Data Gaps

There are several gaps in the available data in the case study organization that affect the simulation results in terms of utilization and queueing over time, notably:

- The frequency (inter-arrival times) of project and maintenance tasks.
- The frequency of administrative / nonproductive tasks.
- The available service times for tickets or tasks.
- The required service times for tickets or tasks.

These gaps are driven by three key shortcomings in current work practices in the case study organization outlined below.

- Time accounting is currently not done to determine how much of each engineer's time is dedicated to operational tickets, project tasks, or administrative activities. Hence, it is difficult to establish a true utilization of resources.

- Ticket completion times are understood only from the time the ticket is opened to the time it is closed (total duration). The amount of effort spent (service time) on any given ticket is unknown.

- Task completion times are not tracked at all. Project tasks are defined during planning but are only tracked against due dates. The difference between them can be very large and is essentially incomparable to the total duration of the tickets.

The case study organization is planning the implementation of a resource management solution to close the first and third gaps above but no usable data was available during the conduct of this research.

The issue of capturing the actual service time (the second gap above) is not a limitation of the ticketing system in use. Rather, the effort required of engineers to accurately estimate their actual time spent on a ticket is not considered worth the potential benefit to gathering the data.

3.5.2 Data Estimates and Simplifications

With the help of organizational leaders, it is possible to define reasonable bounds for uncertain data elements. In the case of administrative tasks, as mentioned previously, there is support in the literature for an estimate of up to 1/6 of staff time being absorbed with such activities. However, all estimates should be considered highly idiosyncratic to the team under consideration. Table 3.1 contains the team leaders' estimates, with the variables corresponding to the exponential coefficients used in the SimEvents model.

Table 3.1: Managerial Estimates for DES Inputs

Dependent Variables	LB	UB	X0	Comments
Required Service Time	0.0600	0.2553	0.2553	Between 30 min and $\tilde{2}$ hours (115 min) required, on average, per task (event)
Available Service Time	0.0600	0.2400	0.1229	Between 30 min and $\tilde{2}$ hours (115 min) available, on average, per engineer (server)
Interarrival rate of admin tasks	0.0638	0.2553	0.2334	Between 4 and 16 tasks per day for the team, on average
Interarrival rate of project tasks	0.0213	0.3424	0.3424	Between 3 and 46 tasks per day for the team, on average
Interarrival rate of maint tasks	0.0639	1.0270	1.0270	Between 1 and 16 tasks per day for the team, on average

For simplicity, the service times for each type of work are assumed to be the same. Although it is reasonably simple to allow for different required service times by type of work (to address observations in the literature that project tasks are more time-intensive, for example), there are no data within the case study organization to justify the additional complexity. This can of course be added to the models in the future should a theoretical basis for it - or actual data - demand it.

3.5.3 Validation of Estimates

To determine the reasonableness of these estimates, a series of regression tests were performed using a sum-of-least-squares approach. A high-level schematic of the regression logic is shown in Figure 3.4. The dependent variables are defined in Table 3.1, with lower bounds (LB), upper bounds (UB), and initial estimates (X0) given. The code for the regression functions is shown in Appendix B, and constructed to call the script with the simulations. Due to the stochastic nature of the DES simulation, nonlinear regression functions were selected.

Initial regression attempts focused on the built-in Matlab function lsqnonlin, a gradient-based optimizer. Unfortunately, it was never able to converge to even a local minimum, despite tuning the various thresholds, the algorithm (Levenberg-Marquardt and trust-region reflective) and the max number of evaluations, as well as starting from multiple points of the dependent variables. The

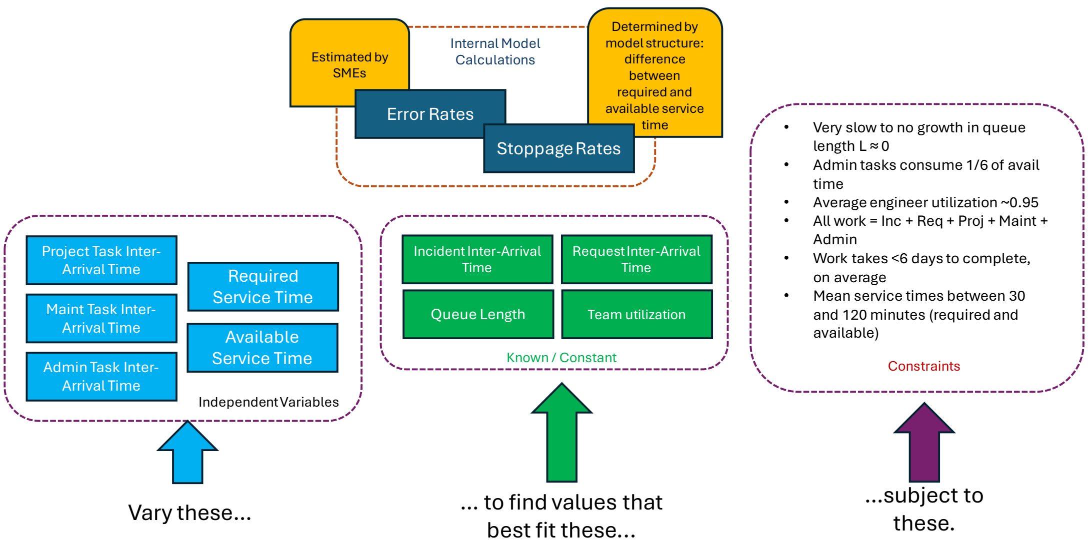

Figure 3.4: Schematic of regression analysis.

Table 3.2: Regression Targets

Independent Variables	**Observed**	**Comments**
Interarrival rate of incident tickets	0.2025	Determined from ServiceNow actuals
Interarrival rate of request tickets	0.0931	Determined from ServiceNow actuals
Utilization	0.95	Estimated to determine maximum throughput before queuing begins
Queue depth	4	Estimated to determine maximum throughput before queuing begins

resulting residuals and optimal values for the dependent variables never stabilized. According to Rheinhart (Section 1.7) [105], this is common when the objective function or its derivatives has discontinuities, multiple optima, stochastic response or ” flat spots” - any of which are possible leveraging a queuing model. Also, while the inter-arrival rates of the three task types (project, maintenance, and administrative) are modeled separately in this research, as they are all rates of work, they can be combined (added) into a single value which represents a three-dimensional set of possible solutions, as long as their sum remains the same - creating a ”flat spot”.

To overcome this issue, direct search techniques were explored, specifically pattern search and particle swarm methods. These are both much more compute-intensive and longer-running processes that require tens of thousands of iterations, particularly for problems where the results of the objective function are highly nonlinear or rugged, as is the case here. Furthermore, external calls to Vensim during simulation execution prevent the use of parallel processing, resulted in an iteration length of roughly 12 seconds, and sometimes resulted in Vensim ”hanging” during execution, stalling the process. In order to speed up the process of determining the optimal values for the dependent variables at the throughput threshold, the Vensim call was removed from the regression script, and only the SimEvents output from a single iteration was leveraged. The results of both direct methods are shown in Table 3.3.

Table 3.3: Optimal Variable Results from Direct Search

	Pattern Search	**Particle Swarm**
Sum of Least Squares	3.0113	1.9681
Available Service Time	0.1959	0.3423
Required Service Time	0.3722	0.5994
Interarrival rate of project tasks	0.2344	0.2411
Interarrival rate of maint tasks	0.1170	0.1543
Interarrival rate of admin tasks	0.2296	0.1713

These results imply a total task arrival rate of between 20.5 and 21.5 per day for the pattern search and particle swarm methods, respectively. However, the calculated ticket arrival rates are low compared to the actual ServiceNow ticket prices, as outlined in Table 3.4. Assuming that the calculated arrival rate for all work is reasonably correct and that the uncertainty is in the relative distribution of the types of work, the implied ”true” value of the task inter arrival rates is between 15 and 16 per day at the threshold where the team is fully utilized and before queuing begins. Therefore, any combination of project, maintenance and administrative task volume that amounts to 15-16 per day is likely to be directionally accurate. Further, assuming that 1/6th of all time is spent on administrative tasks implies a combined project plus maintenance task volume of between 11.5 and 12.5 per day.

Table 3.4: Regression Analysis

	Pattern Search			Particle Swarm		
Calculated Values	**Coeff.**	**Work Items / Day**	**Diff from Actuals**	**Coeff.**	**Work Items / Day**	**Diff from Actuals**
Interarrival rate of Incident Tickets	0.3998	2.5013	55%	0.4089	2.4456	54%
Interarrival rate of Request Tickets	0.5015	1.9940	186%	0.5986	1.6706	155%
Interarrival rate of all tickets	0.2225	4.4953	80%	0.2429	4.1162	73%
Interarrival rate of all tasks	0.0582	17.1686		0.0607	16.4662	
Interarrival rate of all work	0.0462	21.6639		0.0486	20.5824	
Actual Values						
Interarrival rate of Incident Tickets	0.2202	4.5403				
Interarrival rate of Request Tickets	0.9307	1.0744				
Interarrival rate of all tickets	0.1781	5.6147				
Implied "True" Values						
Interarrival rate of all tasks	0.0623	16.0491		0.0668	14.9677	
Assumed admin task interarrival rate	0.2770	3.6106		0.2915	3.4304	
Implied project + maint task interarrival rate	0.0804	12.4385		0.0867	11.5373	

3.6 Modeling the Work Environment of Two Teams

No team operates in a vacuum. In reality, each IT team relies on other teams to complete the work, resulting in additional complexity in both queuing and system dynamics. These interactions have effects that can present additional opportunities for process automation.

Adding a second team to both the SD and DES models introduces additional rules for managing the work that passes between them.

- The models only depicts interactions between two skill-based teams – interactions become much more complex as additional teams become part of the work process, as would be the case with a real cross-functional process such as server provisioning, which can cross multiple departments, technologies, and tools.

- Each team supports mixed work types, for example, both unscheduled and scheduled. Individuals on each team could normally be assigned to one or the other type of work but can work on either if priorities require it at any given time. Note that while researchers (including Rahmandad and Weiss) have demonstrated the preference to protect scheduled work from unscheduled demands, this is not always economically feasible.

- Each team supports multiple concurrent projects / products at different stages of planning and execution, as well as multiple concurrent incidents and requests. Due to the limited number of resources with particular skills, work of both types is queued awaiting completion.

- The models are intended to reflect the flow of work over relatively short timelines, so there is no ability to flex staff through hiring (or contract outsourcing) within time window under analysis – in other words, there is no ability to increase labor capacity.

- Active work in progress can be returned to the queue, due to 1) requiring a higher skill level than the assigned resource; 2) being interrupted by higher priority / urgency work; 3) it can be 'reassigned' to another team's queue due to a lack of certain technical skill in the originating team. Note that this can happen multiple times with a given piece of work even within a single team; with multiple teams involved, a work item can spend significantly more time in queue than being actively worked.

- Reassignments between teams require coordination to ensure efficient completion and is an indicator of a higher level of overall complexity. Work that requires multiple reassignments to complete requires a commensurately higher level of coordination; however, that is often unlikely to happen except for extremely high-priority work.

The expansion of both models to simulate the interaction of multiple teams is planned as a subject of future research.

3.7 Improvement Focus Areas

The models reinforce several common sense improvement targets, such as reducing *completion times* and improving *responsiveness* for operational tasks, improving actual *work quality* and increasing *pickup* and *completion rates*. Similarly, the hybrid model predicts that reducing the rate of *rework* and the generation of *new incidents* and reducing distractions (including preemption caused by managerial pressure) that interrupt work completion and increase *switching costs* will have strong effects on work performance. Less intuitively, the models demonstrate trade-offs between these elements under circumstances of unchanging team capacity.

Strategies for Improvement

The models indicate that the interaction between project and operational work – and likely between teams with differing skills – creates a coupling of work queues and wait times within those queues: each shift of tasks / tickets within and between queues adds queue time to the work. The following strategies can be used independently and in combination to improve outcomes with respect to the improvement targets described above.

- Ensure close coordination between project and functional leaders. Given the complexity involved in large organizations, this quickly becomes impractical at scale where the volume of work and the number of teams combine rapidly.
- Limit the number of projects in the process through a governance and prioritization function, that is, portfolio management. Note that a large organization can have dozens or hundreds of active projects of various sizes and states of implementation even with mature governance. High-priority operational work (esp. incidents affecting production) is essentially impossible to control in this manner, and critical issues often preempt scheduled work.
- Separate operational and project responsibilities between different staff within each department to reduce the amount of work transferred between those work-type queues.This requires larger teams within each skill to support at a higher labor cost to the organization, but does allow project staff to focus on scheduled work and protect it from disruption by operational issues.

- Hire and (and train) multi-skilled resources who can reduce the need to transfer work between department queues. In practice, this quickly becomes expensive; not all resources have the capability to cover multiple technical domains, and those who do are highly recruited externally and must be actively retained.

- Create cross-functional teams to prevent work transfers between department queues. This is more routinely seen in project-driven areas and organizations, and particularly in larger projects. It is also common in product-focused organizations that follow DevOps methodologies. This is difficult to scale in organizations with large active portfolios. In addition, some skills are only needed for small portions of each project, leading to centralization of these skills and offering them as a shared service.

- Automate repeatable and high-volume work to improve response and task completion times and allow staff to focus on unique activities. Automation can also improve quality (assuming adequate testing) by ensuring consistently accurate outcomes, which is especially important for tasks that happen regularly or at scale. Automation can focus on tasks that involve multiple skills and involve multiple departments by reducing the amount of queue time and the amount of management effort required to coordinate completion. This is very common in organizations that use DevOps.

The increased use of automation, in particular, can address several improvement areas simultaneously, as demonstrated by its use in organizations that use DevOps-style methods.

Key Metrics

Finally, the model identifies several key metrics for leaders to understand delivery performance and which of the strategies above may provide the biggest improvements in performance at any given point in the organization's journey. Again, several of these are reasonably straightforward, such as the difference between actual and expected service delivery quality, in terms of both fitness for purpose and fitness for use; the difference between actual and desired staff / team productivity, as measured by task and ticket completion rates as well as the amount of rework created.

Other metrics are less obvious but perhaps more easily managed and include: reassignment and requeuing counts used as indicators of how many times work has been started and stopped; the number of tickets / tasks assigned to teams and individual staff as a proxy for understanding the amount of "work juggling" and the resulting switching costs; and finally the rate of errors resulting in rework and or new incidents resulting from the previous changes.

Identification of Automation Targets

The models highlight several areas which represent targets for automation, summarized here:

- Reduce overall completion times by significantly reducing or eliminating queueing times.
- Improve responsiveness to predictable work items, especially after hours, where skilled staffing is often reduced or primarily on-call.
- Improve actual work quality due to increased accuracy, completeness, consistency, etc. of the work completed, with an associated reduction in the rate of rework generation (under the assumption of high-quality / well-tested automation).
- Reduce switching costs and cross-team coordination by partial or full automation of complex cross-team processes (such as server provisioning).
- Significantly increase various pick-up rates for automated activities, which in turn drive improvements in the scale of work that can be completed in any given time period.
- A final potential target for automation is in work that requires the sequential use of several tools / applications by the assigned resource to complete repetitive tasks, which could be automated together in sequence to improve individual productivity. This is a common target for the types of tools commonly referred to as RPA, or "bots."

In combination, these areas allow organizational leaders to help identify and prioritize opportunities for investment in automation. The specific circumstances of each individual organization will come into play in terms of assessing the relative priority of automation opportunities, both in terms of feasibility and overall ”bang for the buck.” The ”bucks” in this context determine the growth of the underlying SDI platform, and which components of the infrastructure are incorporated into the automation framework.

A canonical example of a process improvement opportunity that spans many of these areas is the automation of server provisioning, including in the book that helped popularize DevOps: *The Phoenix Project* [106]. This is a routine process that in any moderately-sized organization involves multiple skills and tools crossing several teams that must be completed in sequence and are complex enough to lead to substantial variability in execution. The next section will use this process as the driving example that determines the sequence that tools and infrastructure components are incorporated into the SDI.

Chapter 4

Developing an SDI Architectural Framework with MBSE

This section will explore the architecture of a software-defined infrastructure management system capable of orchestrating the administrative functions of an enterprise hybrid data center infrastructure, such as deploying software updates (patches) on a scheduled basis or responding to certain types of incident driven by events. The overarching goal of this architecture is to enable the automation of IT use cases that address the opportunities outlined in Chapter 3. This section successively elaborates the architecture of the SDI following the Model-Based System Architecture Process (MBSAP) outlined by Borky and Bradley [93] using the SysML. For the purposes of this research and to ensure applicability to the entire class of automation solutions in the on-premises data center, the level of analysis will be limited to the Operational Viewpoint (OV) and Logical Viewpoint (LV) as discussed in Section 2.2.8 above. Each viewpoint consists of the following perspectives (where warranted):

- Structural Perspective consisting of block diagrams.
- Behavioral Perspective consisting of use case, activity, and sequence diagrams.
- Data Perspective consisting of Conceptual and Logical Data Models.
- Services Perspective consisting of taxonomies and specifications.
- Contextual Perspective with additional documentation and illustrative graphics.

The goal of this chapter is to develop a reference architecture for IT leaders to leverage in the creation of on-premises SDI.

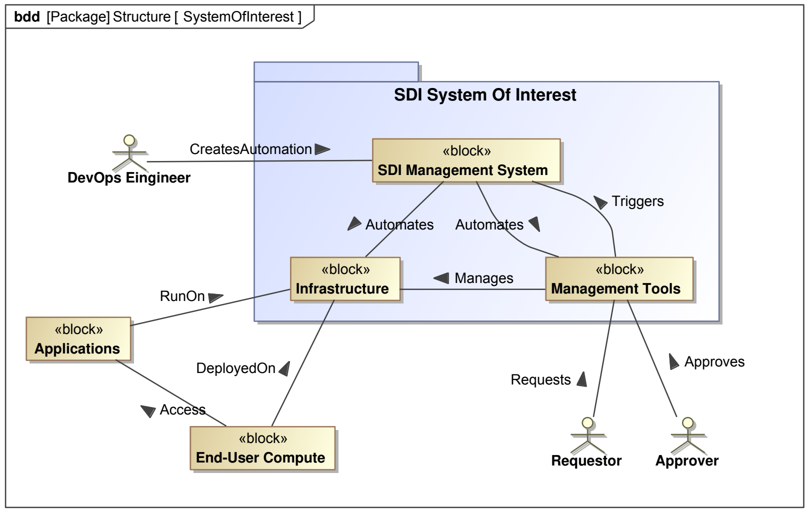

Figure 4.1: SDI System of Interest.

4.1 System of Interest

The general architecture of the system for SDI is composed of existing technical infrastructure and management tools which are API-enabled, along with a subsystem that will be described in this research as the SDI Management System. The three elements of infrastructure, management tools and the SDI in combination represent the System of Interest (SoI) as shown in Figure 4.1. It is the SDI Management System that provides the glue that programmatically ties together the pre-existing infrastructure and tools to enable automation of the IT use cases.

4.2 Solution Requirements

A functional SDI actually represents a platform for automation - what an organization decides to automate within it depends on the analysis of where improvements are most required, the basis for which is discussed in the final section of Chapter 3.

4.2.1 High-level Requirements

Figure 4.2 represents the highest level of functionality the SDI solution, and Figure 4.3 represents the highest level of non-functional requirements (business and performance).

Name	Text
1 Execute Code on Request	the system must be able to execute code to accomplish tasks within the managed infrastructure and management systems based on a request received from a user through a ticketing system
2 Execute Code from Event	the system must be able to execute code to accomplish tasks within the managed infrastructure and management systems based on events received from those systems
3 Execute Code on Schedule	the system must be able to execute code to accomplish tasks within the managed infrastructure and management systems based on a defined schedule
4 Manage Code	the system must enable the management of code that accomplishes tasks by system admins and developers, including publishing, activation, and decommissioning
5 Interface to Management System	the system must provide the ability to execute actions in external management systems (such as directories, monitors, ticketing systems, etc.) via APIs established by those system vendors
6 Interface to Infrastructure	the system must provide the ability to execute actions in virtual and physical infrastructure via APIs established by those system vendors
7 Manage Defined Interface	the system must provide the ability to add, update and remove defined interfaces
8 Log Activity	the system must log provide the ability to log all internal functions, and provide a logging capability to code executed within the system
9 Monitor Activity	the system must provide the ability to provide alerts regarding its activities
10 Manage Identities	the system must provide the ability to add, modify and remove identities used to execute tasks via interfaces

Figure 4.2: High-level functional requirements.

Name	Text
11 Secure Identities	the system must ensure that secrets associated with managed identities are not compromised.
12 Secure software repositories and pipelines	the system must provide the ability to ensure only authorized users can add, modify, or delete code
13 Support high availability	the system must provide the ability to continue operating in the event of component failure
14 Support disaster recovery	the system must support the ability to restore in an alternate location in the event of a disaster
15 Provide adequate responsiveness	the system must support timely execution of code following events and requests, or on a defined schedule

Figure 4.3: High-level non-functional requirements.

4.2.2 Example System Provisioning Requirements

The use cases for an SDI The management system can be considered from two perspectives:

- That of its direct users, e.g., the security admins, system admins, and developers that need to accomplish tasks within the environment.
- That of the tasks that are automated by those users to meet business needs.

For the purposes of this chapter, this paper will focus on the latter, as they serve better to illustrate the structure and function of the system. The subsequent elaboration of the SDI architecture will enable the use case of provisioning infrastructure capacity to support the deployment of a new application. The provisioning function is selected as it is a commonly automated activity in the public cloud that incorporates many of the targets of automation, it addresses many of the opportunities identified in the simulation models, and represents a solid base from which to build automation to address subsequent use cases. The requirements have been derived / expanded in Figure 4.4 and are mapped to the high-level SDI requirements in Figure 4.5.

Name	Text
16 Automate system provisioning on request	Enable users to request systems for provisioning, consisting of compute, storage, network and software resources
16.1 Specify system configuration	Enable collection of system specifications
16.1.1 Specifiy application name	Select application name linked to the application inventory, or create a new application name in the inventory
16.1.2 Specifiy compute configuration	Specify memory and processor or select from standard options
16.1.3 Specify storage configuration	Specify storage type and capacity or select from standard options
16.1.4 Specify network configuration	Specify network attributes
16.1.5 Specify operating system	Select standard operating system image or leave blank
16.1.6 Specify software to install	Specify standard software to install or leave blank
16.8 Accept request	Accept request from ticketing system for execution
16.8.1 Forward request for approval	Submit new requests for technical and financial approval within 10 minutes
16.10 Approve request	Submit request to approver and obtain approval / denial decision, with conditions for selective automatic approval
16.10.1 Forward approved request for provisioning	Submit approved request to the provisioning system within 10 minutes of approval
16.11 Provision capacity	Provision compute, storage and network capacity following approval in the appropriate order
16.12 Install operating system	Install selected operating system onto each provisioned server
16.13 Install software	Install automatic and optionally selected software onto each provisioned server
16.11.1 Install standard software	Install required software such as patching, security, and accounting packages
16.11.2 Install optional software	Install other software on specified servers if required and packages exist in software repository
16.14 Roll back activities	Roll back provisioning activities if comopute, storage or network capacity not available
16.15 Log provisioning activities	Log all actions taken by the provisioning process, including successes and failures
16.17 Complete provisioning activities	Complete provisioning activities within 2 hours of approved request being submitted to system
16.18 Execute in DR environment following declaration	Ensure availability of system and execution code in DR environment to enable automated provisioning there after failover
16.19 Manage provisioning code	Create, update and delete code components as needed
16.20 Secure provisioning code	System must secure code and configuration information remain confidential and are not altered except to authorized users / processes
16.18.1 Secure software repository	Ensure confidentiality and integrity of software packages and images, and enforce change control processes
16.18.2 Secure acces to interfaced systems and infrastructure	Ensure confidentiality and integrity of identities / credentials used to interface with management tools and infrastructure
16.18.3 Secure provisioning pipeline	Ensure integrity of code and execution environments through defined development processes

Figure 4.4: Automated provisioning requirements.

4.3 Operational Viewpoint

According to the MBSAP, the OV establishes the baseline of needs and requirements for the system, and explores the concept through the use of high-level context and use case diagrams.

The SDI Management System under consideration must automatically coordinate with the various external / preexisting tools in the environment (such as event monitoring, workflow request, IP Address Management (IPAM), Domain Name Service (DNS), etc.) as well as the infrastructure itself (servers, switches, etc.) to accomplish its tasks. The system is bounded by the combination of technology that serves as the platform for automation, as well as the custom code built within it leveraging external APIs to manipulate the underlying infrastructure and tools (outside the system boundary). This system will enable the shift of administrative use cases to more automated execution by orchestrating these APIs against key IT infrastructure technologies and tools.

The specific performance requirements of the system will vary depending on the administrative function that is being automated. As stated in the previous section, the system will initially focus on the provisioning of server, storage, and network capacity within a data center environment in the context of a newly requested and purchased application. The primary sponsor for the SDI Management System is the CIO for the organization, and the key stakeholders are the teams who

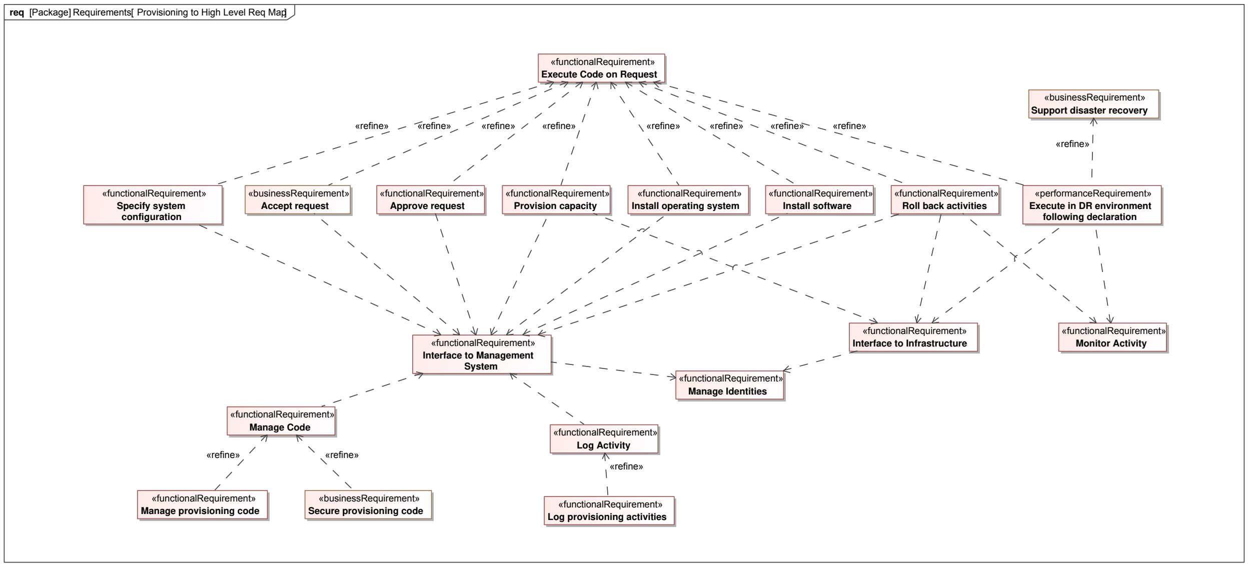

Figure 4.5: Mapping of provisioning requirements to high-level SDI requirements.

deploy and support these systems manually today. This particular use case is not time-critical; however, the system must ultimately be capable of both request-based and event-driven / real-time activity. Figure 4.6 presents a Block Definition Diagram (BDD) context diagram for the system, which is semantically equivalent to the more common Visio diagram shown in Figure 4.7

As discussed previously, organizations remain heavily dependent on purchased COTS applications with minimal custom development creating isolated pockets of critical data that must be cobbled together through transactional integration and scheduled data extract, transform and load ETL processes to enable consolidated decision making. Others have a much higher percentage of critical systems which are custom-developed, either on-premises or in the cloud. Historically, these applications and their underlying components have been provisioned and built "by hand," with IT engineers with various skills manually deploying servers, assigning storage, and installing operating systems, databases, and other application components. Enterprise IT is increasingly shifting towards SDI to manage these environments, which is the combination of public, private, or hybrid cloud technologies with management via automation. SDI and its subtopics of software-defined networking, DevOps, infrastructure automation, and IaC are well-studied in the context

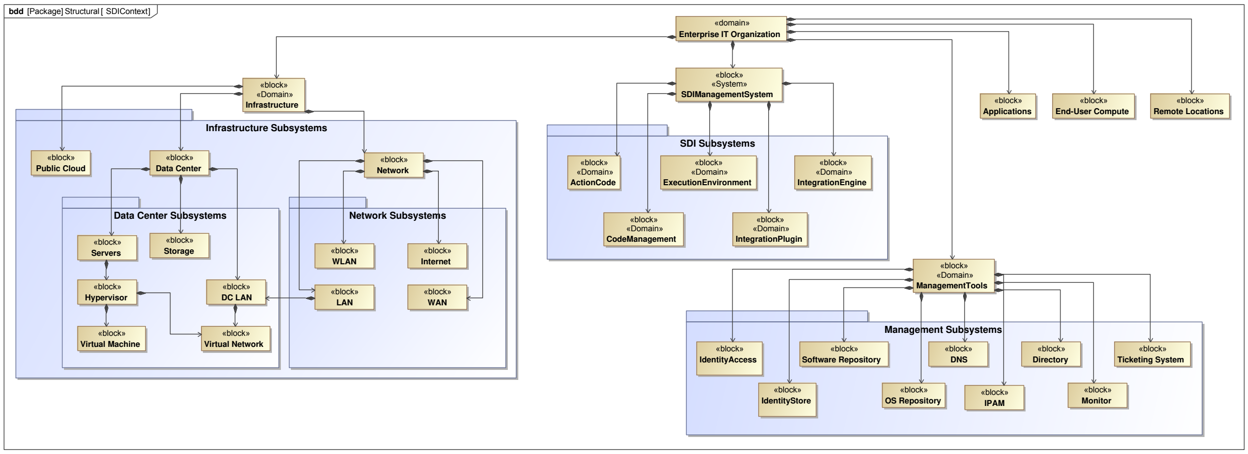

Figure 4.6: SDI Management System in the context of existing technical infrastructure and management tools.

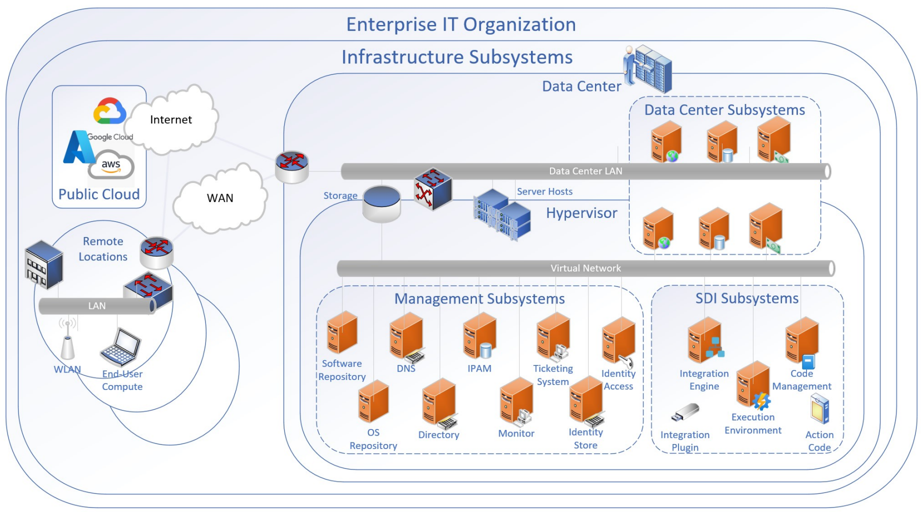

Figure 4.7: Equivalent Visio Diagram of SDI Management System in the context of existing environment.

of off-premises cloud service providers, but there is limited application of these concepts in to on-premises IT infrastructure (e.g., private and hybrid clouds).

4.3.1 Structural Perspective

Domain Definition

The overall SDI system is comprised of the *SDI Management System*, the *Infrastructure* and the *ManagementTools*. The *Infrastructure* domain is comprised of the various devices that can be managed through APIs. The *ManagementTools* domain consists of the various subsystems

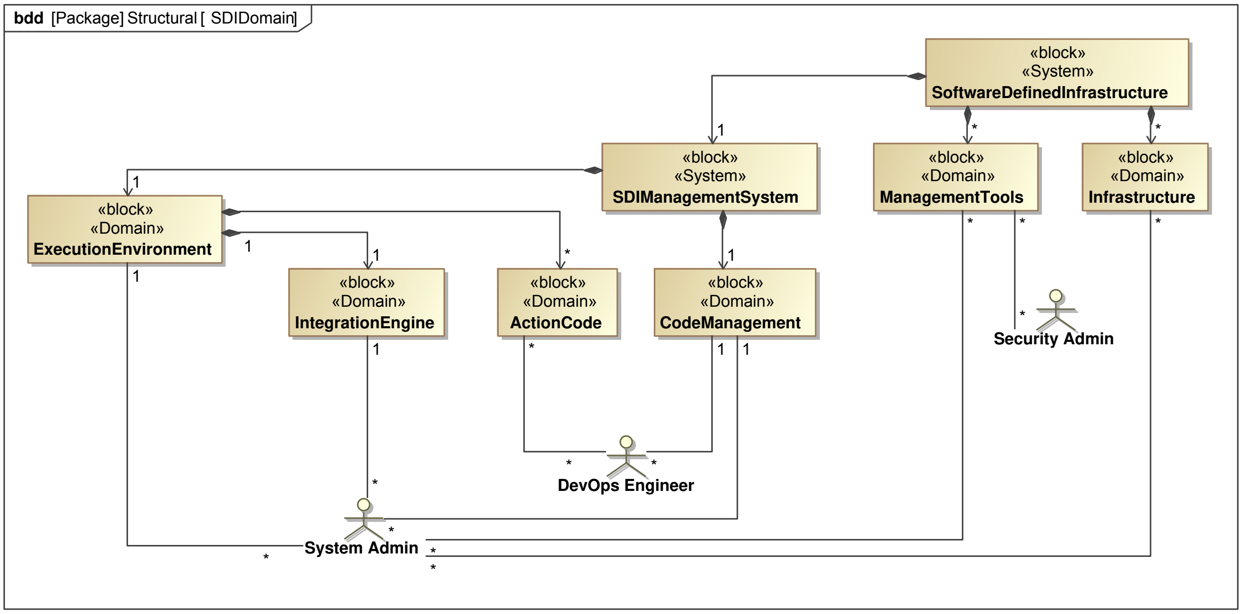

Figure 4.8: High level domain diagram of the SDI.

used to enable and manage the enterprise infrastructure, such as address management, monitoring, and directory systems. The systems in this domain are also managed through APIs. The SDI Management System can be further broken into four separate domains:

- *ActionCode* which represents the code created by DevOps engineers to perform specific tasks within the overall environment and acts on the *SDSystems*.
- The *ExecutionEnvironment* domain where the *ActionCode* runs.
- An *IntegrationEngine* which will integrate with *SDSystems*.
- A *CodeManagement* domain where infrastructure management code will be deployed and managed.

The domain diagram in Figure 4.8 provides the most generic view of the SDI system without specifying particular infrastructure components or network management systems.

Specifications for four key domains are shown below. Note that all of these attributes can and should be embedded in individual domain blocks in the system model.

- *IntegrationEngine*
  - Owner: SDI Management System Developers
  - Description: Provides the ability to define supported external systems' APIs as Interfaces, such as network management tools and virtual and physical infrastructure components
  - Operations: Load new and update existing external systems Interface definitions and / or plugin modules
  - Data: Interface description, interface configuration, interface version
  - Interfaces: *IdentityAccess* and *ExecutionEnvironment*
  - Allocated Requirements: 5, 6, 7
- *Infrastructure* and *ManagementTools*
  - Owner: Customer system administrators
  - Description: represent the external infrastructure and management systems which are managed by the system (such as the ticketing system, or a hypervisor environment)
  - Operations: varies based on the APIs exposed by the vendors of those systems
  - Data: varies based on the function of the interfaced systems
  - Interfaces: defined in the APIs defined by the vendors
  - Allocated Requirements: 5, 6
- *ExecutionEnvironment*
  - Owner: DevOps engineers
  - Description: this is the environment in which the *ActionCode* runs, and includes the Orchestration Engine (that coordinates and schedules automated activities) The Automation Engine (which executes specific *ActionCode*), logging functions and often the *IntegrationEngine* and associated plugins

- Operations: retrieval, scheduling and execution of the *ActionCode*; management and execution of integration plugins; and logging of all activity
- Data: code repository, code version, code status
- Interfaces: varies depending on the plugins configured by the DevOps engineer
- Allocated Requirements: 1, 2, 3, 5, 6, 7, 8, 9

• *ActionCode*

- Owner: DevOps engineers
- Description: this is the custom code written to accomplish tasks within the overall environment, such as the proper provisioning of compute and storage capacity in support of a new application
- Operations: varies depending on the intent of the *ActionCode* and support of APIs exposed by *InterfacedSystems*
- Data: code repository, code version, code status
- Interfaces: varies depending on the intent of the *ActionCode*
- Allocated Requirements: 1, 2, 3, 4, 8, 9

The requirements diagram in Figure 4.9 shows the allocation of high-level requirements to the SDI domains.

4.3.2 Behavioral Perspective

Per the MBSAP the Behavioral Perspective leverages Use Case, Activity, and Sequence diagrams (and others if needed, such as) to define how the system of interest interacts with users, internal subsystems, and the environment. At the OV level of abstraction, this would be between classes of users and the domains defined in the Structural Perspective.

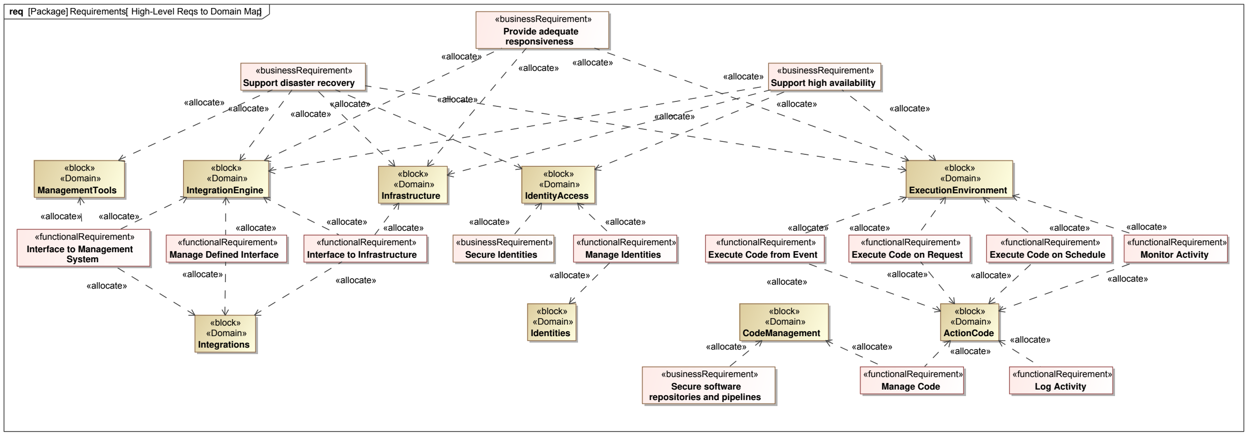

Figure 4.9: Allocation of high-level requirements to SDI domains.

Use Cases

The use case diagram in Figure 4.10 shows the highest level of functionality required by the software defined infrastructure: the solution can respond to a user request (with or without manual approval) and respond to an event that occurs in the infrastructure as reported by a management tool, which could include a "human in the loop" to supervise the response.

Activity Diagrams

The activity diagram in Figure 4.11 outlines the activities that could occur during the response to an event (such as a system failure), in this case including the interaction within the SDI system between the specific ActionCode created to handle the response and the IntegrationEngine that brokers the invocation of the APIs to the infrastructure and network management systems that may be involved in the response. These elements are discussed in more detail below.

Sequence Diagrams

As an example of the interactions between domain elements, the sequence diagram in Figure 4.12 shows the flow of interactions to enable generic automation of a request or event, iterating through changes to both the infrastructure and various tools, with some actions requiring the involvement of a systems administrator. This corresponds to the Use Case diagram shown in Figure 4.10.

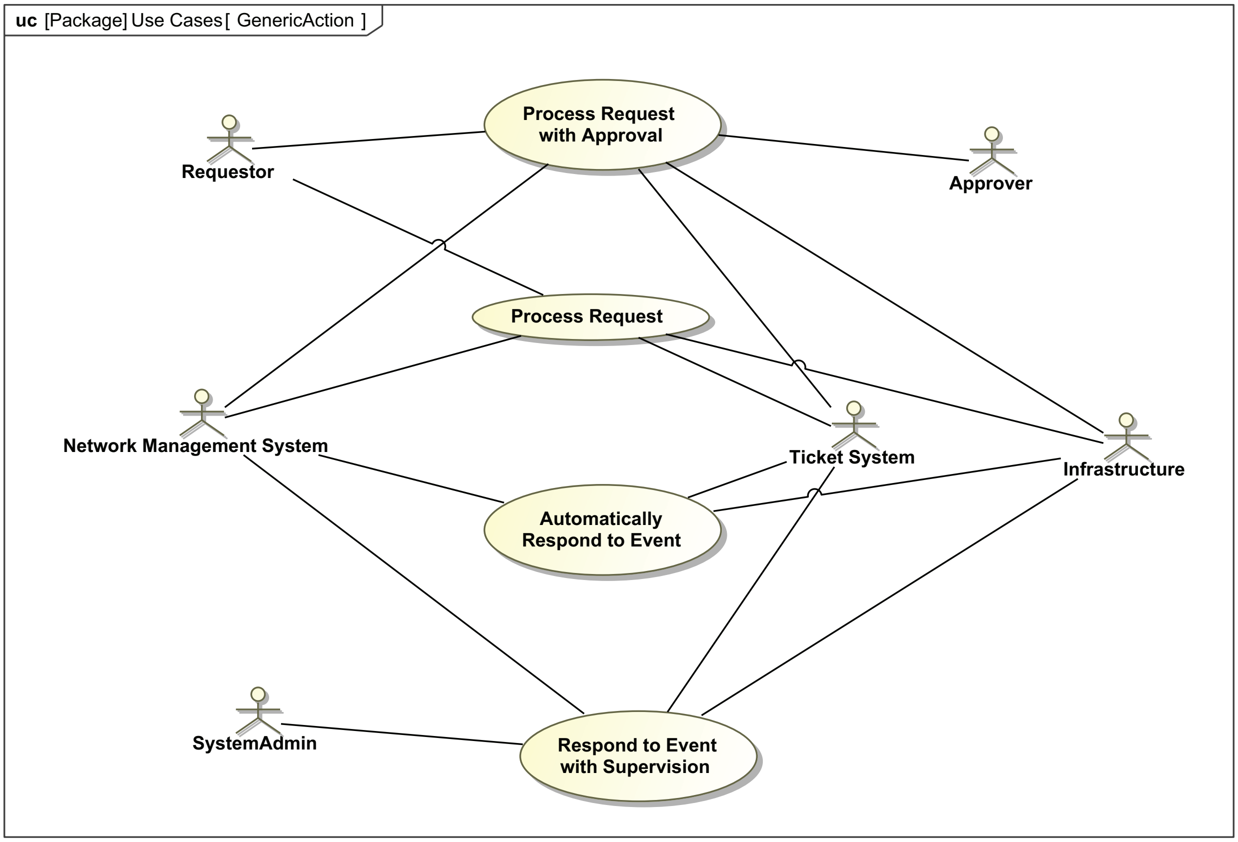

Figure 4.10: Use case depicting the response of the SDI to a request or event.

4.3.3 Data Perspective

The Conceptual Data Model (CDM) captures the overall information categories of a system. The CDM for the SDI is shown in Figure 4.13. Of course, the specific APIs leveraged via the code will dictate the required and optional data necessary as parameters to accomplish specific actions (e.g., the creation of a virtual machine in the hypervisor). These data must either be captured in the request process, raised in the management tools during an event, or supplied by a systems administrator during execution of the automated response. Note that the CDM shown is specific to the server provisioning process and would need to be expanded to perform other functions.

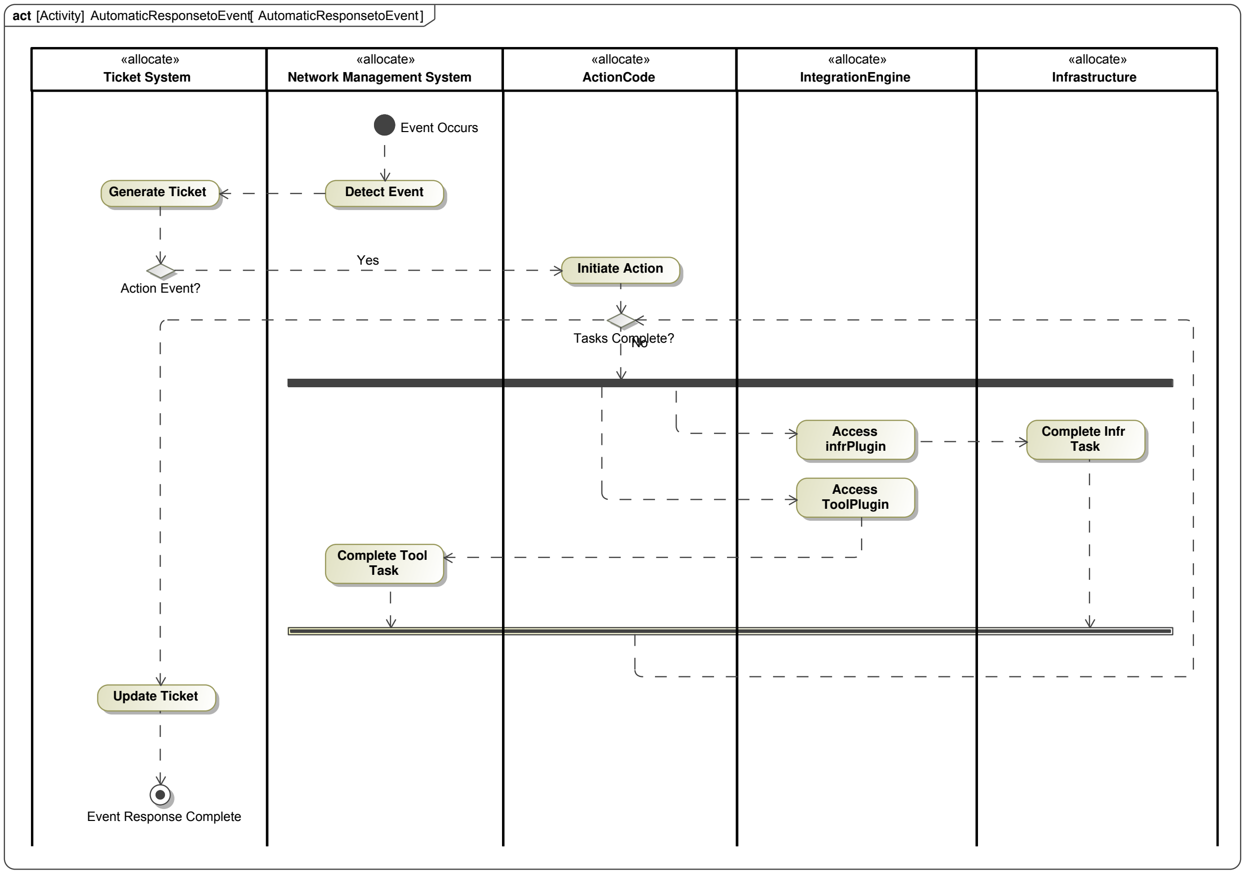

Figure 4.11: Activity diagram outlining the SDI response to a generic event.

4.3.4 Services Perspective

Many services are supplied by the systems within the three domains and exposed via APIs to the *AutomationCode*. These could include, but are not limited to, the following services leveraged by the provisioning function:

- *SDIManagementSystem* domain services.
  - *ConfigureFunction*: provides the ability to register a new automated function in the SDI system.
  - *MonitorRequests*: provides the ability to recognize that a new request for a configured automated function has been received in the ticketing system.

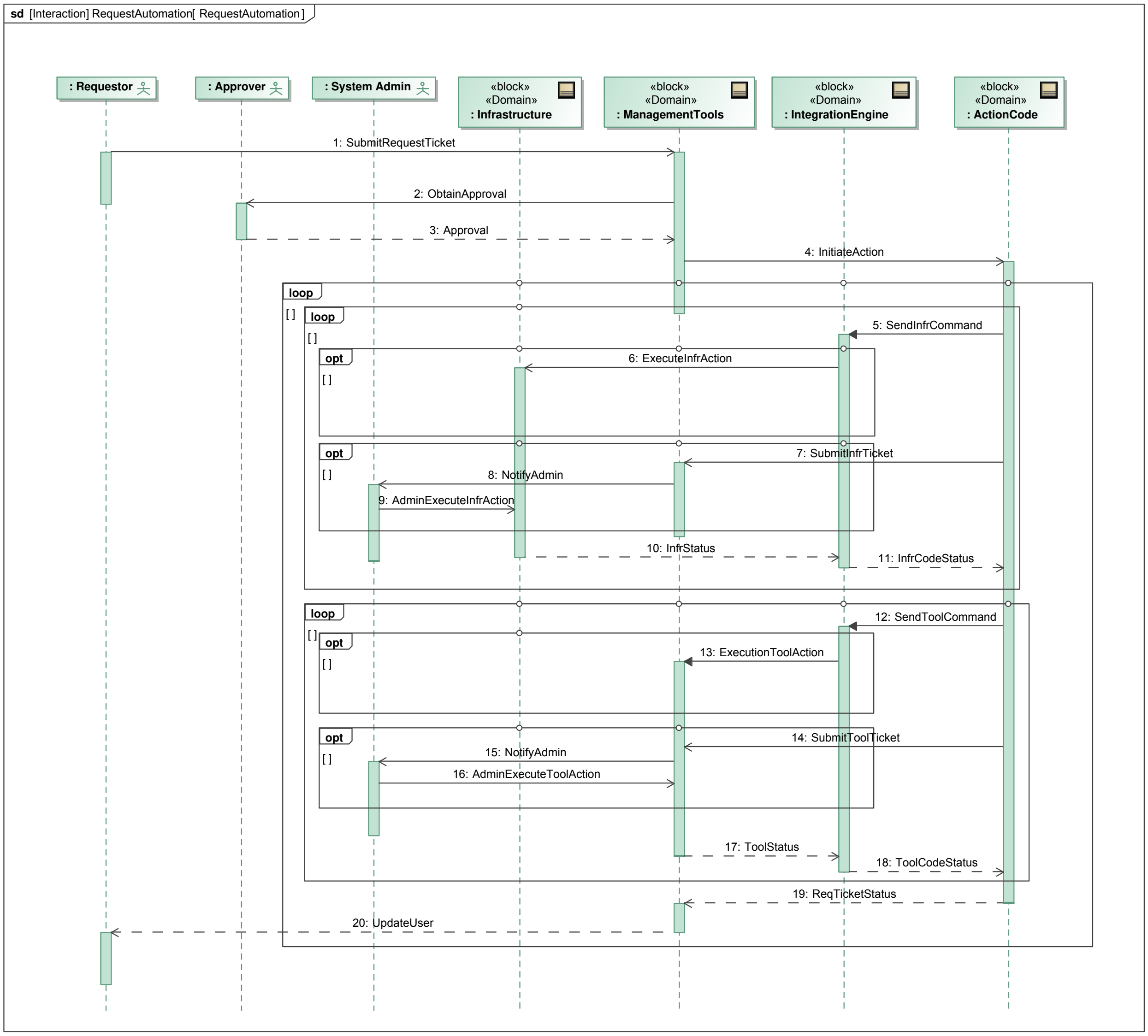

Figure 4.12: Sequence of activities between system modules in the provisioning process.

– *MonitorEvents*: provides the ability to recognize that a new event has occurred in a monitoring tool that triggers a configured automated function.

– *PauseFunction*: provides the ability to temporarily disable an automated function, either based on a schedule or indefinitely.

– *ResumeFunction*: provides the ability to resume an automated function that had been previously paused.

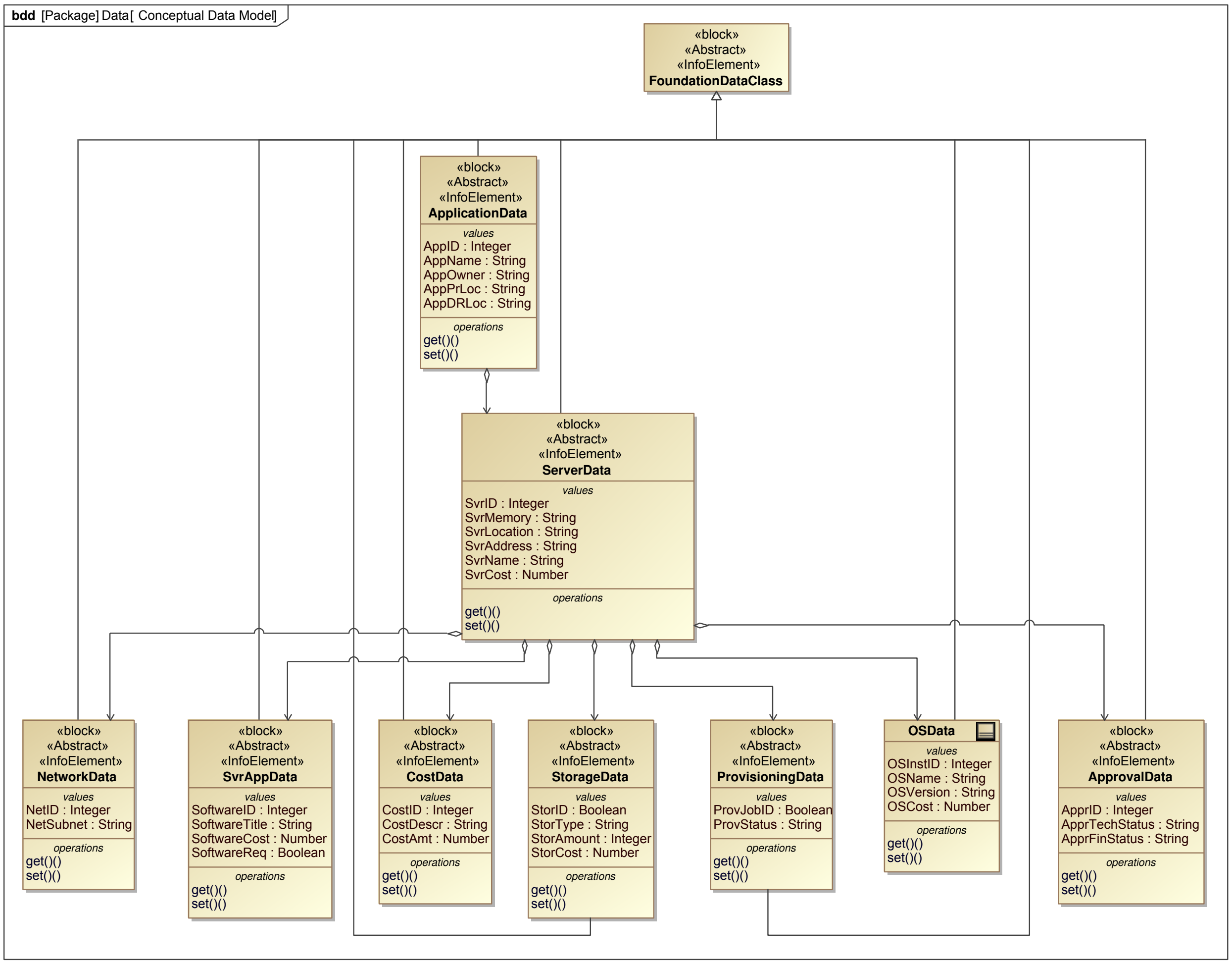

Figure 4.13: Conceptual Data Model for the SDI.

- *RemoveFunction*: provides the ability to de-register an existing automated function in the SDI system.

- *Infrastructure* domain services.

  - *ConfigurePort*: provides the ability to configure a network port on a network device.

  - *CreateVirtualMachine*: provides the ability to create a new Virtual Machine within a hypervisor environment, with parameters specified for memory, processor, and perhaps operating system. Additional services for stopping, starting, and deleting virtual machines should also be available.

- *ConfigureStorage*: provides the ability to provision (or de-provision) storage of a specific type and speed, and associate it with an existing VM.
- *AttachImage*: provides the ability to boot a VM from a specific OS image, in the event the VM is not ”cloned” from an existing VM or template. Additional services should be available to detach an image from a VM, as well as to create and delete images from the repository itself.
- *InstallSoftware*: provides the ability to install software from a specific repository within an existing operating system.

• *ManagementTools* domain services

- *ProvideStaticIP*: provides the ability to obtain an IP address from a range through an IPAM tool. A service to reclaim or release a static IP address should also be available.
- *ProvideCredentials*: provides the ability to obtain administrative credentials required to execute code on the infrastructure from an *IdentityStore* through the *IdentityAccess* system.
- *RegisterName*: provides the ability to register a new IP address within a DNS system. A service to de-register a name should also be available.
- *RegisterDirectory*: provides the ability to register a new server (or other types of object) within a directory, such as Microsoft Active Directory or a CMDB. A service to de-register a server should also be available.
- *SetupBackup*: provides the ability to register a new server or storage location for scheduled data or configuration backups. Additional services should also be available to update and remove backup configurations, as well as to manage individual backup instances.
- *SetAlert*: provides the ability to establish attributes and thresholds for monitoring and alerting. Additional services for updating and deleting alerts should also be available.

- *GetOS*: provides the ability to download an OS from a central repository for installation on a newly provisioned VM. Additional services for creating a new OS image or updating an existing image should also be available.
- *GetSoftware*: provides the ability to download a software package from a central repository to install on a newly provisioned VM (post-OS install). a Additional services to update or uninstall specific software packages should also be available.

This is intended to provide a sampling of possible services within the SDI. Many others would be necessary for the full range of automated tasks that can be performed. In addition, it is expected that the subsystems in the various domains are competitive and commercially available systems, with a large variety of services exposed by .

4.3.5 Contextual Perspective

The following artifacts should be considered by any adopting organization that will influence the overall SDI system:

- Financial policies affecting issues such as approval authority to incur charges, as well as charge-back of the provisioned capacity and software (including both the OS and additional deployed software) to requesting departments.
- Technical architecture standards that would limit the choices available for infrastructure and tool systems and subsystems, as well as their configuration.
- Process documentation and standards that would potentially constrain automation to employ ”human-in-the-loop” steps, or alternatively integrate the new automated functions into larger automation frameworks such as CI/CD tool-chains.

4.4 Logical / Functional Viewpoint

For the purpose of illustration, the remainder of this chapter will follow the canonical example in DevOps of the automation of a server provisioning process, which is the allocation and configuration of hardware (compute and storage) and software (operating system, application packages, patches, etc.) into a functioning system. While this is a trivial task (or set of tasks) in a public cloud environment, it is not trivial in an environment heavily dependent on on-premises infrastructure and skill-based teams — in fact, at the beginning of this research the case study organization routinely averaged 6 weeks to provision a simple system from request to full functionality, largely due to queuing of tasks between teams, as well as the need to manually execute each task. In addition, the results of the exiting process did not always align with the standards, leading to long-term variability in quality.

Capacity must be provisioned and provided to the requester securely and must be integrated into the requisite tools to enable long-term management of the application after deployment. This capacity must also be deployed to an appropriate data center (or multiple data centers), either on-premises, in the cloud, or both. The SDI must also support the creation of multiple application instances in multiple locations to perform application roles such as development, testing, or disaster recovery. The system must accommodate appropriate approval steps to ensure that committed capacity and associated costs align with financial budgets and technical standards.

4.4.1 Structural Perspective

The LV begins the work of decomposing the domains defined in the OV into subsystems and components.

Decompose the ExecutionEnvironment, CodeManagement, Infrastructure, and / or ManagementTools.

The BDD in Figure 4.14 decomposes the *IntegrationEngine* domain to the next level of detail, leveraging a "plugin" design pattern to enable various infrastructure and tool vendors to provide their own modules that handle the specific activities enabled by their products' APIs, while allowing the SDI *ActionCode* developer to concern themselves with the functions of the *IntegrationEngine* API alone.

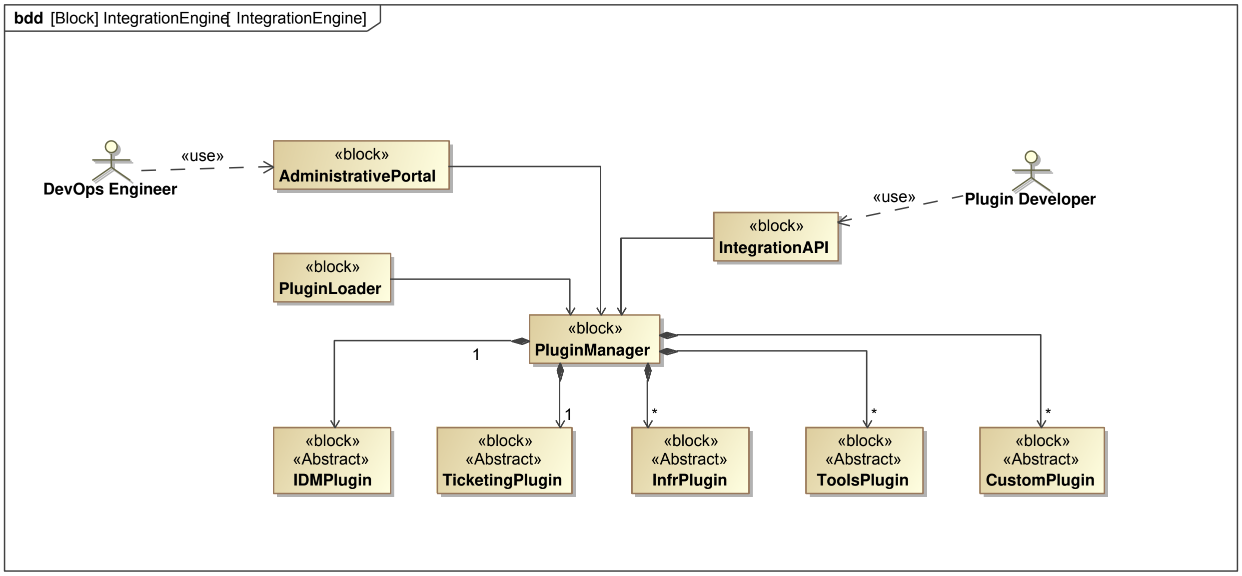

Figure 4.14: Block Definition Diagram of the integration engine.

The Internal Block Diagram (IBD) shown in Figure 4.15 illustrates a subset of interfaces between the SDI domains, as well as a representative sample of plugins within the interface engine that handle connections to a ticketing system, the identity management system, and the infrastructure. Note that even simple automated functions can require many different plugins, depending on the APIs exposed by the systems and vendor products involved in the activity. In the provisioning example, additional plugins could be necessary for the software and OS repository, the monitoring and alerting system, the backup system, the naming system, and the logging systems - not to mention potentially different network, virtualization, server and storage products.

4.4.2 Behavioral Perspective

There are two sets of high-level use cases related to provisioning infrastructure: *Requesting Capacity* which is focused on interaction with stakeholders to define what infrastructure is needed for the deployment of a new application, and *Provisioning Capacity* which interacts primarily with technology component and external system APIs to deliver the infrastructure required to deploy the new application. The Request Capacity use cases are shown in Figure 4.16. The Provision Capacity use cases are shown below 4.17.

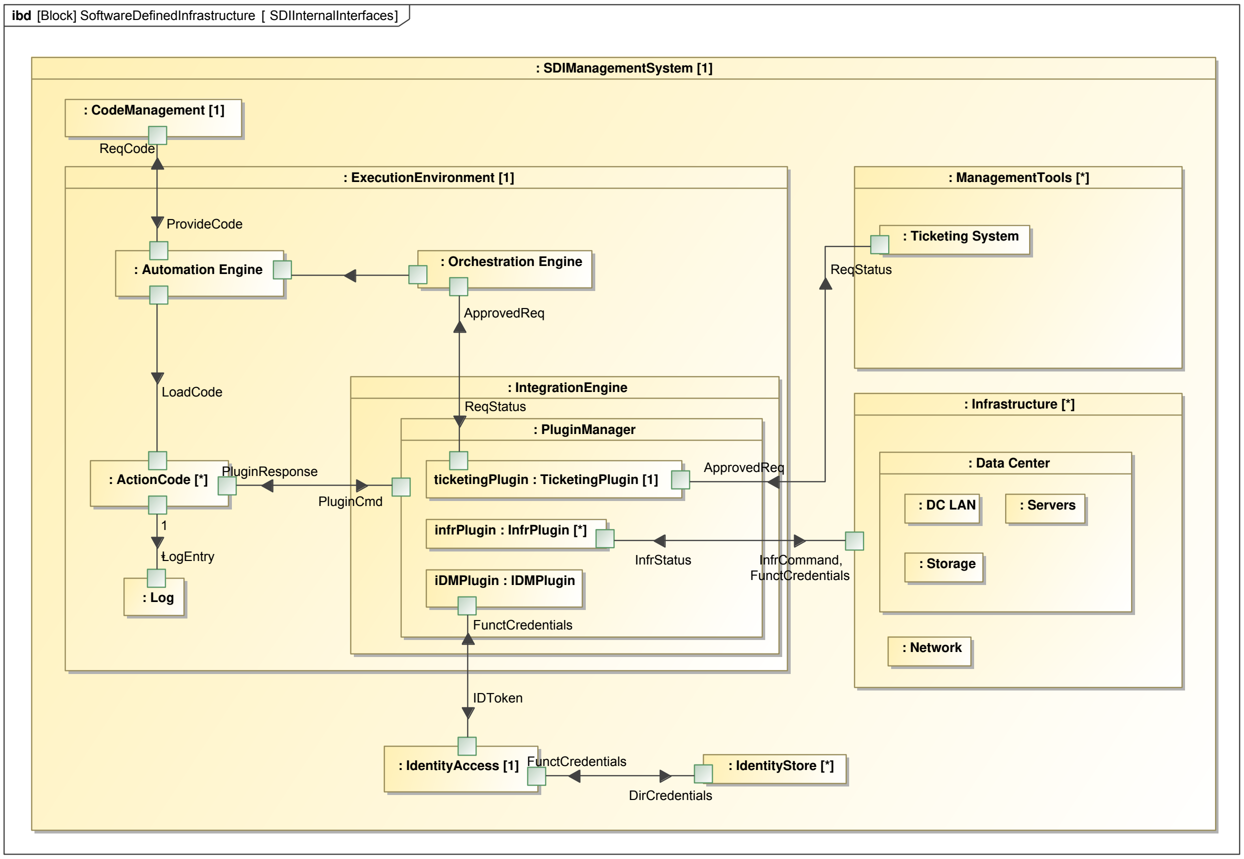

Figure 4.15: IBD showing interfaces between the SDI domains.

Use Case Specifications

The following are specifications for selected use cases:

- Specify Compute Configuration
  - Owner – customer's server system admins
  - General Description – provides the ability for a requester to define the computing resources needed for each server requested, in terms of processor, memory, and local disk configuration or select from pre-configured options
  - Preconditions – application must already be approved by management and defined in the CMDB
  - Trigger – a new application has been approved and the project to implement it has begun

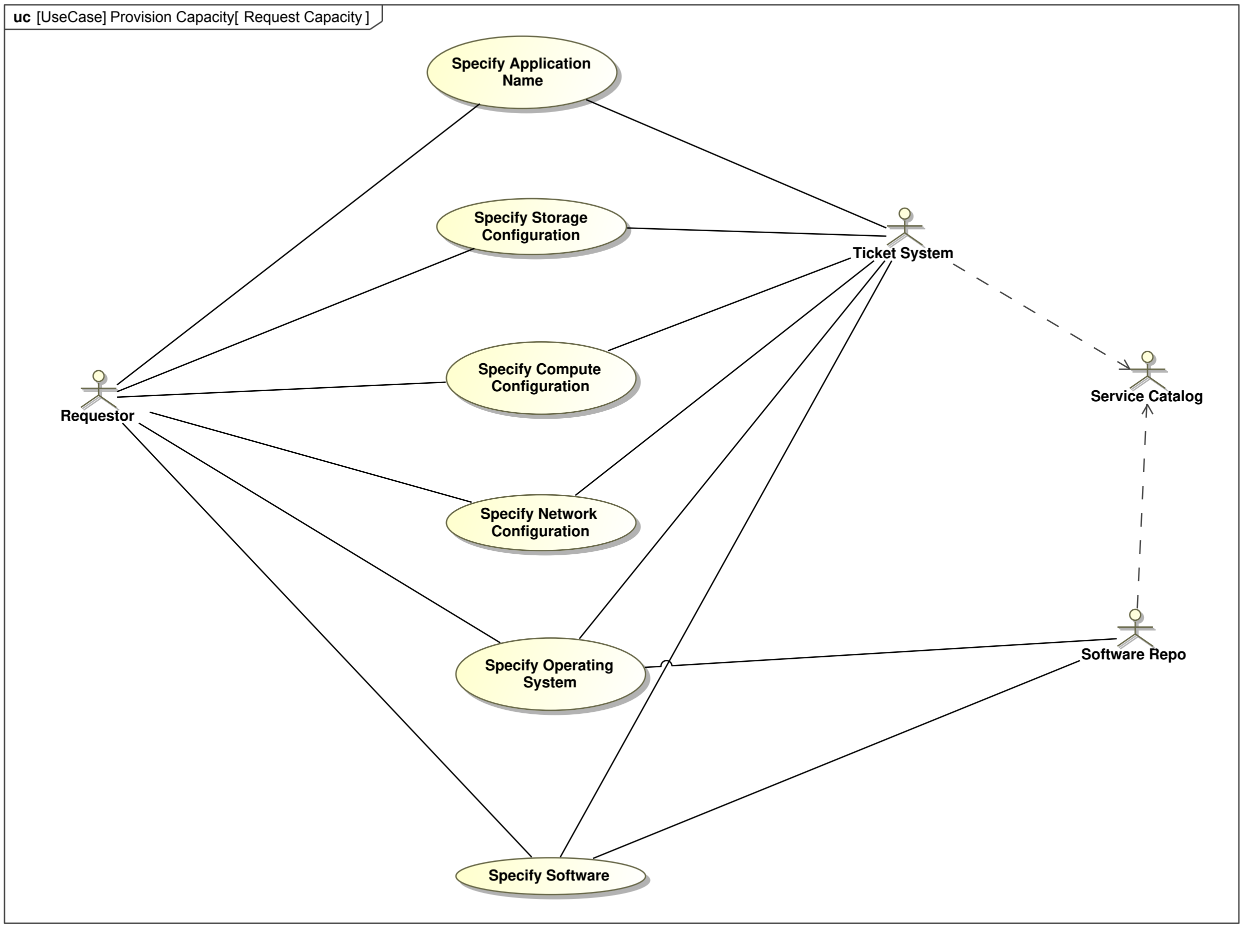

Figure 4.16: Use case for requesting server and storage capacity.

- Postconditions – all server configurations for the application are defined
- User Roles – requester
- Data Objects – memory, processor, disk
- Primary Scenario
  - For each server needed:
    - Requester selects from pre-configured server options (e.g., “basic” or “high-performance”)
    - Requester selects from pre-configured server options
    - System determines cost of each configuration
- Secondary Scenario(s)

        * For each server needed:
            · Requester defines custom values for memory, processor, and local disk
            · Steps 2-3 remain the same
    – Allocated Requirements: 6, 7, 8

• Install OS
    – Owner – Customer System Admin
    – General Description – installs the selected operating system and patches on each configured server instance
    – Preconditions – server capacity (compute, storage, and network) are configured, connectivity established to OS repository
    – Trigger – on approval and deployment of server capacity
    – Postconditions – current OS patches deployed to each OS instance
    – User Roles – automated administrator identity
    – Data Objects – OSData, ServerData, ApplicationData
    – Primary Scenario – server request associated with a new application is approved, with one or more servers requested
    – Secondary Scenario(s) – additional server(s) requested for an existing application
    – Allocated Requirements – 1.11 Install Operating System

• Log Activity
    – Owner – Customer System Admin
    – General Description – records all actions taken, along with success or failure, to the system logs
    – Preconditions – provisioning actions performed or exceptions handled

- Trigger – any automated activity performed
- Postconditions – N/A
- User Roles – automated administrator identity
- Data Objects –
- Primary Scenario – record made of each action completed
- Secondary Scenario(s) – record made of each action failed / rolled back
- Allocated Requirements – 1.12 Log Activities

State Machine Diagrams

The diagram in Figure 4.18 depicts the various states the request to provision the can take, from the initial drafting by the requester to the complete provisioning or cancelation of the request.

To be completed.

4.4.3 Data Perspective

Logical Data Model

To be completed.

4.4.4 Services Perspective

To be completed.

4.4.5 Contextual Perspective

The architectural layers for the SDI are shown in Figure 4.19. **add more context!**

4.5 Organization Specific Elaboration

The examples at this level of abstraction are vendor- and product-agnostic (i.e. functional), but can, of course, be continually refined to the physical design level using specific product characteristics, versions, capabilities, and APIs as the infrastructure is further elaborated. As an example, the modeler could explicitly define HashiCorp's Terraform product as the "execution environment" and specific REST code stored under version control in the organization's GitLabs environment as the "action code". Additionally, the modeler could define the virtual and physical environments to which each component is deployed. Each of these subordinate systems can be modeled separately in the MBSE model/tool by the responsible IT systems engineering team to the level of detail deemed useful and necessary. This is demonstrated in Figure 4.20, which highlights the ability to continue to develop systems from conceptual models into detailed designs suitable for engineers to build.

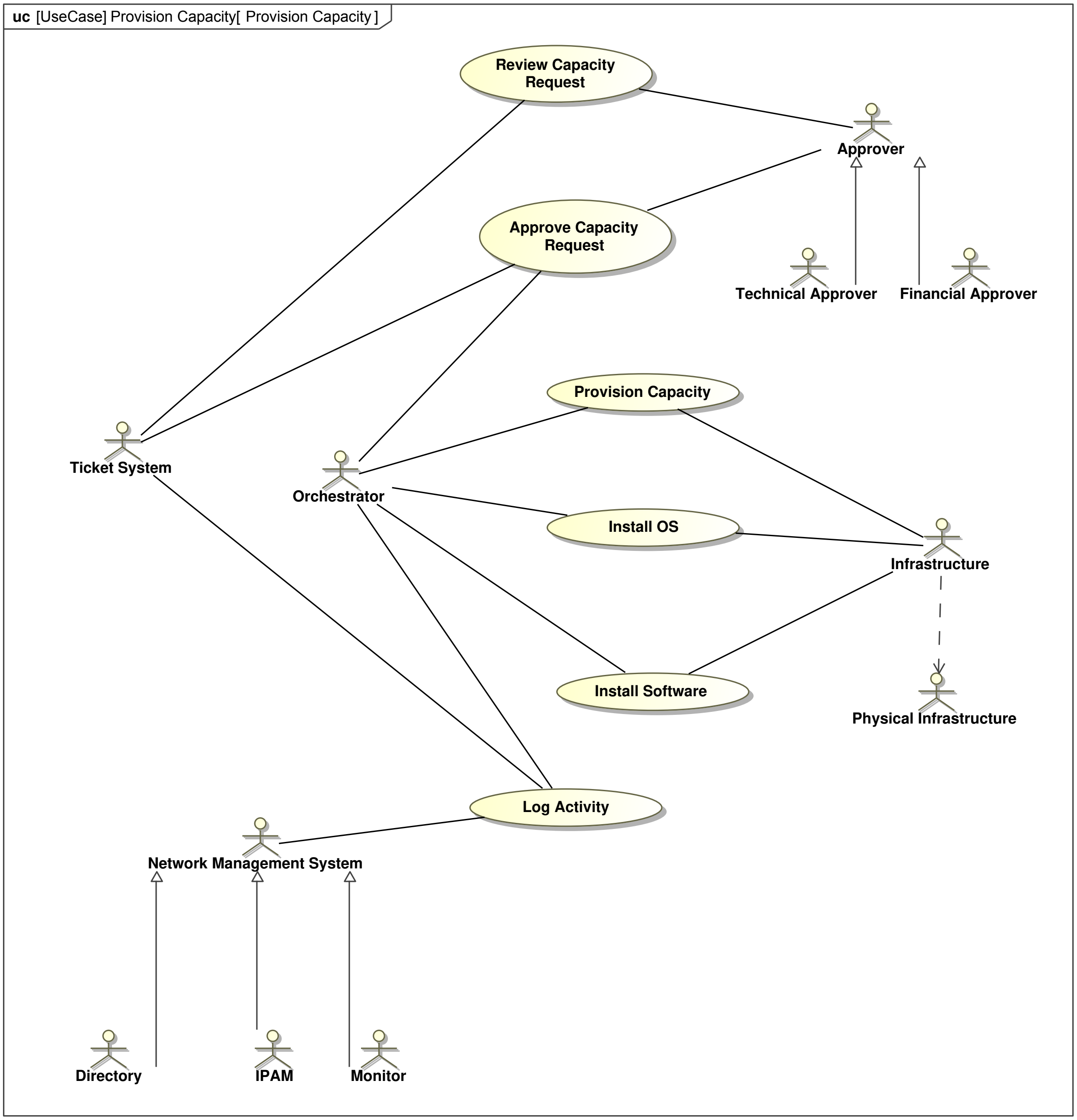

Figure 4.17: Use case for provisioning server and storage capacity.

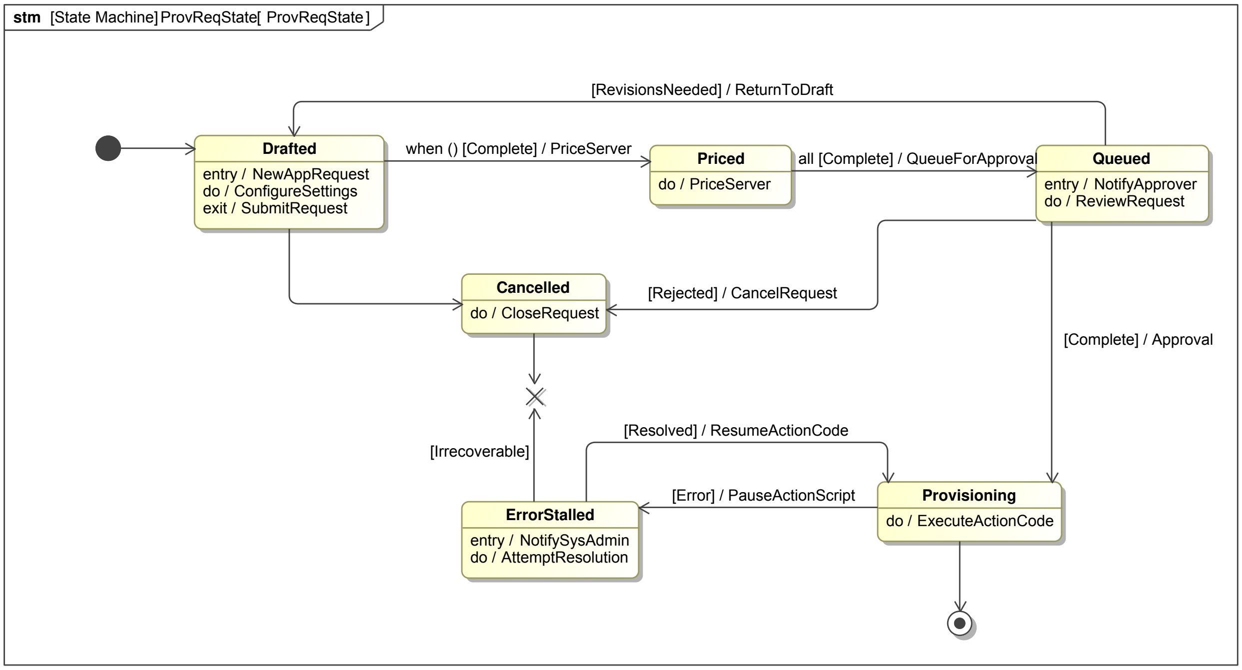

Figure 4.18: Possible states of a provisioning request.

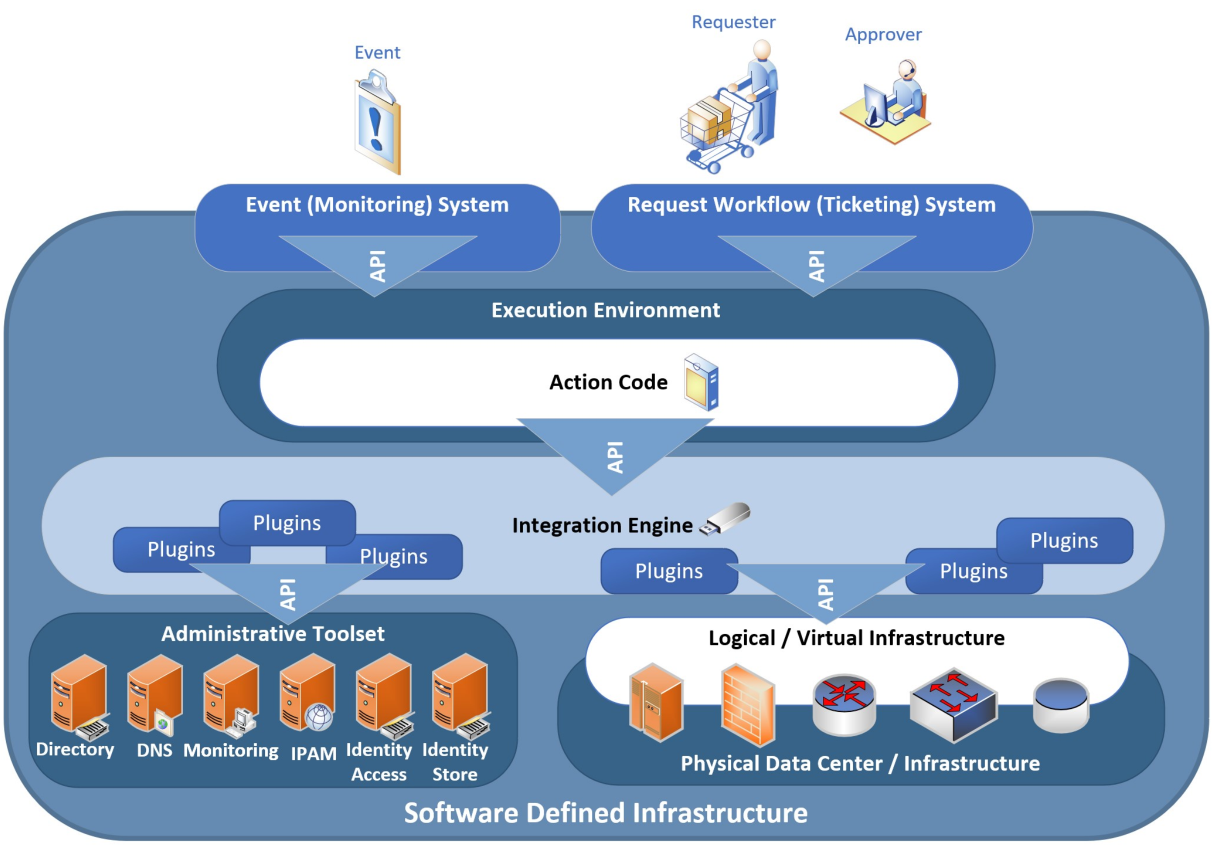

Figure 4.19: SDI architectural layers.

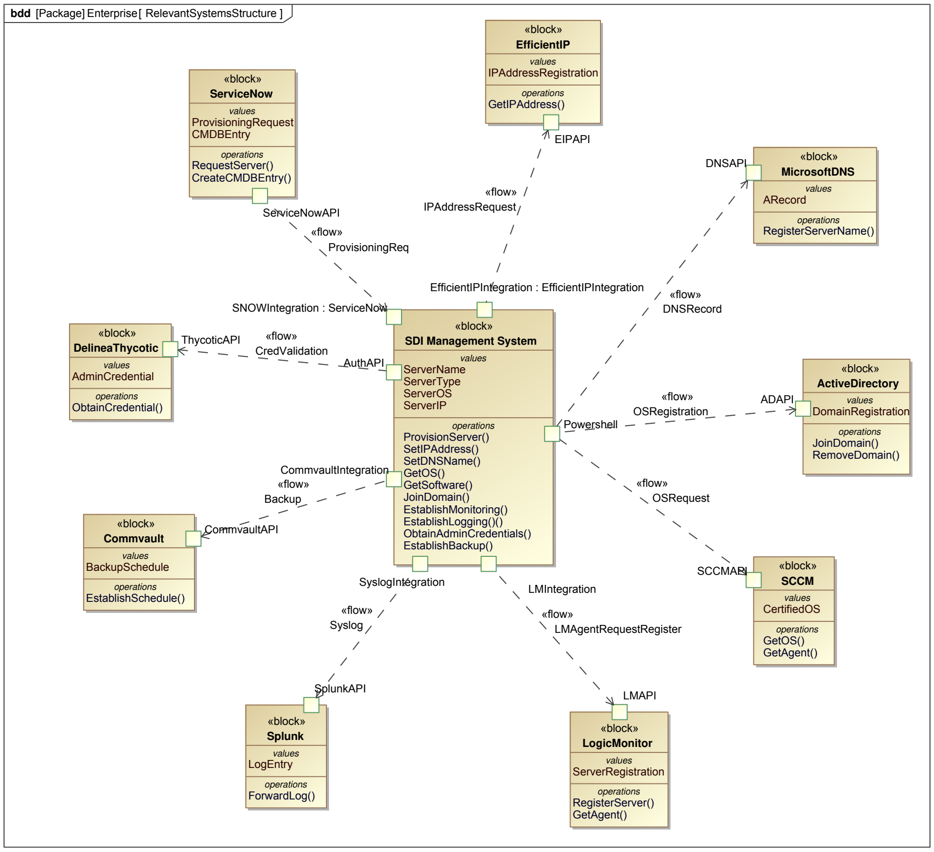

Figure 4.20: Product-specific design example for the SDI Management System.

Chapter 5

Presentation of Results

5.1 Introduction

In the course of trying to answer the original research questions, it becomes apparent that there are really two underpinning issues that need to be addressed to determine the value of SDI:

- What activities should be prioritized for automation from a benefit point of view?
- How can those activities be itemized and what would be the cost?

Section 3 addresses how IT leaders should begin to identify and prioritize activities that should be automated to the extent possible. At the top level, this includes work items that happen *frequently* enough to justify the investment in automation and that are *repeatable* enough that the sequence for execution is predictable and can be parameterized to address the specifics of execution in each instance. Rare or unique tasks should not generally be automated, unless it involves a scale of effort where the effort to automate can be justified. For example, even a unique task that has to be implemented over hundreds of thousands of devices can be worth automating to ensure consistency and accuracy of implementation and enable completion much more quickly. Building on the characteristics above, work items that are time sensitive (e.g., response to a critical event) should be considered for automation, with the goal of significantly reducing or eliminating queue time and improving responsiveness and work item pickup rates.

Section 4 develops an architectural framework for an SDI that can enable the activities identified above, using MBSAP to explore the solution space and SysML to document it. The MBSE model treats the existing IT infrastructure and subsystems (esp. management tools) as black boxes defined in terms of their capabilities and interfaces, and focuses on the successive elaboration of the SDI Management System that ties them together through APIs and provides the environment in which automation code can execute.

5.2 Example Business Case for SDI

The focus of this analysis will be on the creation of the initial SDI framework. The example analysis below will stay with the canonical example of server provisioning used throughout this research.

5.2.1 Objectives and Scope

For each use case identified as the initial drivers, leaders should quantify the expected outcomes to the extent possible. To start, obtain historical estimates for how often these tasks occur and, to the extent possible, how much effort they take by each technical team, as well as the total duration to complete.

In the case study organization, new servers are provisioned in the central data centers at a generally consistent rate of between 20 and 25 per month; however, roughly 3 of those were requests for multiple (2-6) servers at one time, usually part of a new system deployment, so a total of 35 servers per month is assumed in the following analysis. Each individual server requested required the participation of six separate technical teams, several of which touched the request more than once as the request was routed to completion. From Section 3.5.3, each task can take 3 to 4.5 hours on average to complete, which implies 18-27 hours of effort; this seems high for the tasks involved, so a lower number of 10 hours will be used for this analysis. Before automating the process, the case study organization observed total durations of up to 6 weeks before a requested server was completely provisioned and ready for use.

The objective was to completely automate 95% of the server provisioning process in the organization's primary data centers, requiring no manual intervention by any technical team, with completion in 8 hours (assuming adequate server and storage capacity already deployed). This represents a reduction in hours spent by 350 hours per month (4,200 hours per year) and a reduction in the elapsed time to complete by 29 working days (from 30 working days to one).

The scope of the initial effort included closing any gaps in the underlying infrastructure and tools (with exposed APIs), establishing the SDI Management System, developing the automation code to support provisioning, establishing or modifying appropriate management processes, and making any changes required to requisite staffing and appropriate skills.

5.2.2 Proposed Solution

The case study organization chose to take advantage of the capabilities of VMware vRealize Automation (now known as VMware Aria Automation) as the basis for the SDI Management System, as it was already a licensed (but unused) component of the virtual infrastructure. Otherwise, the team used the APIs of their existing infrastructure and tools, where no major gaps were identified. As the organization does not develop software products, there was a lack of coding skills in the IT team, so the decision was made to leverage a third party partner with the requisite technical skills in the VMware tool. The solution required the adoption of a code repository and versioning system (GitLabs was chosen), as well as the creation of processes for the testing and review of automation code, as well as the maintenance of the code over time (for example, as the components of the underlying infrastructure and tools are upgraded or replaced). The first phase of the solution would provide the ability to provision a single server at a time in a specific segment of the data center. Further phases would expand this to enable bulk server provisioning (multiples at a time), include registering the servers in monitoring and backup tools, and expand the different data center environments where servers could be provisioned.

5.2.3 Cost Analysis

Initial Investment

In the case study organization, no new hardware was required, which reduced initial costs. In an organization where these would be required, sufficient capital for initial acquisition as well as operating funds for annual maintenance or subscription costs would be required. Funding for the third party DevOps engineer was provided by the vendor in this case; otherwise, this would have been a direct cost to the case study organization (potentially capitalizable depending on the financial policies in place). No other software was required, as all other tools in use supported programmatic use via API. Process change efforts were led by internal staff and leaders, with no incremental cost to the organization, estimated at approximately 120 hours.

Ongoing Operational Costs

The adoption of GitLabs for the code repository and version control entailed a nominal new annual subscription (operating) cost. As no hardware or additional software was required, no new maintenance costs were required to support the project. The organization estimated an annual cost to maintain the automated provisioning code to be roughly 40 hours a year, after transitioning from the third party DevOps engineer.

5.2.4 Benefits Analysis

Tangible Benefits

In general, automation of server provisioning was estimated to ”save” the equivalent of two equivalent full-time staff; however, these were not directly realizable as those hours were spread across six teams. Instead, these represent hours available to apply to other tasks within those teams. Server provisioning times were reduced to <2 hours even for multiple systems requested in bulk, substantially improving response times to requesters and resulting in faster deployment cycles, even at scale.

Intangible Benefits

Anecdotally, the teams expected to observe improved reliability due to consistent configurations and reduced manual errors, and as a result improved compliance with security policies.

ROI and Financial Metrics

As initial investment by the case study organization was negligible due to the support for the installed infrastructure and tools for automation, the payback period was near immediate. The use of financial tools such as net present value (NPV) was unnecessary to justify the effort.

5.2.5 Assumptions and Constraints

The provisioning of servers to remote data centers (in the hospitals) was excluded from scope, due to a lack of underlying infrastructure with exposed APIs, in turn a result of insufficient historical capital investment.

Sufficient training and adoption by IT staff. Stable or predictable infrastructure demands during the automation rollout. Management support and resources (budget, personnel) remain consistent. Vendor or consulting availability meets project timelines.

Organizational policies or regulations (e.g., security, data privacy) that limit tool choice. Budget or resource limits that may cap the extent or pace of automation. Legacy systems that may not fully support automation tools, requiring additional effort.

5.2.6 Risks and Mitigation

Technical Risks. Integration Issues: Legacy systems may be difficult to automate. Tool Reliability: Dependence on automation tools introduces new single points of failure. Security Vulnerabilities: Automated processes may inadvertently expose systems if not properly secured. Mitigation: Perform proofs of concept, use staged rollouts, maintain robust security controls.

organizational risks. Resistance to Change: Staff may be reluctant to shift from manual methods. Skill Gaps: Lack of in-house expertise with new automation tools. Mitigation: Comprehensive training, change management strategies, pilot programs, strong communication.

Financial risks. Budget Overruns: Unexpected costs for licenses, professional services, or additional infrastructure. Benefit Shortfall: Actual cost savings or productivity gains may be less than projected. Mitigation: Contingency funding, phased investments, regular project audits.

Operational risks. Downtime During Implementation: Possible service disruptions when integrating automation solutions. Mitigation: Schedule maintenance windows, implement robust testing procedures, have fallback processes.

5.3 Roadmap for SDI Implementation

- Identify specific use cases and quantify the anticipated benefit. Note that these benefits may include hard savings (elimination or avoidance of cost) or quality attributes such as improved quality or timeliness as outlined ini the example above. These use cases should be driven by each organization's unique data for task and ticket completion; if these data are not yet available, the case cannot be empirically made.

- Identify the underlying infrastructure and management tool subsystems needed to automate these use cases and determine their support for automation. Not every organization has implemented solutions with appropriate APIs.

  - Any subsystems that do not currently expose the necessary APIs may need to be upgraded or replaced, which may require additional funding.

  - Any subsystems that do not currently exist (e.g., credential management tools) may need to be priced and procured.

  - In particular, the deployment of appropriate commercial tools for credentials management and IPAM are not yet ubiquitous but should be considered preconditions to broader adoption of on-premises SDI. Those efforts may either limit the initial scope of the automation effort, or become a prerequisite to it.

- Evaluate the capabilities of available SDI Management System products against the solution requirements and obtain pricing for the requisite capabilities. Note that while the development of a custom SDI Management System may be possible, it is inadvisable for organizations without an existing development skill set, as it becomes yet another code base to be managed over time.

- Evaluate the capabilities of current staff to develop the custom automation code, manage it over time, and expand it as new use cases arise. Note that in some cases individual staff may have some experience automating at a small scale (e.g., their own repeatable tasks), but they may not be free to dedicate sufficient time to this function. Other staff will not have the required skills for automation. Incremental staff may need to be on-boarded, either due to a lack of skills or insufficient capacity of existing skills.

- Evaluate existing management processes and tools for developing, testing, deploying, and managing custom code. In healthcare IT providers, these are likely to be minimal if not completely absent. Deficiencies will have to be addressed through training, the on-boarding of leaders with these skills, and possibly the procurement of enabling tools (e.g., code repositories).

Each healthcare provider IT organization must determine whether the total cost of investments required in enabling tools and staff justifies the identified (and perhaps future potential) benefits.

5.4 Findings

5.4.1 Research Question 1: What are the basic components and capabilities of an on-premises SDI system?

This question is explored in Section 4 as a vendor- / product-agnostic architectural framework. It is assumed that the IT infrastructure is composed of subsystems of the network, data center, and management tools that the vendors have exposed to programmatic manipulation as APIs. The SDI Management System itself minimally consists of an environment in which to execute custom automation code, and an integration engine to enable consumption of the infrastructure subsystem vendors' APIs. More broadly, it should also support components to manage the custom automation code as well as manage the credentials required to execute the infrastructure subsystem APIs.

"Productized" examples of the SDI Management System that can be leveraged on-premises at the time of this writing include SaltStack, Terraform, and VMware Aria Automation (formerly vRealize Automation). Due to the degree of change within IT infrastructures as they evolve, a key benefit to obtaining a product to enable SDI management is the likelihood that the vendor will maintain ongoing support to interface new and changed subsystems. However, it must be emphasized that these products are necessary but insufficient in and of themselves. IT leaders must recognize that the inclusion of the underlying infrastructure and management tools (and their associated APIs is critical to understanding the overall solution.

5.4.2 Research Question 2: Should healthcare providers implement on-premises SDI?

This question is partially explored in Section 3 and is supported by existing research described in Section 2. The preliminary answer to the question "should healthcare providers implement on-premises SDI?" is a qualified "yes," insofar as it relates to identifying clear potential benefits. However, the question of whether the general investments required to build itself are worth those benefits given the constraints of healthcare provider IT is subject to other factors. The initial cost of implementation for an SDI solution goes beyond the adoption of the SDI Management System (whether procured or custom-built) and the development of the custom automation code itself - this code must be maintained over time and expanded to include additional use cases. This means that the skills required to perform this function must be maintained within the organization as a key function, i.e., the organization must build a DevOps team and maintain the tools they need to perform that function.

5.4.3 Research Question 3: How should provider organizations proceed to implement on-premises SDI, if at all?

The synthesis of completed work in Sections 3 (that addresses priorities) and 4 (that addresses components and prerequisites) enabled the creation of the proposed roadmap earlier in this section for implementation of on-premises SDI by healthcare providers.

5.4.4 Research Products

The following products have been published or submitted for publication in support of this work:

Using Hybrid System Dynamics and Discrete Event Simulations to Identify High Leverage Targets for Process Improvement in a Skill-based Organizational Structure has been submitted for publication as an academic paper and presentation at IEEE's 18th Annual International Systems Conference (SysCon) scheduled in April, 2024. This paper is focused on the single-team simulation models built to partially answer Research Question 2.

Bringing Systems Engineering to IT Systems Engineers? SysML and MBSE for IT has been submitted to IEEE's IT Professional journal. This paper explores the utility of MBSE in the creation of an IT infrastructure, using the SDI architectural framework as an example. While UML is commonly used in the development of software, the use of SysML represents an opportunity for use in IT infrastructure, where physical components and logistics come into play.

Proposal for a Reference Architecture for On-Premises Software-Defined Infrastructure has been submitted to Sage Journals' Journal of Information Technology. This paper explores the architecture of a software-defined infrastructure management system capable of orchestrating the administrative functions of an enterprise hybrid data center infrastructure, such as deploying software updates (patches) on a scheduled basis or responding to certain types of incidents as driven by events. The overarching goal of this architecture is to guide IT leaders in how to enable automation of IT use cases that address the opportunities outlined in the AS-YET PUBLISHED IT PROFESSIONAL PAPER. This paper focuses on the components of SDI in answer to Research Questions 1 and 3.

5.5 Conclusions

The available results from the current versions of the coupled simulations indicate that there are identifiable benefits to the management of IT work by using automation on-premises SDI, but the question remains open as to whether it is worth the time and financial investment of healthcare provider organizations to build it in the first place. Of course, this question can only be answered by the specific circumstances of each individual organization.

The case study used in this paper is a canonical process in DevOps, and could just as easily represent a public cloud-based environment and have been modeled in UML by DevOps engineers. In fact, the use case and sequence diagrams *are* UML diagrams. So what do SysML and MBSE bring to the table to develop a reference architecture for SDI? In short, they enable the design and documentation of physical infrastructure technologies (and engineers that design and support them) that still underpin a huge portion of the enterprise IT environment and yet do not have any common standards today in addition to software systems and components where traditional UML is more commonly understood. This lack of standardization and dependence on unstructured documentation inhibits clear and precise communication between different domains of IT systems engineers, as well as with software engineers. This, in turn, increases the probability of errors that result in project rework or production incidents, decreases efficiency and effectiveness, and increases overall costs.

5.5.1 Software Engineering Integration

UML is generally well understood and database-supported (Computer-Aided Software Engineering (CASE)) tools are more commonly adopted by software engineers. As a UML profile, many software engineers readily understand SysML v1.X, and there are SysML plugins for some of the commonly adopted tools used by them, including Rational Rhapsody, Sparx Systems Enterprise Architect, Visual Paradigm, and Dassault/Catia/MagicDraw. In addition, there is also explicit support for architectural frameworks and methodologies such as TOGAF and Zachman in several of these products. In parallel to MBSE, various software modeling and development frameworks such as Model-Based Engineering (MBE) and Model-Driven Software Engineering (MDSE) have enjoyed the attention of researchers and practitioners for the last 20 years [107, 108]. Interestingly, UML, CASE, and the various model-based development frameworks provide similar benefits — and suffer from many of the same barriers to adoption — as SysML and MBSE [109].

5.5.2 Barriers to MBSE Adoption in IT

In parallel to software engineering adoption of UML and CASE tools, Chami and Bruel highlight 10 major barriers to adoption of MBSE generally, notably including upfront tool cost, tool adoption and executive support, and the complexity of the models — especially at scale [88]. In addition to the cultural factors that affect adoption, an organization needs to adopt or modify a modeling framework that provides just enough benefit without imposing excessive overhead on the engineering teams and their management to encourage or enforce use. These issues seem to bias success towards larger and more mature organizations. Also, note that a strong success factor is related to training engineers in the language, the modeling process, and the tool itself.

From a practical standpoint, IT vendors do not currently provide pre-built packages representing their products that can be imported into a SysML modeling tool. This would require the creation of subsystem block diagrams by consuming IT systems engineers and would limit the value of modeling these products below the conceptual or ”black box” level without investment from the IT team to build them. Additionally, while SysML can support the modeling of physical locations *technically* it does not automatically depict these in a visually intuitive manner.

As with any organizational change activity, it is necessary to focus on achievable goals to demonstrate initial success and then expand upon those; any attempt to model the existing IT infrastructure in even a moderately complex organization will be an exercise in “boiling the ocean” and risks not providing the level of value to outweigh the costs of implementation. The leaders responsible for any adoption of SysML and MBSE to IT must be able to articulate a long-term program for adoption, structuring the models and methodology with an initial focus on elements that remain stable over the long term while supporting the incremental expansion of the models as the infrastructure evolves in line with the architectural road maps. Models should only represent systems or subsystems to the degree necessary to solve the problem at hand and resist the temptation to “boil the ocean”, allowing successive projects to continue to elaborate on the model. Finally, modelers should avoid representing in detail components with a high velocity of change and limited utility to other domain engineers unless the requisite data can be automatically fed into the model through integration with operational management toolsets.

Chapter 6

Summary, Implications and Future Work

6.1 Summary

Insert summary

6.2 Implications

Insert implications

Limitations

6.3 Future Work

Beyond the immediate objectives of the research for this dissertation, there are opportunities for future research branching from both the process modeling and simulation efforts and the on-premises SDI architectures being addressed.

This research focuses on one particular organizational structure and work management model that appears throughout healthcare provider IT and in other functions and industries: that of skill-based teams with responsibility for both scheduled (project) and unscheduled (operational) work types. At the end of Section 3 I outline six interventions that the models indicate could improve results, only one of which I address in detail, that of automation. However, given the ubiquity of this organizational structure, I believe that the coupled models as designed could be leveraged to justify the use and predict the benefits of the other five options, as well as to explore any limits to their validity without - for example - the necessity to add resources, which is (currently) beyond the scope of the models. Two areas in particular could be fruitfully explored:

- The utility of separating operational and project work responsibilities into different teams in order to prevent high-priority unscheduled work from interrupting important scheduled work, and vice versa.

- The potential for multi-skilling of resources to prevent cross-team work queuing, as well as the creation of cross-functional teams.

Of course, the limits to the effectiveness of these improvements must also take into account the financial costs associated with making these organizational changes and the additional dynamics that financial considerations would introduce into the SD model.

The evolution of the SD and DES models to address the interacting dynamics of work management across two (or more) separate teams to further flesh out the recommendations is a natural extension of this research that has not been addressed in the literature. The complexity of work management is expected to increase exponentially as the number of teams increases and introduce new dynamics related to the coordination of work across the teams.

In addition, the influence of "managerial pressure" (and related upstream and downstream factors) on SD models is supported by the literature in terms of *direction* of impact, but not in terms of *degree* of impact. In other words, the literature explores *how* managerial pressure can qualitatively impact work performance (positively and negatively), but not by *how much* it does quantitatively. Research to determine the degree of effect these factors have would certainly be difficult, but would be highly beneficial to the ability to verify and validate the simulation of models.

With regard to the on-premises SDI architecture, the research can take several directions. Clearly, it is the continued dependence on large-scale on-premises IT infrastructure, and the inability of healthcare providers to consume public cloud services for a significant proportion of their environment, that makes the question of on-premises automation relevant. This is driven by the limited support of many healthcare COTS solutions for being deployed in the cloud, due to technical limitations (e.g. older software code and architectures) or because they are coupled with physical infrastructure (such as patient monitoring systems). Although solutions will increasingly support partial or full public cloud deployment, the on-premises limitation will remain a reality for many. An area for future research is the expansion of the use cases addressed by the proposed framework and road map. As an example, the potential for automation of the COTS solutions themselves (not just the infrastructure) in support of CI/CD of changes to the application configuration could be investigated. Although not every DevOps practice or use case may be viable in on-premises SDI or the systems running in it, a more expansive exploration of which would allow provider organizations to further leverage their investment.

More generally, the utility of leveraging SysML and MBSE in the design and management of IT infrastructure is an area that has attracted little research to date. These tools and methods are generally not formally taught to IT systems engineers nor are they widely applied to complex evolving systems-of-systems such as IT infrastructures (or any infrastructure, including civil). The combined value of both addresses the observed limitations in the management of IT infrastructures over time, especially with regard to establishing a definitive, centralized repository of requirements and design decisions in lieu of various unstructured documents. However, the barriers to adoption of SysML and MBSE - in particular the complexity and cost of the tools and training to make use of them - serve to limit both the speed of adoption and scope of their use.

The expansion of research into this area would span several existing INCOSE working groups (e.g. Architecture, Complex Systems, Critical Infrastructure Protection and Recovery, Information Communications Technology, Infrastructure and System of Systems) and could perhaps justify its own working group if sufficient interest exists.

Appendix A

Appendix A: Simulation Scripts

A.1 Simulink Script

```
%Initialize SD Variables for first run

    BaseMgmtPreempt = 1;
    BaseMgmtPress = 1;
    BaseFatigue = 1;
    BaseQueuedTasks = 0;
    BaseQueuedTickets = 0;
    BaseCompleteTickets = 0;
    BaseCompleteTasks = 0;
    ErrorRate = .005;       %assumed base rate of 1 in 200 (.5%)
    Iteration = 1;
    IncTaskArrivalRate = 0.2024906;
    ReqTaskArrivalRate = 0.0930726;
    util_obs = 0.95;
    queue_obs = 4.0;
    ReqTimePDFLambdaObs = 0.1961;
    IncTimePDFLambdaObs = 0.1469;

%Independent variables determined through regression
    ServiceTimeAct = 0.3423;           %Value derived from non-linear
        regression, 1/10th of a day = 48 min
    ReqServiceTime = 0.5994;           %Initial guess average amount of
         time a work item requires .08 of a day = 38.4 min
```

```
    ProjTaskArrivalRate = 0.2411;      %Initial guess project task interarrival rate coefficient
    MaintTaskArrivalRate = 0.1543;     %Initial guess maint task interarrival rate coefficient
    AdminTaskArrivalRate = 0.1713;     %Initial guess admin task interarrival rate coefficient

    DESInputLabels = cat(1,"Error Rate","Service Time", "Iteration");
    DESInputHistory = cat(1,ErrorRate,ServiceTimeAct,Iteration);
    DESInputHistory = cat(2,DESInputLabels,DESInputHistory);

%iterate script

MaxIterations = 1;

for Iteration = 1:MaxIterations

%Run SimEvents simulation

mdlName = "iterateddes";

simIn = Simulink.SimulationInput(mdlName);

    simIn = setBlockParameter(simIn,"iterateddes/IncGen","IntergenerationTimeAction","dt = -"+IncTaskArrivalRate+"*log(1-rand());");
    simIn = setBlockParameter(simIn,"iterateddes/ReqGen","IntergenerationTimeAction","dt = -"+ReqTaskArrivalRate+"*log(1-rand());");
```

```
simIn = setBlockParameter(simIn,"iterateddes/ProjTaskGen","IntergenerationTimeAction","dt = -"+ProjTaskArrivalRate+"*log(1-rand());");
simIn = setBlockParameter(simIn,"iterateddes/MaintTaskGen","IntergenerationTimeAction","dt = -"+MaintTaskArrivalRate+"*log(1-rand());");
simIn = setBlockParameter(simIn,"iterateddes/AdminTaskGen","IntergenerationTimeAction","dt = -"+AdminTaskArrivalRate+"*log(1-rand());");

     simIn = setBlockParameter(simIn,"iterateddes/IncGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|1|1|0|1|"+ServiceTimeAct+"|2");
     simIn = setBlockParameter(simIn,"iterateddes/IncidentsfromReworkGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|1|1|0|1|"+ServiceTimeAct+"|2");
     simIn = setBlockParameter(simIn,"iterateddes/ReqGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|2|1|0|1|"+ServiceTimeAct+"|2");
     simIn = setBlockParameter(simIn,"iterateddes/ProjTaskGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|3|1|0|1|"+ServiceTimeAct+"|2");
     simIn = setBlockParameter(simIn,"iterateddes/MaintTaskGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|4|1|0|1|"+ServiceTimeAct+"|2");
simIn = setBlockParameter(simIn,"iterateddes/AdminTaskGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|5|1|0|1|"+ServiceTimeAct+"|1");
```

```
out = sim(simIn);

%Generate data

%WorkTypes = [1;2;3;4];
%WorkTypesV = [1 2 3 4 5];
IterationLength = out.SimulationMetadata.ModelInfo.StopTime;

%Totals

    %Stopped work E1
        SWE1TS = get(out.logsout,"CompSwitchEng1Stop").Values.WorkType;
        if isempty (SWE1TS.Time)
            StoppedIncidentsE1 = 0;
            StoppedRequestsE1 = 0;
            StoppedProjectTasksE1 = 0;
            StoppedMaintenanceTasksE1 = 0;
            StoppedAdminTasksE1 = 0;
        else
            StoppedWorkE1 = groupsummary(timetable2table(
               timeseries2timetable(SWE1TS)),"WorkType");
        %Incidents
            StoppedIncidentsE1T = StoppedWorkE1(find(StoppedWorkE1.
               WorkType == [1]),"GroupCount");
            if isempty(StoppedIncidentsE1T)
                StoppedIncidentsE1 = 0;
            else
                StoppedIncidentsE1 = StoppedIncidentsE1T{1,1};
```

```
        end
    %Requests
        StoppedRequestsE1T = StoppedWorkE1(find(StoppedWorkE1.WorkType == [2]),"GroupCount");
        if isempty(StoppedRequestsE1T)
            StoppedRequestsE1 = 0;
        else
            StoppedRequestsE1 = StoppedRequestsE1T{1,1};
        end
    %Project Tasks
        StoppedProjectTasksE1T = StoppedWorkE1(find(StoppedWorkE1.WorkType == [3]),"GroupCount");
        if isempty(StoppedProjectTasksE1T)
            StoppedProjectTasksE1 = 0;
        else
            StoppedProjectTasksE1 = StoppedProjectTasksE1T{1,1};
        end
    %Maintenance Tasks
        StoppedMaintenanceTasksE1T = StoppedWorkE1(find(StoppedWorkE1.WorkType == [4]),"GroupCount");
        if isempty(StoppedMaintenanceTasksE1T)
            StoppedMaintenanceTasksE1 = 0;
        else
            StoppedMaintenanceTasksE1 = StoppedMaintenanceTasksE1T{1,1};
        end
    %Admin Tasks
        StoppedAdminTasksE1T = StoppedWorkE1(find(StoppedWorkE1.WorkType == [5]),"GroupCount");
```

```
            if isempty(StoppedAdminTasksE1T)
                StoppedAdminTasksE1 = 0;
            else
                StoppedAdminTasksE1 = StoppedAdminTasksE1T{1,1};
            end
        end
        AllTicketsStoppedE1 = StoppedIncidentsE1 + StoppedRequestsE1;
        AllWorkTasksStoppedE1 = StoppedProjectTasksE1 + StoppedMaintenanceTasksE1;
        AllWorkStoppedE1 = AllTicketsStoppedE1 + AllWorkTasksStoppedE1;
    %Picked up work E1, not including admin tasks (serviced, but not
    %necessarily complete
        WBAE1 = get(out.logsout,"IndivQueueE1Out").Values.IsAdmin;
        if isempty (WBAE1.Time)
            PickedUpIncidentsE1 = 0;
            PickedUpRequestsE1 = 0;
            PickedUpProjectTasksE1 = 0;
            PickedUpMaintTasksE1 = 0;
        else
        WorkByAdminE1 = timetable2table(timeseries2timetable(WBAE1));
        WorkByTypeE1 = timetable2table(timeseries2timetable(get(out.logsout,"IndivQueueE1Out").Values.WorkType));
        WorkByTypeE1(:,1) = [];
        WorkByAdminTypeE1 = renamevars(addvars(WorkByAdminE1, WorkByTypeE1.WorkType),["Var3"],["WorkType"]);
        WorkByTypeE1NoAdmin = WorkByAdminTypeE1(~(WorkByAdminTypeE1.IsAdmin == 1),:);
        WorkByTypeE1NoAdmin = groupsummary(WorkByTypeE1NoAdmin,"WorkType");
```

```
        %Incidents
            PickedUpIncidentsE1T = WorkByTypeE1NoAdmin(find(
               WorkByTypeE1NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsE1T)
                PickedUpIncidentsE1 = 0;
            else
                PickedUpIncidentsE1 = PickedUpIncidentsE1T{1,1};
            end
        %Requests
            PickedUpRequestsE1T = WorkByTypeE1NoAdmin(find(
               WorkByTypeE1NoAdmin.WorkType == [2]),"GroupCount");
            if isempty(PickedUpRequestsE1T)
                PickedUpRequestsE1 = 0;
            else
                PickedUpRequestsE1 = PickedUpRequestsE1T{1,1};
            end
        %ProjectTasks
            PickedUpProjectTasksE1T = WorkByTypeE1NoAdmin(find(
               WorkByTypeE1NoAdmin.WorkType == [3]),"GroupCount");
            if isempty(PickedUpProjectTasksE1T)
                PickedUpProjectTasksE1 = 0;
            else
                PickedUpProjectTasksE1 = PickedUpProjectTasksE1T{1,1};
            end
        %MaintTasks
            PickedUpMaintTasksE1T = WorkByTypeE1NoAdmin(find(
               WorkByTypeE1NoAdmin.WorkType == [4]),"GroupCount");
            if isempty(PickedUpMaintTasksE1T)
                PickedUpMaintTasksE1 = 0;
```

```
            else
                PickedUpMaintTasksE1 = PickedUpMaintTasksE1T{1,1};
            end
        end
        AllTicketsPickedUpE1 = PickedUpIncidentsE1 + PickedUpRequestsE1;
        AllTasksPickedUpE1 = PickedUpProjectTasksE1 + PickedUpMaintTasksE1;
        AllWorkedPickedUpE1 = AllTicketsPickedUpE1 + AllTasksPickedUpE1;


    %Stopped work SE1
        SWSE1TS = get(out.logsout,"CompSwitchSrEng1Stop").Values.WorkType;
        if isempty (SWSE1TS.Time)
            StoppedIncidentsSE1 = 0;
            StoppedRequestsSE1 = 0;
            StoppedProjectTasksSE1 = 0;
            StoppedMaintenanceTasksSE1 = 0;
            StoppedAdminTasksSE1 = 0;
        else
        StoppedWorkSE1 = groupsummary(timetable2table(timeseries2timetable(SWSE1TS)),"WorkType");
        %Incidents
            StoppedIncidentsSE1T = StoppedWorkSE1(find(StoppedWorkSE1.WorkType == [1]),"GroupCount");
            if isempty(StoppedIncidentsSE1T)
                StoppedIncidentsSE1 = 0;
```

```
            else
                StoppedIncidentsSE1 = StoppedIncidentsSE1T{1,1};
            end
        %Requests
            StoppedRequestsSE1T = StoppedWorkSE1(find(StoppedWorkSE1.WorkType == [2]),"GroupCount");
            if isempty(StoppedRequestsSE1T)
                StoppedRequestsSE1 = 0;
            else
                StoppedRequestsSE1 = StoppedRequestsSE1T{1,1};
            end
        %Project Tasks
            StoppedProjectTasksSE1T = StoppedWorkSE1(find(StoppedWorkSE1.WorkType == [3]),"GroupCount");
            if isempty(StoppedProjectTasksSE1T)
                StoppedProjectTasksSE1 = 0;
            else
                StoppedProjectTasksSE1 = StoppedProjectTasksSE1T{1,1};
            end
        %Maintenance Tasks
            StoppedMaintenanceTasksSE1T = StoppedWorkSE1(find(StoppedWorkSE1.WorkType == [4]),"GroupCount");
            if isempty(StoppedMaintenanceTasksSE1T)
                StoppedMaintenanceTasksSE1 = 0;
            else
                StoppedMaintenanceTasksSE1 = StoppedMaintenanceTasksSE1T{1,1};
            end
        %Admin Tasks
```

```
            StoppedAdminTasksSE1T = StoppedWorkSE1(find(StoppedWorkSE1.WorkType == [5]),"GroupCount");
            if isempty(StoppedAdminTasksSE1T)
                StoppedAdminTasksSE1 = 0;
            else
                StoppedAdminTasksSE1 = StoppedAdminTasksSE1T{1,1};
            end
        end
        AllTicketsStoppedSE1 = StoppedIncidentsSE1 + StoppedRequestsSE1;
        AllWorkTasksStoppedSE1 = StoppedProjectTasksSE1 + StoppedMaintenanceTasksSE1;
        AllWorkStoppedSE1 = AllTicketsStoppedSE1 + AllWorkTasksStoppedSE1;

    %Picked up work SE1, not including admin tasks (serviced, but not
    %necessarily complete
        WBASE1 = get(out.logsout,"IndivQueueSE1Out").Values.IsAdmin;
        if isempty (WBASE1.Time)
            PickedUpIncidentsSE1 = 0;
            PickedUpRequestsSE1 = 0;
            PickedUpProjectTasksSE1 = 0;
            PickedUpMaintTasksSE1 = 0;
        else
        WorkByAdminSE1 = timetable2table(timeseries2timetable(WBASE1));
        WorkByTypeSE1 = timetable2table(timeseries2timetable(get(out.logsout,"IndivQueueSE1Out").Values.WorkType));
        WorkByTypeSE1(:,1) = [];
```

```
        WorkByAdminTypeSE1 = renamevars(addvars(WorkByAdminSE1,WorkByTypeSE1.WorkType),["Var3"],["WorkType"]);
        WorkByTypeSE1NoAdmin = WorkByAdminTypeSE1(~(WorkByAdminTypeSE1.IsAdmin == 1),:);
        WorkByTypeSE1NoAdmin = groupsummary(WorkByTypeSE1NoAdmin,"WorkType");
        %Incidents
            PickedUpIncidentsSE1T = WorkByTypeSE1NoAdmin(find(WorkByTypeSE1NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsSE1T)
                PickedUpIncidentsSE1 = 0;
            else
                PickedUpIncidentsSE1 = PickedUpIncidentsSE1T{1,1};
            end
        %Requests
            PickedUpRequestsSE1T = WorkByTypeSE1NoAdmin(find(WorkByTypeSE1NoAdmin.WorkType == [2]),"GroupCount");
            if isempty(PickedUpRequestsSE1T)
                PickedUpRequestsSE1 = 0;
            else
                PickedUpRequestsSE1 = PickedUpRequestsSE1T{1,1};
            end
        %ProjectTasks
            PickedUpProjectTasksSE1T = WorkByTypeSE1NoAdmin(find(WorkByTypeSE1NoAdmin.WorkType == [3]),"GroupCount");
            if isempty(PickedUpProjectTasksSE1T)
                PickedUpProjectTasksSE1 = 0;
            else
```

```
                PickedUpProjectTasksSE1 = PickedUpProjectTasksSE1T{1,1};
            end
        %MaintTasks
            PickedUpMaintTasksSE1T = WorkByTypeSE1NoAdmin(find(WorkByTypeSE1NoAdmin.WorkType == [4]),"GroupCount");
            if isempty(PickedUpMaintTasksSE1T)
                PickedUpMaintTasksSE1 = 0;
            else
                PickedUpMaintTasksSE1 = PickedUpMaintTasksSE1T{1,1};
            end
        end
        AllTicketsPickedUpSE1 = PickedUpIncidentsSE1 + PickedUpRequestsSE1;
        AllTasksPickedUpSE1 = PickedUpProjectTasksSE1 + PickedUpMaintTasksSE1;
        AllWorkedPickedUpSE1 = AllTicketsPickedUpSE1 + AllTasksPickedUpSE1;

    %Stopped work SE2
        SWSE2TS = get(out.logsout,"CompSwitchSrEng2Stop").Values.WorkType;
        if isempty (SWSE2TS.Time)
            StoppedIncidentsSE2 = 0;
            StoppedRequestsSE2 = 0;
            StoppedProjectTasksSE2 = 0;
            StoppedMaintenanceTasksSE2 = 0;
            StoppedAdminTasksSE2 = 0;
        else
```

```
        StoppedWorkSE2 = groupsummary(timetable2table(
           timeseries2timetable(SWSE2TS)),"WorkType");
        %Incidents
            StoppedIncidentsSE2T = StoppedWorkSE2(find(StoppedWorkSE2.
               WorkType == [1]),"GroupCount");
            if isempty(StoppedIncidentsSE2T)
                StoppedIncidentsSE2 = 0;
            else
                StoppedIncidentsSE2 = StoppedIncidentsSE2T{1,1};
            end
        %Requests
            StoppedRequestsSE2T = StoppedWorkSE2(find(StoppedWorkSE2.
               WorkType == [2]),"GroupCount");
            if isempty(StoppedRequestsSE2T)
                StoppedRequestsSE2 = 0;
            else
                StoppedRequestsSE2 = StoppedRequestsSE2T{1,1};
            end
        %Project Tasks
            StoppedProjectTasksSE2T = StoppedWorkSE2(find(
               StoppedWorkSE2.WorkType == [3]),"GroupCount");
            if isempty(StoppedProjectTasksSE2T)
                StoppedProjectTasksSE2 = 0;
            else
                StoppedProjectTasksSE2 = StoppedProjectTasksSE2T{1,1};
            end
        %Maintenance Tasks
            StoppedMaintenanceTasksSE2T = StoppedWorkSE2(find(
               StoppedWorkSE2.WorkType == [4]),"GroupCount");
```

```
        if isempty(StoppedMaintenanceTasksSE2T)
            StoppedMaintenanceTasksSE2 = 0;
        else
            StoppedMaintenanceTasksSE2 = StoppedMaintenanceTasksSE2T{1,1};
        end
    %Admin Tasks
        StoppedAdminTasksSE2T = StoppedWorkSE2(find(StoppedWorkSE2.WorkType == [5]),"GroupCount");
        if isempty(StoppedAdminTasksSE2T)
            StoppedAdminTasksSE2 = 0;
        else
            StoppedAdminTasksSE2 = StoppedAdminTasksSE2T{1,1};
        end
    end
    AllTicketsStoppedSE2 = StoppedIncidentsSE2 + StoppedRequestsSE2;
    AllWorkTasksStoppedSE2 = StoppedProjectTasksSE2 + StoppedMaintenanceTasksSE2;
    AllWorkStoppedSE2 = AllTicketsStoppedSE2 + AllWorkTasksStoppedSE2;
%Picked up work SE2, not including admin tasks (serviced, but not
%necessarily complete
    WBASE2 = get(out.logsout,"IndivQueueSE2Out").Values.IsAdmin;
    if isempty (WBASE2.Time)
        PickedUpIncidentsSE2 = 0;
        PickedUpRequestsSE2 = 0;
        PickedUpProjectTasksSE2 = 0;
        PickedUpMaintTasksSE2 = 0;
```

```
        else
        WorkByAdminSE2 = timetable2table(timeseries2timetable(WBASE2));
        WorkByTypeSE2 = timetable2table(timeseries2timetable(get(out.logsout,"IndivQueueSE2Out").Values.WorkType));
        WorkByTypeSE2(:,1) = [];
        WorkByAdminTypeSE2 = renamevars(addvars(WorkByAdminSE2,WorkByTypeSE2.WorkType),["Var3"],["WorkType"]);
        WorkByTypeSE2NoAdmin = WorkByAdminTypeSE2(~(WorkByAdminTypeSE2.IsAdmin == 1),:);
        WorkByTypeSE2NoAdmin = groupsummary(WorkByTypeSE2NoAdmin,"WorkType");
        %Incidents
            PickedUpIncidentsSE2T = WorkByTypeSE2NoAdmin(find(WorkByTypeSE2NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsSE2T)
                PickedUpIncidentsSE2 = 0;
            else
                PickedUpIncidentsSE2 = PickedUpIncidentsSE2T{1,1};
            end
        %Requests
            PickedUpRequestsSE2T = WorkByTypeSE2NoAdmin(find(WorkByTypeSE2NoAdmin.WorkType == [2]),"GroupCount");
            if isempty(PickedUpRequestsSE2T)
                PickedUpRequestsSE2 = 0;
            else
                PickedUpRequestsSE2 = PickedUpRequestsSE2T{1,1};
            end
        %ProjectTasks
```

```
        PickedUpProjectTasksSE2T = WorkByTypeSE2NoAdmin(find(
           WorkByTypeSE2NoAdmin.WorkType == [3]),"GroupCount");
        if isempty(PickedUpProjectTasksSE2T)
            PickedUpProjectTasksSE2 = 0;
        else
            PickedUpProjectTasksSE2 = PickedUpProjectTasksSE2T
               {1,1};
        end
    %MaintTasks
        PickedUpMaintTasksSE2T = WorkByTypeSE2NoAdmin(find(
           WorkByTypeSE2NoAdmin.WorkType == [4]),"GroupCount");
        if isempty(PickedUpMaintTasksSE2T)
            PickedUpMaintTasksSE2 = 0;
        else
            PickedUpMaintTasksSE2 = PickedUpMaintTasksSE2T{1,1};
        end
    end
    AllTicketsPickedUpSE2 = PickedUpIncidentsSE2 +
       PickedUpRequestsSE2;
    AllTasksPickedUpSE2 = PickedUpProjectTasksSE2 +
       PickedUpMaintTasksSE2;
    AllWorkedPickedUpSE2 = AllTicketsPickedUpSE2 +
       AllTasksPickedUpSE2;

%Stopped work SE3
    SWSE3TS = get(out.logsout,"CompSwitchSrEng3Stop").Values.
       WorkType;
    if isempty (SWSE3TS.Time)
        StoppedIncidentsSE3 = 0;
```

```
        StoppedRequestsSE3 = 0;
        StoppedProjectTasksSE3 = 0;
        StoppedMaintTasksSE3 = 0;
    else
    StoppedWorkSE3 = groupsummary(timetable2table(
       timeseries2timetable(SWSE3TS)),"WorkType");
    %Incidents
        StoppedIncidentsSE3T = StoppedWorkSE3(find(StoppedWorkSE3.
           WorkType == [1]),"GroupCount");
        if isempty(StoppedIncidentsSE3T)
            StoppedIncidentsSE3 = 0;
        else
            StoppedIncidentsSE3 = StoppedIncidentsSE3T{1,1};
        end
    %Requests
        StoppedRequestsSE3T = StoppedWorkSE3(find(StoppedWorkSE3.
           WorkType == [2]),"GroupCount");
        if isempty(StoppedRequestsSE3T)
            StoppedRequestsSE3 = 0;
        else
            StoppedRequestsSE3 = StoppedRequestsSE3T{1,1};
        end
    %Project Tasks
        StoppedProjectTasksSE3T = StoppedWorkSE3(find(
           StoppedWorkSE3.WorkType == [3]),"GroupCount");
        if isempty(StoppedProjectTasksSE3T)
            StoppedProjectTasksSE3 = 0;
        else
            StoppedProjectTasksSE3 = StoppedProjectTasksSE3T{1,1};
```

```
            end
         %MaintTasks
            StoppedMaintTasksSE3T = StoppedWorkSE3(find(StoppedWorkSE3.WorkType == [4]),"GroupCount");
            if isempty(StoppedMaintTasksSE3T)
                StoppedMaintTasksSE3 = 0;
            else
                StoppedMaintTasksSE3 = StoppedMaintTasksSE3T{1,1};
            end
        end
        AllTicketsStoppedSE3 = StoppedIncidentsSE3 + StoppedRequestsSE3;
        AllWorkTasksStoppedSE3 = StoppedProjectTasksSE3 + StoppedMaintTasksSE3;
        AllWorkStoppedSE3 = AllTicketsStoppedSE3 + AllWorkTasksStoppedSE3;
    %Picked up work SE3, not including admin tasks (serviced, but not
    %necessarily complete
        WBASE3 = get(out.logsout,"IndivQueueSE3Out").Values.IsAdmin
        if isempty (WBASE3.Time)
            PickedUpIncidentsSE3 = 0;
            PickedUpRequestsSE3 = 0;
            PickedUpProjectTasksSE3 = 0;
            PickedUpMaintTasksSE3 = 0;
        else
        WorkByAdminSE3 = timetable2table(timeseries2timetable(WBASE3));
        WorkByTypeSE3 = timetable2table(timeseries2timetable(get(out.logsout,"IndivQueueSE3Out").Values.WorkType));
        WorkByTypeSE3(:,1) = [];
```

```
        WorkByAdminTypeSE3 = renamevars(addvars(WorkByAdminSE3,WorkByTypeSE3.WorkType),["Var3"],["WorkType"]);
        WorkByTypeSE3NoAdmin = WorkByAdminTypeSE3(~(WorkByAdminTypeSE3.IsAdmin == 1),:);
        WorkByTypeSE3NoAdmin = groupsummary(WorkByTypeSE3NoAdmin,"WorkType");
        %Incidents
            PickedUpIncidentsSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsSE3T)
                PickedUpIncidentsSE3 = 0;
            else
                PickedUpIncidentsSE3 = PickedUpIncidentsSE3T{1,1};
            end
        %Requests
            PickedUpRequestsSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [2]),"GroupCount");
            if isempty(PickedUpRequestsSE3T)
                PickedUpRequestsSE3 = 0;
            else
                PickedUpRequestsSE3 = PickedUpRequestsSE3T{1,1};
            end
        %ProjectTasks
            PickedUpProjectTasksSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [3]),"GroupCount");
            if isempty(PickedUpProjectTasksSE3T)
                PickedUpProjectTasksSE3 = 0;
            else
```

```
                PickedUpProjectTasksSE3 = PickedUpProjectTasksSE3T{1,1};
            end
        %MaintTasks
            PickedUpMaintTasksSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [4]),"GroupCount");
            if isempty(PickedUpMaintTasksSE3T)
                PickedUpMaintTasksSE3 = 0;
            else
                PickedUpMaintTasksSE3 = PickedUpMaintTasksSE3T{1,1};
            end
        end
        AllTicketsPickedUpSE3 = PickedUpIncidentsSE3 + PickedUpRequestsSE3;
        AllTasksPickedUpSE3 = PickedUpProjectTasksSE3 + PickedUpMaintTasksSE3;
        AllWorkedPickedUpSE3 = AllTicketsPickedUpSE3 + AllTasksPickedUpSE3;

    %Total stopped work - all engineers
    AllTicketsStopped = AllTicketsStoppedE1 + AllTicketsStoppedSE1 + AllTicketsStoppedSE2 + AllTicketsStoppedSE3;
    AllWorkTasksStopped = AllWorkTasksStoppedE1 + AllWorkTasksStoppedSE1 + AllWorkTasksStoppedSE2 + AllWorkTasksStoppedSE3;
    AllWorkStopped = AllTicketsStopped + AllWorkTasksStopped;

    %Total picked up work - all engineers (excluding admin tasks)
```

```
    AllTicketsPickedUp = AllTicketsPickedUpE1 + AllTicketsPickedUpSE1 +
        AllTicketsPickedUpSE2 + AllTicketsPickedUpSE3;
    AllTasksPickedUp = AllTasksPickedUpE1 + AllTasksPickedUpSE1 +
       AllTasksPickedUpSE2 + AllTasksPickedUpSE3;
    AllWorkPickedUp = AllTicketsPickedUp + AllTasksPickedUp;

    %Generated work
        %Incidents
            if isempty(get(out.logsout,"IncGen").Values.IsAdmin.Data)
                IncidentsGenerated = 0;
            else
                IncidentsGenerated = height(timetable2table(
                   timeseries2timetable(get(out.logsout,"IncGen").
                   Values.IsAdmin)));
            end
        %Incidents from Change
            if isempty(get(out.logsout,"IncidentsfromReworkGen").Values
               .IsAdmin.Data)
                IncFromChgGenerated = 0;
            else
                IncFromChgGenerated = height(timetable2table(
                   timeseries2timetable(get(out.logsout,"
                   IncidentsfromReworkGen").Values.IsAdmin)));
            end
        %Requests
            if isempty(get(out.logsout,"ReqGen").Values.IsAdmin.Data)
                RequestsGenerated = 0;
            else
```

```
            RequestsGenerated = height(timetable2table(timeseries2timetable(get(out.logsout,"ReqGen").Values.IsAdmin)));
        end
    %Project Tasks
        if isempty(get(out.logsout,"ProjTaskGen").Values.IsAdmin.Data)
            ProjectTasksGenerated = 0;
        else
            ProjectTasksGenerated = height(timetable2table(timeseries2timetable(get(out.logsout,"ProjTaskGen").Values.IsAdmin)));
        end
    %Maintenance Tasks
        if isempty(get(out.logsout,"MaintTaskGen").Values.IsAdmin.Data)
            MaintenanceTasksGenerated = 0;
        else
            MaintenanceTasksGenerated = height(timetable2table(timeseries2timetable(get(out.logsout,"MaintTaskGen").Values.IsAdmin)));
        end
    %Admin Tasks
        if isempty(get(out.logsout,"AdminTaskGen").Values.IsAdmin.Data)
            AdminTasksGenerated = 0;
        else
```

```
                AdminTasksGenerated = height(timetable2table(timeseries2timetable(get(out.logsout,"AdminTaskGen").Values.IsAdmin)));
            end
        AllTicketsGenerated = RequestsGenerated + IncidentsGenerated;
        AllWorkTasksGenerated = ProjectTasksGenerated + IncFromChgGenerated + MaintenanceTasksGenerated;
        AllWorkGenerated = AllTicketsGenerated + AllWorkTasksGenerated;
        PercentTasks = AllWorkTasksGenerated./AllWorkGenerated;
        PercentTickets = AllTicketsGenerated./AllWorkGenerated;

    %Completed work
        %Incidents
            IncidentsCompletedTS = get(out.logsout,"IncComp").Values.IsAdmin;
            if isempty (IncidentsCompletedTS.Time)
                IncidentsCompletedTS = timeseries(0, 0);
            end
            IncidentsCompleted = height(timetable2table(timeseries2timetable(IncidentsCompletedTS)));
        %Requests
            RequestsCompletedTS = get(out.logsout,"ReqComp").Values.IsAdmin;
            if isempty (RequestsCompletedTS.Time)
                RequestsCompletedTS = timeseries(0, 0);
            end
            RequestsCompleted = height(timetable2table(timeseries2timetable(RequestsCompletedTS)));
        %Project Tasks
```

```
            ProjectTasksCompletedTS = get(out.logsout,"ProjTaskComp").
               Values.IsAdmin;
            if isempty (ProjectTasksCompletedTS.Time)
                ProjectTasksCompletedTS = timeseries(0, 0);
            end
            ProjectTasksCompleted = height(timetable2table(
               timeseries2timetable(ProjectTasksCompletedTS)));
        %Maintenance Tasks
            MaintenanceTasksCompletedTS = get(out.logsout,"
               MaintTaskComp").Values.IsAdmin;
            if isempty (MaintenanceTasksCompletedTS.Time)
                MaintenanceTasksCompletedTS = timeseries(0, 0);
            end
            MaintenanceTasksCompleted = height(timetable2table(
               timeseries2timetable(MaintenanceTasksCompletedTS)));
        %Admin Tasks
            AdminTasksCompletedTS = get(out.logsout,"AdminWorkComp").
               Values.IsAdmin;
            if isempty (AdminTasksCompletedTS.Time)
                AdminTasksCompletedTS = timeseries(0, 0);
            end
            AdminTasksCompleted = height(timetable2table(
               timeseries2timetable(AdminTasksCompletedTS)));
        AllTicketsCompleted = RequestsCompleted + IncidentsCompleted;
        AllWorkTasksCompleted = ProjectTasksCompleted +
           MaintenanceTasksCompleted;
        AllWorkCompleted = AllTicketsCompleted + AllWorkTasksCompleted;

    %Rates
```

```
    TicketStoppageRate = AllTicketsStopped./AllTicketsCompleted;
    TicketsStoppedPerDay = AllTicketsStopped./IterationLength;
    TicketsCompletedPerDay = AllTicketsCompleted./IterationLength;
    TaskStoppageRate = AllWorkTasksStopped./AllWorkTasksCompleted;
    TasksStoppedPerDay = AllWorkTasksStopped./IterationLength;
    TasksCompletedPerDay = AllWorkTasksCompleted./IterationLength;
    ReworkPerDay = IncFromChgGenerated./IterationLength;
    TicketPickupRate = AllTicketsPickedUp./AllTicketsGenerated;
    TicketsPickedUpPerDay = AllTicketsPickedUp./IterationLength;
    TaskPickupRate = AllTasksPickedUp./AllWorkTasksGenerated;
    TasksPickedUpPerDay = AllTasksPickedUp./IterationLength;
    WorkPickedUpPerDay = AllWorkPickedUp./IterationLength;
    TotalWorkSessions = AllTasksPickedUp + AllTicketsPickedUp;
    WorkSessionsPerDay = TotalWorkSessions./IterationLength;
    ErrorsPerDay = AllWorkCompleted*ErrorRate;
    BaseReworkRate = ErrorsPerDay*PercentTasks;
    BaseIncFromChange = ErrorsPerDay*PercentTickets;

    %Analyze DES Utilization data

%     IncrTime = seconds(0:0.0325:5);
    E1UtilTS = out.E1Utilization;
    if isempty(E1UtilTS.Time)
        E1UtilTS = timeseries(0,0);
    end
    E1UtilTT = timeseries2timetable(E1UtilTS);

    SE1UtilTS = out.SE1Utilization;
```

```
    if isempty(SE1UtilTS.Time)
        SE1UtilTS = timeseries(0,0);
    end
    SE1UtilTT = timeseries2timetable(SE1UtilTS);

    SE2UtilTS = out.SE2Utilization;
    if isempty(SE2UtilTS.Time)
        SE2UtilTS = timeseries(0,0);
    end
    SE2UtilTT = timeseries2timetable(SE2UtilTS);
%     SE2UtilTT = timeseries2timetable(out.SE2Utilization);

    SE3UtilTS = out.SE3Utilization;
    if isempty(SE3UtilTS.Time)
        SE3UtilTS = timeseries(0,0);
    end
    SE3UtilTT = timeseries2timetable(SE3UtilTS);
%     SE3UtilTT = timeseries2timetable(out.SE3Utilization);

    TTUtilsync = synchronize(E1UtilTT, SE1UtilTT, SE2UtilTT, SE3UtilTT,
        'union', 'previous');
    TTUtilsync.AvgUtil = mean(TTUtilsync{:,{'Data_E1UtilTT','
       Data_SE1UtilTT','Data_SE2UtilTT','Data_SE3UtilTT'}},2);

    util_pred = TTUtilsync.AvgUtil(end);

    %Analyze DES Total Time data
    IncTimeTS = get(out.logsout,"TotalTimeInc").Values;
    if isempty(IncTimeTS.Time)
```

```
        IncTimeTS = timeseries(0,0);
    end
    IncTimeTT = timeseries2timetable(IncTimeTS);
    IncTimeTTColName = IncTimeTT.Properties.VariableNames{1};
    if isempty(IncTimeTS.Time)
        IncTimePDFLambdaSim = 0;
    else
        IncTimeMean = mean(IncTimeTT.(IncTimeTTColName));
        IncTimePDFLambdaSim = 1/IncTimeMean;
    end

    ReqTimeTS = get(out.logsout,"TotalTimeReq").Values;
    if isempty(ReqTimeTS.Time)
        ReqTimeTS = timeseries(0,0);
    end
    ReqTimeTT = timeseries2timetable(ReqTimeTS);
    ReqTimeTTColName = ReqTimeTT.Properties.VariableNames{1};
    if isempty(ReqTimeTS.Time)
        ReqTimePDFLambdaSim = 0;
    else
        ReqTimeMean = mean(ReqTimeTT.(ReqTimeTTColName));
        ReqTimePDFLambdaSim = 1/ReqTimeMean;
    end

    ProjTimeTS = get(out.logsout,"TotalTimeProj").Values;
    if isempty(ProjTimeTS.Time)
        ProjTimeTS = timeseries(0,0);
    end
    ProjTimeTT = timeseries2timetable(ProjTimeTS);
```

```
MaintTimeTS = get(out.logsout,"TotalTimeMaint").Values;
if isempty(MaintTimeTS.Time)
    MaintTimeTS = timeseries(0,0);
end
MaintTimeTT = timeseries2timetable(MaintTimeTS);

AdminTimeTS = get(out.logsout,"TotalTimeAdmin").Values;
if isempty(AdminTimeTS.Time)
    AdminTimeTS = timeseries(0,0);
end
AdminTimeTT = timeseries2timetable(AdminTimeTS);

IncTimeTTData = IncTimeTT;
IncTimeTTData.Properties.VariableNames{(IncTimeTTColName)} = 'Data';
ReqTimeTTData = ReqTimeTT;
ReqTimeTTData.Properties.VariableNames{(ReqTimeTTColName)} = 'Data';
ProjTimeTTData = ProjTimeTT;
ProjTimeTTData.Properties.VariableNames{'TotalTimeProj'} = 'Data';
MaintTimeTTData = MaintTimeTT;
MaintTimeTTData.Properties.VariableNames{'TotalTimeMaint'} = 'Data';
AdminTimeTTData = AdminTimeTT;
AdminTimeTTData.Properties.VariableNames{'TotalTimeAdmin'} = 'Data';
AllTimeTT = [IncTimeTTData; ReqTimeTTData; ProjTimeTTData;
    MaintTimeTTData; AdminTimeTTData];
```

```
  %Determine exponential distribution for all data and extract
  %coefficient (lambda)
  AllTimePDF = fitdist(AllTimeTT.Data,'Exponential');
  AllTimePDFlambda = 1/AllTimePDF.mu;


 %Analyze queue depth
  E1QueueTS = out.E1QueueLength;
  if isempty(E1QueueTS.Time)
      E1QueueTS = timeseries(0,0);
  end
  E1QueueTT = timeseries2timetable(E1QueueTS);

  SE1QueueTS = out.SE1QueueLength;
  if isempty(SE1QueueTS.Time)
      SE1QueueTS = timeseries(0,0);
  end
  SE1QueueTT = timeseries2timetable(SE1QueueTS);

  SE2QueueTS = out.SE2QueueLength;
  if isempty(SE2QueueTS.Time)
      SE2QueueTS = timeseries(0,0);
  end
  SE2QueueTT = timeseries2timetable(SE2QueueTS);

  SE3QueueTS = out.SE3QueueLength;
  if isempty(SE3QueueTS.Time)
      SE3QueueTS = timeseries(0,0);
```

```
    end
    SE3QueueTT = timeseries2timetable(SE3QueueTS);

    TTQueuesync = synchronize(E1QueueTT, SE1QueueTT, SE2QueueTT, SE3QueueTT, 'union', 'previous');
    TTQueuesync.AvgQueue = mean(TTQueuesync{:,{'Data_E1QueueTT','Data_SE1QueueTT','Data_SE2QueueTT','Data_SE3QueueTT'}},2);

    queue_pred = TTQueuesync.AvgQueue(end);

    %Close the sddatain.xlsx file to prevent write error
        % Attach to an existing instance of Excel
        try
            excelApp = actxGetRunningServer('Excel.Application');
        catch
            error('No running instance of Excel found.');
        end

        % Specify the workbook name to close
        targetWorkbookName = 'sddatain.xlsx'; % The name of the workbook to search for

        % List all open workbooks
        disp('Listing all open workbooks...');
        for i = 1:excelApp.Workbooks.Count
            disp(['Workbook ', num2str(i), ': ', excelApp.Workbooks.Item(i).FullName]);
        end

```

```
        % Search for any workbook with the specified name and close it
        for i = excelApp.Workbooks.Count:-1:1 % Iterate backwards to avoid indexing issues when closing
            if strcmpi(excelApp.Workbooks.Item(i).Name, targetWorkbookName)
                disp(['Closing workbook: ', excelApp.Workbooks.Item(i).FullName]);
                % Close the workbook (use false to discard changes)
                excelApp.Workbooks.Item(i).Close(false);
            end
        end

        % Release the COM object (optional, if no further operations needed)
        release(excelApp);
        disp('All matching workbooks closed.');

    %Output to Vensim
    SDParameters = cat(1,TicketsPickedUpPerDay,TasksPickedUpPerDay,TicketsStoppedPerDay,TasksStoppedPerDay,TicketsCompletedPerDay,TasksCompletedPerDay,BaseReworkRate,BaseIncFromChange,BaseMgmtPress,BaseFatigue,BaseQueuedTasks,BaseQueuedTickets,BaseMgmtPreempt,BaseCompleteTasks,BaseCompleteTickets,Iteration);
    writematrix(SDParameters, "C:\Users\enos9\OneDrive - Colostate\combined\sddatain.xlsx",'Sheet',1);

    %Record DES iteration data
```

```
    DESDataLabels = ["TicketsPickedUpPerDay";"TasksPickedUpPerDay";"TicketsStoppedPerDay";"TasksStoppedPerDay";"TicketsCompletedPerDay";"TasksCompletedPerDay";"BaseReworkRate";"BaseIncFromChange";"BaseMgmtPress";"BaseFatigue";"BaseQueuedTasks";"BaseQueuedTickets";"BaseMgmtPreempt";"BaseCompleteTasks";"BaseCompleteTickets";"Iteration"];
    if Iteration == 1
        DESOutputHistory = cat(2,DESDataLabels,SDParameters);
    else
        DESOutputHistory = cat(2,DESOutputHistory,SDParameters);
    end

%Trigger Vensim simulation, uses sddatain.xlsx and generates sddataout.xlsx

    %!"C:\Program Files\Vensim\vendss64.exe" "C:\Users\enos9\OneDrive -
    %Colostate\combined\scripts\sdcommand.cmd" is the old command

    % Define external command
    externalCommand = '"C:\Program Files\Vensim\vendss64.exe" "C:\Users\enos9\OneDrive - Colostate\combined\scripts\sdcommand.cmd"';

    % Use the start command to launch the process asynchronously
    system(['start "" ' externalCommand]);

    % Define timeout duration in seconds
    timeoutDuration = 90;

    % Monitor execution
```

```
    disp('Monitoring the external program...');
    startTime = tic;

    while true
        % Check if the process is running
        [~, result] = system('tasklist');
        if ~contains(result, 'taskkill /IM vendss64.exe') % Program no
           longer running
            disp('External program completed successfully.');
            break;
        end

        % Check elapsed time
        elapsedTime = toc(startTime);
        if elapsedTime > timeoutDuration
            disp('Timeout exceeded. Terminating external program...');

            % Kill the process if it exceeds the timeout
            system('taskkill /IM vendss64.exe /F /T');
            break;
        end

        % Pause briefly to avoid overloading the system with checks
        pause(1);

    end

%Post-process the Vensim output
```

```
SDDataOut = readmatrix("C:\Users\enos9\OneDrive - Colostate\combined\
    sddataout.xlsx",'Sheet',1); %extract data from Excel file

%Find Error Rate
    SDDataOutVal = SDDataOut;
    SDDataOutVal(:,1) = []; % remove labels
    SDDataOutValWithTime = SDDataOutVal; %Retain data with Time row
    SDDataOutVal(1,:) = []; % remove time row
    SDDataOutValMean = mean(SDDataOutVal,2); %average of each row
       across 5 days
    SDReworkRateMean = SDDataOutValMean(7,1); %average rework rate
    SDIncFrChgMean = SDDataOutValMean(6,1); %average rate of incidents
       from change
    SDErrorRateMean = SDReworkRateMean+SDIncFrChgMean; %Average
       combined errors / day
    SDErrorsOnRun = SDErrorRateMean*5; %Total errors during 5 day run
    SDCompTasksMean = SDDataOutValMean(1,1);
    if SDCompTasksMean < 0
        SDCompTasksMean = 0;
    end
    SDCompTicketsMean = SDDataOutValMean(2,1);
    SDAvgReworkPercentage = SDReworkRateMean / SDCompTasksMean;
    SDAvgIncFromChangePercentage = SDIncFrChgMean / SDCompTicketsMean;
    ErrorRate = (SDReworkRateMean + SDIncFrChgMean) / (SDCompTasksMean
       + SDCompTicketsMean); %Average error rate during run


%Find total service Time
    SDTasksCompleted = SDDataOut(2,7);
```

```
    SDTicketsCompleted = SDDataOut(3,7);
    SDTotalWorkCompleted = SDTasksCompleted+SDTicketsCompleted;
    SDAvgCompPerDay = SDTotalWorkCompleted / 5;
    SDServiceTimeActNew = 1 / SDAvgCompPerDay;  %THIS IS NOT THE RIGHT VALUE FOR DES Service Time - this is total elapsed time

%Find change in service time and calculate next iteration service time
    SDMgmtPreemptEnd = SDDataOut(5,7);
    SDMgmtChange = SDMgmtPreemptEnd-BaseMgmtPreempt;
    SDSvcTimeChange = (1+SDMgmtChange);
    ServiceTimeAct = ServiceTimeAct*SDSvcTimeChange;

%Record SD iteration data
    SDDataLabels = ["Time";"Completed Tasks";"Completed Tickets";"Fatigue";"Management Preemption";"Management Pressure";"Rate Inc from Change";"Rework Rate";"Task Completion Rate";"Task Pickup Rate";"Task Stop Rate";"Ticket Completion Rate";"Ticket Pickup Rate";"Ticket Stop Rate";"Time Required to Complete";"Total Queued Tasks";"Total Queued Tickets"];
    if Iteration == 1
        SDOutputHistory = cat(2,SDDataLabels,SDDataOutValWithTime);
    else
        SDOutputHistory = cat(2,SDOutputHistory,SDDataOutValWithTime);
    end

%Record DES next iteration data
    IterationNew = Iteration+1;
    DESInputHistoryNew = cat(1,ErrorRate,ServiceTimeAct,IterationNew);
    DESInputHistory = cat(2,DESInputHistory,DESInputHistoryNew);
```

```
%Extract carry-over variables
    BaseMgmtPreempt = SDDataOut(5,7);
    BaseMgmtPress = SDDataOut(6,7);
    BaseFatigue = SDDataOut(4,7)*.75; %Fatigue reduces somewhat over a
        weekend (between iterations)
    BaseQueuedTasks = SDDataOut(16,7);
    BaseQueuedTickets = SDDataOut(17,7);
    BaseCompleteTickets = SDTicketsCompleted;
    BaseCompleteTasks = SDTasksCompleted;

%Calculate residuals
    residuals(1) = util_pred - util_obs;
    residuals(2) = IncTimePDFLambdaSim - IncTimePDFLambdaObs;
    residuals(3) = ReqTimePDFLambdaSim - ReqTimePDFLambdaObs;
    residuals(4) = queue_pred - queue_obs;
    residuals = residuals(:);

end
```

A.2 Vensim Script

```
SPECIAL>NOINTERACTION
SPECIAL>LOADMODEL|"C:\Users\enos9\OneDrive - Colostate\combined\iteratedsd.
MENU>RUN|O
MENU>VDF2XLSX|!|sddataout.xlsx|sddataout.lst
MENU>EXIT
```

Appendix B

Appendix B: Regression Scripts

B.1 Regression Script

```
% Establish parameters - intial guesses and bounds

    ServiceTimeAct = 0.3424;                %Starting average amount of time engineer has available to work
    ServiceTimeActMin = 0.06;            %Lower bound average amount of time engineer has available to work, ~.5 min (28.8 min)
    ServiceTimeActMax = 0.24;            %Upper bound average amount of time engineer has available to work, ~2 hrs (115.2 min)

    ReqServiceTime = 1.0270;                %Initial guess average amount of time a work item requires
    ReqServiceTimeMin = 0.06;            %Lower bound average amount of time a work item requires, ~.5 min (28.8 min)
    ReqServiceTimeMax = 0.24;            %Upper bound Average amount of time a work item requires, ~2 hrs (115.2 min)

    ProjTaskArrivalRate = 0.2553;           %Initial guess project task interarrival rate coefficient
    ProjTaskArrivalRateMin = 0.02128;    %Lower bound project task interarrival rate coefficient
    ProjTaskArrivalRateMax = 0.34235;    %Upper bound project task interarrival rate coefficient

```

```
    MaintTaskArrivalRate = 0.1229;       %Initial guess maint task interarrival rate coefficient
    MaintTaskArrivalRateMin = 0.06386;  %Lower bound maint task interarrival rate coefficient
    MaintTaskArrivalRateMax = 1.02704;  %Upper bound maint task interarrival rate coefficient

    AdminTaskArrivalRate = 0.2334;         %Initial guess admin task interarrival rate coefficient
    AdminTaskArrivalRateMin = 0.06383;  %Lower bound admin task interarrival rate coefficient
    AdminTaskArrivalRateMax = 0.25532;  %Upper bound admin task interarrival rate coefficient

    lb = [ProjTaskArrivalRateMin, MaintTaskArrivalRate, AdminTaskArrivalRate, ServiceTimeActMin, ReqServiceTimeMin];
    ub = [ProjTaskArrivalRateMax, MaintTaskArrivalRateMax, AdminTaskArrivalRateMax, ServiceTimeActMax, ReqServiceTimeMax];
    x0 = [ProjTaskArrivalRate, MaintTaskArrivalRate, AdminTaskArrivalRate, ServiceTimeAct, ReqServiceTime];

% Automatically calculate Jacobian (fails as undefined with certain
% options)

    % Establish objective function for use with Automatic Jacobian calculation
%function residuals = myObjectiveFun(lambdas)
    % lambdas: [ProjTaskArrivalRate, MaintTaskArrivalRate, AdminTaskArrivalRate, ReqServiceTime, ServiceTimeAct]
```

```
    %    ProjTaskArrivalRate = lambdas(1);
    %    MaintTaskArrivalRate = lambdas(2);
    %    AdminTaskArrivalRate = lambdas(3);
    %    ReqServiceTime = lambdas(4);
    %    ServiceTimeAct = lambdas(5);
    %    run('CombinedSim.m');
    %    residuals(1) = util_pred - util_obs;
    %    residuals(2) = IncTimePDFLambdaSim - IncTimePDFLambdaObs;
    %    residuals(3) = ReqTimePDFLambdaSim - ReqTimePDFLambdaObs;
    %    residuals(4) = queue_pred - queue_obs;
    %    residuals = residuals(:);
%end

    %function [r, J] = myObjFunAndJacobian(x)
    % x: [ProjTaskArrivalRate, MaintTaskArrivalRate,
       AdminTaskArrivalRate, ReqServiceTime, ServiceTimeAct]
%    ProjTaskArrivalRate = x(1);
%    MaintTaskArrivalRate = x(2);
%    AdminTaskArrivalRate = x(3);
%    ReqServiceTime = x(4);
%    ServiceTimeAct = x(5);
%    r = runSimEventsModel(x);
%    n = numel(x);
%    m = numel(r);
%    J = zeros(m, n);
%    stepSize = 1e-8;
%    for i = 1:n
%        xMinus = x;
%        xPlus = x;
```

```
%         xMinus(i) = xMinus(i) - stepSize;
%         xPlus(i) = xPlus(i) + stepSize;
%         rMinus = runSimEventsModel(xMinus);
%         rPlus = runSimEventsModel(xPlus);
%         J(:, i) = (rPlus - rMinus) / (2 * stepSize);
%     end
%end

%Derivative-based regression using lsqnonlin

    %Automatic Jacobian calculation
%     options = optimoptions('lsqnonlin','Display','iter','MaxFunctionEvaluations', 300, 'FiniteDifferenceType','central', 'StepTolerance', 1e-12,'FunctionTolerance', 1e-12,'OptimalityTolerance', 1e-12,'Algorithm','trust-region-reflective');
%     [estimatedLambdas, resnorm, residuals, exitflag, output] = lsqnonlin(@myObjectiveFun, x0, lb, ub, options);

    %Manual Jacobian calculation
    %options = optimoptions('lsqnonlin','Display','iter','MaxFunctionEvaluations', 50, 'StepTolerance', 1e-12,'FunctionTolerance', 1e-12,'OptimalityTolerance', 1e-12,'Algorithm','trust-region-reflective','Jacobian','on');
    %[estimatedLambdas, ~, residuals, exitflag, output] = lsqnonlin(@myObjFunAndJacobian, x0, lb, ub, options);

%Derivative-free regression

function cost = myCostFun(x)
```

```
    ProjTaskArrivalRate = x(1);
    MaintTaskArrivalRate = x(2);
    AdminTaskArrivalRate = x(3);
    ReqServiceTime = x(4);
    ServiceTimeAct = x(5);

    run('CombinedSim.m');

    residuals(1) = util_pred - util_obs;
    residuals(2) = IncTimePDFLambdaSim - IncTimePDFLambdaObs;
    residuals(3) = ReqTimePDFLambdaSim - ReqTimePDFLambdaObs;
    residuals(4) = queue_pred - queue_obs;
    residuals = residuals(:);

    cost = residuals(1)^2 + residuals(2)^2 + residuals(3)^2 + residuals
        (4)^2;

end



%Patternsearch
%    options = optimoptions('patternsearch', ...
%        'Display','iter', ...
%        'InitialMeshSize', 1,  ...
%        'MeshExpansionFactor', 2, ...
%        'MeshContractionFactor', 0.5, ...
%        'MaxFunctionEvaluations', 5000, ...
```

```
%         'StepTolerance', 1e-8, ...
%         'FunctionTolerance', 1e-8,...
%         'UseParallel',true);
%     [xOpt, fval, exitflag, output] = patternsearch(@myCostFun, x0, [], [], [], [], lb, ub, [], options)

%Particleswarm (can't run with 'UseParallel', true due to Vensim call)
%With nVars =5, recommendation is for 'SwarmSize', 100 - 150,  'MaxIterations',
%500 - 1000
    nVars = 5;
    options = optimoptions('particleswarm', ...
        'Display','iter', ...
        'SwarmSize', 100,  ...
        'MaxIterations', 1000, ...
        'MaxStallIterations', 20);
    [xOpt, fval, exitflag, output] = particleswarm(@myCostFun, nVars, lb, ub, options)

%Genetic Algorithm
%With nVars =5, recommendation is for 'PopulationSize', 75,  'MaxGenerations', 500
%     nVars = 5;
%     options = optimoptions('ga','Display','iter', 'PopulationSize', 75,  'MaxGenerations', 500,  'MaxStallGenerations', 20);
%     [xOpt, fval, exitflag, output, population, scores] = ga(@myCostFun, nVars, [], [], [], [], lb, ub, [], options)


```

```
%Extract residuals based on xOpt
%    function resVec = computeResiduals(x)
%        run('CombinedSim.m');
%        logs = simOut.logsout;
%        util_pred = logs.getElement('util_pred').Values.Data(end);
%        util_obs = logs.getElement('util_obs').Values.Data(end);
%        IncTimePDFLambdaSim = logs.getElement('IncTimePDFLambdaSim').Values.Data(end);
%        IncTimePDFLambdaObs = logs.getElement('IncTimePDFLambdaObs').Values.Data(end);
%        ReqTimePDFLambdaSim = logs.getElement('ReqTimePDFLambdaSim').Values.Data(end);
%        ReqTimePDFLambdaObs = logs.getElement('ReqTimePDFLambdaObs').Values.Data(end);
%        queue_pred = logs.getElement('queue_pred').Values.Data(end);
%        queue_obs = logs.getElement('queue_obs').Values.Data(end);
%        residuals(1) = util_pred - util_obs;
%        residuals(2) = IncTimePDFLambdaSim - IncTimePDFLambdaObs;
%        residuals(3) = ReqTimePDFLambdaSim - ReqTimePDFLambdaObs;
%        residuals(4) = queue_pred - queue_obs;
%        residuals = residuals(:);
%    end
%finalRes = computeResiduals(xOpt);


\secition{Simplified SimEvents Script}

%Initialize SD Variables for first run

```

```
    IncTaskArrivalRate = 0.2024906;
    ReqTaskArrivalRate = 0.0930726;
    util_obs = 0.95;
    queue_obs = 4.0;
    ReqTimePDFLambdaObs = 0.1961;
    IncTimePDFLambdaObs = 0.1469;

    BaseMgmtPreempt = 1;
    BaseMgmtPress = 1;
    BaseFatigue = 1;
    BaseQueuedTasks = 0;
    BaseQueuedTickets = 0;
    BaseCompleteTickets = 0;
    BaseCompleteTasks = 0;
    ErrorRate = .005;       %assumed base rate of 1 in 200 (.5%)
    Iteration = 1;


    DESInputLabels = cat(1,"Error Rate","Service Time", "Iteration");
    DESInputHistory = cat(1,ErrorRate,ServiceTimeAct,Iteration);
    DESInputHistory = cat(2,DESInputLabels,DESInputHistory);



%iterate script

MaxIterations = 1;

for Iteration = 1:MaxIterations
```

```
%Run SimEvents simulation

mdlName = "iterateddes";

simIn = Simulink.SimulationInput(mdlName);

    simIn = setBlockParameter(simIn,"iterateddes/IncGen","
       IntergenerationTimeAction","dt = -"+IncTaskArrivalRate+"*log(1-
       rand());");
    simIn = setBlockParameter(simIn,"iterateddes/ReqGen","
       IntergenerationTimeAction","dt = -"+ReqTaskArrivalRate+"*log(1-
       rand());");
    simIn = setBlockParameter(simIn,"iterateddes/ProjTaskGen","
       IntergenerationTimeAction","dt = -"+ProjTaskArrivalRate+"*log(1-
       rand());");
    simIn = setBlockParameter(simIn,"iterateddes/MaintTaskGen","
       IntergenerationTimeAction","dt = -"+MaintTaskArrivalRate+"*log
       (1-rand());");
    simIn = setBlockParameter(simIn,"iterateddes/AdminTaskGen","
       IntergenerationTimeAction","dt = -"+AdminTaskArrivalRate+"*log
       (1-rand());");

        simIn = setBlockParameter(simIn,"iterateddes/IncGen","
           AttributeInitialValue","0|0|0|"+ReqServiceTime
           +"|0|0|0|0|0|"+ErrorRate+"|1|1|0|1|"+ServiceTimeAct+"|2");
        simIn = setBlockParameter(simIn,"iterateddes/
           IncidentsfromReworkGen","AttributeInitialValue","0|0|0|"+
```

```
ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|1|1|0|1|"+ServiceTimeAct+"|2");
        simIn = setBlockParameter(simIn,"iterateddes/ReqGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|2|1|0|1|"+ServiceTimeAct+"|2");
        simIn = setBlockParameter(simIn,"iterateddes/ProjTaskGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|3|1|0|1|"+ServiceTimeAct+"|2");
        simIn = setBlockParameter(simIn,"iterateddes/MaintTaskGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|4|1|0|1|"+ServiceTimeAct+"|2");
    simIn = setBlockParameter(simIn,"iterateddes/AdminTaskGen","AttributeInitialValue","0|0|0|"+ReqServiceTime+"|0|0|0|0|0|"+ErrorRate+"|5|1|0|1|"+ServiceTimeAct+"|1");

out = sim(simIn);

%Generate data

%WorkTypes = [1;2;3;4];
%WorkTypesV = [1 2 3 4 5];
IterationLength = out.SimulationMetadata.ModelInfo.StopTime;

%Totals

    %Stopped work E1
        SWE1TS = get(out.logsout,"CompSwitchEng1Stop").Values.WorkType;
        if isempty (SWE1TS.Time)
            StoppedIncidentsE1 = 0;
```

```
            StoppedRequestsE1 = 0;
            StoppedProjectTasksE1 = 0;
            StoppedMaintenanceTasksE1 = 0;
            StoppedAdminTasksE1 = 0;
        else
            StoppedWorkE1 = groupsummary(timetable2table(
               timeseries2timetable(SWE1TS)),"WorkType");
        %Incidents
            StoppedIncidentsE1T = StoppedWorkE1(find(StoppedWorkE1.
               WorkType == [1]),"GroupCount");
            if isempty(StoppedIncidentsE1T)
                StoppedIncidentsE1 = 0;
            else
                StoppedIncidentsE1 = StoppedIncidentsE1T{1,1};
            end
        %Requests
            StoppedRequestsE1T = StoppedWorkE1(find(StoppedWorkE1.
               WorkType == [2]),"GroupCount");
            if isempty(StoppedRequestsE1T)
                StoppedRequestsE1 = 0;
            else
                StoppedRequestsE1 = StoppedRequestsE1T{1,1};
            end
        %Project Tasks
            StoppedProjectTasksE1T = StoppedWorkE1(find(StoppedWorkE1.
               WorkType == [3]),"GroupCount");
            if isempty(StoppedProjectTasksE1T)
                StoppedProjectTasksE1 = 0;
            else
```

```
                StoppedProjectTasksE1 = StoppedProjectTasksE1T{1,1};
            end
        %Maintenance Tasks
            StoppedMaintenanceTasksE1T = StoppedWorkE1(find(StoppedWorkE1.WorkType == [4]),"GroupCount");
            if isempty(StoppedMaintenanceTasksE1T)
                StoppedMaintenanceTasksE1 = 0;
            else
                StoppedMaintenanceTasksE1 = StoppedMaintenanceTasksE1T{1,1};
            end
        %Admin Tasks
            StoppedAdminTasksE1T = StoppedWorkE1(find(StoppedWorkE1.WorkType == [5]),"GroupCount");
            if isempty(StoppedAdminTasksE1T)
                StoppedAdminTasksE1 = 0;
            else
                StoppedAdminTasksE1 = StoppedAdminTasksE1T{1,1};
            end
        end
        AllTicketsStoppedE1 = StoppedIncidentsE1 + StoppedRequestsE1;
        AllWorkTasksStoppedE1 = StoppedProjectTasksE1 + StoppedMaintenanceTasksE1;
        AllWorkStoppedE1 = AllTicketsStoppedE1 + AllWorkTasksStoppedE1;
    %Picked up work E1, not including admin tasks (serviced, but not
    %necessarily complete
        WBAE1 = get(out.logsout,"IndivQueueE1Out").Values.IsAdmin;
        if isempty (WBAE1.Time)
            PickedUpIncidentsE1 = 0;
```

```
            PickedUpRequestsE1 = 0;
            PickedUpProjectTasksE1 = 0;
            PickedUpMaintTasksE1 = 0;
        else
        WorkByAdminE1 = timetable2table(timeseries2timetable(WBAE1));
        WorkByTypeE1 = timetable2table(timeseries2timetable(get(out.logsout,"IndivQueueE1Out").Values.WorkType));
        WorkByTypeE1(:,1) = [];
        WorkByAdminTypeE1 = renamevars(addvars(WorkByAdminE1, WorkByTypeE1.WorkType),["Var3"],["WorkType"]);
        WorkByTypeE1NoAdmin = WorkByAdminTypeE1(~(WorkByAdminTypeE1.IsAdmin == 1),:);
        WorkByTypeE1NoAdmin = groupsummary(WorkByTypeE1NoAdmin,"WorkType");
        %Incidents
            PickedUpIncidentsE1T = WorkByTypeE1NoAdmin(find(WorkByTypeE1NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsE1T)
                PickedUpIncidentsE1 = 0;
            else
                PickedUpIncidentsE1 = PickedUpIncidentsE1T{1,1};
            end
        %Requests
            PickedUpRequestsE1T = WorkByTypeE1NoAdmin(find(WorkByTypeE1NoAdmin.WorkType == [2]),"GroupCount");
            if isempty(PickedUpRequestsE1T)
                PickedUpRequestsE1 = 0;
            else
                PickedUpRequestsE1 = PickedUpRequestsE1T{1,1};
```

```
            end
        %ProjectTasks
            PickedUpProjectTasksE1T = WorkByTypeE1NoAdmin(find(WorkByTypeE1NoAdmin.WorkType == [3]),"GroupCount");
            if isempty(PickedUpProjectTasksE1T)
                PickedUpProjectTasksE1 = 0;
            else
                PickedUpProjectTasksE1 = PickedUpProjectTasksE1T{1,1};
            end
        %MaintTasks
            PickedUpMaintTasksE1T = WorkByTypeE1NoAdmin(find(WorkByTypeE1NoAdmin.WorkType == [4]),"GroupCount");
            if isempty(PickedUpMaintTasksE1T)
                PickedUpMaintTasksE1 = 0;
            else
                PickedUpMaintTasksE1 = PickedUpMaintTasksE1T{1,1};
            end
        end
        AllTicketsPickedUpE1 = PickedUpIncidentsE1 + PickedUpRequestsE1;
        AllTasksPickedUpE1 = PickedUpProjectTasksE1 + PickedUpMaintTasksE1;
        AllWorkedPickedUpE1 = AllTicketsPickedUpE1 + AllTasksPickedUpE1;


    %Stopped work SE1
        SWSE1TS = get(out.logsout,"CompSwitchSrEng1Stop").Values.WorkType;
```

```
        if isempty (SWSE1TS.Time)
            StoppedIncidentsSE1 = 0;
            StoppedRequestsSE1 = 0;
            StoppedProjectTasksSE1 = 0;
            StoppedMaintenanceTasksSE1 = 0;
            StoppedAdminTasksSE1 = 0;
        else
        StoppedWorkSE1 = groupsummary(timetable2table(
           timeseries2timetable(SWSE1TS)),"WorkType");
        %Incidents
            StoppedIncidentsSE1T = StoppedWorkSE1(find(StoppedWorkSE1.
               WorkType == [1]),"GroupCount");
            if isempty(StoppedIncidentsSE1T)
                StoppedIncidentsSE1 = 0;
            else
                StoppedIncidentsSE1 = StoppedIncidentsSE1T{1,1};
            end
        %Requests
            StoppedRequestsSE1T = StoppedWorkSE1(find(StoppedWorkSE1.
               WorkType == [2]),"GroupCount");
            if isempty(StoppedRequestsSE1T)
                StoppedRequestsSE1 = 0;
            else
                StoppedRequestsSE1 = StoppedRequestsSE1T{1,1};
            end
        %Project Tasks
            StoppedProjectTasksSE1T = StoppedWorkSE1(find(
               StoppedWorkSE1.WorkType == [3]),"GroupCount");
            if isempty(StoppedProjectTasksSE1T)
```

```
                StoppedProjectTasksSE1 = 0;
            else
                StoppedProjectTasksSE1 = StoppedProjectTasksSE1T{1,1};
            end
        %Maintenance Tasks
            StoppedMaintenanceTasksSE1T = StoppedWorkSE1(find(
               StoppedWorkSE1.WorkType == [4]),"GroupCount");
            if isempty(StoppedMaintenanceTasksSE1T)
                StoppedMaintenanceTasksSE1 = 0;
            else
                StoppedMaintenanceTasksSE1 =
                   StoppedMaintenanceTasksSE1T{1,1};
            end
        %Admin Tasks
            StoppedAdminTasksSE1T = StoppedWorkSE1(find(StoppedWorkSE1.
               WorkType == [5]),"GroupCount");
            if isempty(StoppedAdminTasksSE1T)
                StoppedAdminTasksSE1 = 0;
            else
                StoppedAdminTasksSE1 = StoppedAdminTasksSE1T{1,1};
            end
        end
        AllTicketsStoppedSE1 = StoppedIncidentsSE1 + StoppedRequestsSE1
           ;
        AllWorkTasksStoppedSE1 = StoppedProjectTasksSE1 +
           StoppedMaintenanceTasksSE1;
        AllWorkStoppedSE1 = AllTicketsStoppedSE1 +
           AllWorkTasksStoppedSE1;

```

```
    %Picked up work SE1, not including admin tasks (serviced, but not
    %necessarily complete
        WBASE1 = get(out.logsout,"IndivQueueSE1Out").Values.IsAdmin;
        if isempty (WBASE1.Time)
            PickedUpIncidentsSE1 = 0;
            PickedUpRequestsSE1 = 0;
            PickedUpProjectTasksSE1 = 0;
            PickedUpMaintTasksSE1 = 0;
        else
        WorkByAdminSE1 = timetable2table(timeseries2timetable(WBASE1));
        WorkByTypeSE1 = timetable2table(timeseries2timetable(get(out.
           logsout,"IndivQueueSE1Out").Values.WorkType));
        WorkByTypeSE1(:,1) = [];
        WorkByAdminTypeSE1 = renamevars(addvars(WorkByAdminSE1,
           WorkByTypeSE1.WorkType),["Var3"],["WorkType"]);
        WorkByTypeSE1NoAdmin = WorkByAdminTypeSE1(~(WorkByAdminTypeSE1.
           IsAdmin == 1),:);
        WorkByTypeSE1NoAdmin = groupsummary(WorkByTypeSE1NoAdmin,"
           WorkType");
        %Incidents
            PickedUpIncidentsSE1T = WorkByTypeSE1NoAdmin(find(
               WorkByTypeSE1NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsSE1T)
                PickedUpIncidentsSE1 = 0;
            else
                PickedUpIncidentsSE1 = PickedUpIncidentsSE1T{1,1};
            end
        %Requests
```

```
        PickedUpRequestsSE1T = WorkByTypeSE1NoAdmin(find(WorkByTypeSE1NoAdmin.WorkType == [2]),"GroupCount");
        if isempty(PickedUpRequestsSE1T)
            PickedUpRequestsSE1 = 0;
        else
            PickedUpRequestsSE1 = PickedUpRequestsSE1T{1,1};
        end
    %ProjectTasks
        PickedUpProjectTasksSE1T = WorkByTypeSE1NoAdmin(find(WorkByTypeSE1NoAdmin.WorkType == [3]),"GroupCount");
        if isempty(PickedUpProjectTasksSE1T)
            PickedUpProjectTasksSE1 = 0;
        else
            PickedUpProjectTasksSE1 = PickedUpProjectTasksSE1T{1,1};
        end
    %MaintTasks
        PickedUpMaintTasksSE1T = WorkByTypeSE1NoAdmin(find(WorkByTypeSE1NoAdmin.WorkType == [4]),"GroupCount");
        if isempty(PickedUpMaintTasksSE1T)
            PickedUpMaintTasksSE1 = 0;
        else
            PickedUpMaintTasksSE1 = PickedUpMaintTasksSE1T{1,1};
        end
    end
    AllTicketsPickedUpSE1 = PickedUpIncidentsSE1 + PickedUpRequestsSE1;
    AllTasksPickedUpSE1 = PickedUpProjectTasksSE1 + PickedUpMaintTasksSE1;
```

```
        AllWorkedPickedUpSE1 = AllTicketsPickedUpSE1 + AllTasksPickedUpSE1;

    %Stopped work SE2
        SWSE2TS = get(out.logsout,"CompSwitchSrEng2Stop").Values.WorkType;
        if isempty (SWSE2TS.Time)
            StoppedIncidentsSE2 = 0;
            StoppedRequestsSE2 = 0;
            StoppedProjectTasksSE2 = 0;
            StoppedMaintenanceTasksSE2 = 0;
            StoppedAdminTasksSE2 = 0;
        else
        StoppedWorkSE2 = groupsummary(timetable2table(timeseries2timetable(SWSE2TS)),"WorkType");
        %Incidents
            StoppedIncidentsSE2T = StoppedWorkSE2(find(StoppedWorkSE2.WorkType == [1]),"GroupCount");
            if isempty(StoppedIncidentsSE2T)
                StoppedIncidentsSE2 = 0;
            else
                StoppedIncidentsSE2 = StoppedIncidentsSE2T{1,1};
            end
        %Requests
            StoppedRequestsSE2T = StoppedWorkSE2(find(StoppedWorkSE2.WorkType == [2]),"GroupCount");
            if isempty(StoppedRequestsSE2T)
                StoppedRequestsSE2 = 0;
            else
```

```
                StoppedRequestsSE2 = StoppedRequestsSE2T{1,1};
            end
        %Project Tasks
            StoppedProjectTasksSE2T = StoppedWorkSE2(find(
               StoppedWorkSE2.WorkType == [3]),"GroupCount");
            if isempty(StoppedProjectTasksSE2T)
                StoppedProjectTasksSE2 = 0;
            else
                StoppedProjectTasksSE2 = StoppedProjectTasksSE2T{1,1};
            end
        %Maintenance Tasks
            StoppedMaintenanceTasksSE2T = StoppedWorkSE2(find(
               StoppedWorkSE2.WorkType == [4]),"GroupCount");
            if isempty(StoppedMaintenanceTasksSE2T)
                StoppedMaintenanceTasksSE2 = 0;
            else
                StoppedMaintenanceTasksSE2 =
                   StoppedMaintenanceTasksSE2T{1,1};
            end
        %Admin Tasks
            StoppedAdminTasksSE2T = StoppedWorkSE2(find(StoppedWorkSE2.
               WorkType == [5]),"GroupCount");
            if isempty(StoppedAdminTasksSE2T)
                StoppedAdminTasksSE2 = 0;
            else
                StoppedAdminTasksSE2 = StoppedAdminTasksSE2T{1,1};
            end
        end
```

```
        AllTicketsStoppedSE2 = StoppedIncidentsSE2 + StoppedRequestsSE2;
        AllWorkTasksStoppedSE2 = StoppedProjectTasksSE2 + StoppedMaintenanceTasksSE2;
        AllWorkStoppedSE2 = AllTicketsStoppedSE2 + AllWorkTasksStoppedSE2;
    %Picked up work SE2, not including admin tasks (serviced, but not
    %necessarily complete
        WBASE2 = get(out.logsout,"IndivQueueSE2Out").Values.IsAdmin;
        if isempty (WBASE2.Time)
            PickedUpIncidentsSE2 = 0;
            PickedUpRequestsSE2 = 0;
            PickedUpProjectTasksSE2 = 0;
            PickedUpMaintTasksSE2 = 0;
        else
        WorkByAdminSE2 = timetable2table(timeseries2timetable(WBASE2));
        WorkByTypeSE2 = timetable2table(timeseries2timetable(get(out.logsout,"IndivQueueSE2Out").Values.WorkType));
        WorkByTypeSE2(:,1) = [];
        WorkByAdminTypeSE2 = renamevars(addvars(WorkByAdminSE2, WorkByTypeSE2.WorkType),["Var3"],["WorkType"]);
        WorkByTypeSE2NoAdmin = WorkByAdminTypeSE2(~(WorkByAdminTypeSE2.IsAdmin == 1),:);
        WorkByTypeSE2NoAdmin = groupsummary(WorkByTypeSE2NoAdmin,"WorkType");
        %Incidents
            PickedUpIncidentsSE2T = WorkByTypeSE2NoAdmin(find(WorkByTypeSE2NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsSE2T)
```

```
                PickedUpIncidentsSE2 = 0;
            else
                PickedUpIncidentsSE2 = PickedUpIncidentsSE2T{1,1};
            end
        %Requests
            PickedUpRequestsSE2T = WorkByTypeSE2NoAdmin(find(
               WorkByTypeSE2NoAdmin.WorkType == [2]),"GroupCount");
            if isempty(PickedUpRequestsSE2T)
                PickedUpRequestsSE2 = 0;
            else
                PickedUpRequestsSE2 = PickedUpRequestsSE2T{1,1};
            end
        %ProjectTasks
            PickedUpProjectTasksSE2T = WorkByTypeSE2NoAdmin(find(
               WorkByTypeSE2NoAdmin.WorkType == [3]),"GroupCount");
            if isempty(PickedUpProjectTasksSE2T)
                PickedUpProjectTasksSE2 = 0;
            else
                PickedUpProjectTasksSE2 = PickedUpProjectTasksSE2T
                   {1,1};
            end
        %MaintTasks
            PickedUpMaintTasksSE2T = WorkByTypeSE2NoAdmin(find(
               WorkByTypeSE2NoAdmin.WorkType == [4]),"GroupCount");
            if isempty(PickedUpMaintTasksSE2T)
                PickedUpMaintTasksSE2 = 0;
            else
                PickedUpMaintTasksSE2 = PickedUpMaintTasksSE2T{1,1};
            end
```

```
        end
        AllTicketsPickedUpSE2 = PickedUpIncidentsSE2 + PickedUpRequestsSE2;
        AllTasksPickedUpSE2 = PickedUpProjectTasksSE2 + PickedUpMaintTasksSE2;
        AllWorkedPickedUpSE2 = AllTicketsPickedUpSE2 + AllTasksPickedUpSE2;

    %Stopped work SE3
        SWSE3TS = get(out.logsout,"CompSwitchSrEng3Stop").Values.WorkType;
        if isempty (SWSE3TS.Time)
            StoppedIncidentsSE3 = 0;
            StoppedRequestsSE3 = 0;
            StoppedProjectTasksSE3 = 0;
            StoppedMaintTasksSE3 = 0;
        else
        StoppedWorkSE3 = groupsummary(timetable2table(timeseries2timetable(SWSE3TS)),"WorkType");
        %Incidents
            StoppedIncidentsSE3T = StoppedWorkSE3(find(StoppedWorkSE3.WorkType == [1]),"GroupCount");
            if isempty(StoppedIncidentsSE3T)
                StoppedIncidentsSE3 = 0;
            else
                StoppedIncidentsSE3 = StoppedIncidentsSE3T{1,1};
            end
        %Requests
```

```
        StoppedRequestsSE3T = StoppedWorkSE3(find(StoppedWorkSE3.WorkType == [2]),"GroupCount");
        if isempty(StoppedRequestsSE3T)
            StoppedRequestsSE3 = 0;
        else
            StoppedRequestsSE3 = StoppedRequestsSE3T{1,1};
        end
    %Project Tasks
        StoppedProjectTasksSE3T = StoppedWorkSE3(find(StoppedWorkSE3.WorkType == [3]),"GroupCount");
        if isempty(StoppedProjectTasksSE3T)
            StoppedProjectTasksSE3 = 0;
        else
            StoppedProjectTasksSE3 = StoppedProjectTasksSE3T{1,1};
        end
     %MaintTasks
        StoppedMaintTasksSE3T = StoppedWorkSE3(find(StoppedWorkSE3.WorkType == [4]),"GroupCount");
        if isempty(StoppedMaintTasksSE3T)
            StoppedMaintTasksSE3 = 0;
        else
            StoppedMaintTasksSE3 = StoppedMaintTasksSE3T{1,1};
        end
    end
    AllTicketsStoppedSE3 = StoppedIncidentsSE3 + StoppedRequestsSE3;
    AllWorkTasksStoppedSE3 = StoppedProjectTasksSE3 + StoppedMaintTasksSE3;
```

```
        AllWorkStoppedSE3 = AllTicketsStoppedSE3 + AllWorkTasksStoppedSE3;
    %Picked up work SE3, not including admin tasks (serviced, but not
    %necessarily complete
        WBASE3 = get(out.logsout,"IndivQueueSE3Out").Values.IsAdmin
        if isempty (WBASE3.Time)
            PickedUpIncidentsSE3 = 0;
            PickedUpRequestsSE3 = 0;
            PickedUpProjectTasksSE3 = 0;
            PickedUpMaintTasksSE3 = 0;
        else
        WorkByAdminSE3 = timetable2table(timeseries2timetable(WBASE3));
        WorkByTypeSE3 = timetable2table(timeseries2timetable(get(out.logsout,"IndivQueueSE3Out").Values.WorkType));
        WorkByTypeSE3(:,1) = [];
        WorkByAdminTypeSE3 = renamevars(addvars(WorkByAdminSE3,WorkByTypeSE3.WorkType),["Var3"],["WorkType"]);
        WorkByTypeSE3NoAdmin = WorkByAdminTypeSE3(~(WorkByAdminTypeSE3.IsAdmin == 1),:);
        WorkByTypeSE3NoAdmin = groupsummary(WorkByTypeSE3NoAdmin,"WorkType");
        %Incidents
            PickedUpIncidentsSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [1]),"GroupCount");
            if isempty(PickedUpIncidentsSE3T)
                PickedUpIncidentsSE3 = 0;
            else
                PickedUpIncidentsSE3 = PickedUpIncidentsSE3T{1,1};
            end
```

```
    %Requests
        PickedUpRequestsSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [2]),"GroupCount");
        if isempty(PickedUpRequestsSE3T)
            PickedUpRequestsSE3 = 0;
        else
            PickedUpRequestsSE3 = PickedUpRequestsSE3T{1,1};
        end
    %ProjectTasks
        PickedUpProjectTasksSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [3]),"GroupCount");
        if isempty(PickedUpProjectTasksSE3T)
            PickedUpProjectTasksSE3 = 0;
        else
            PickedUpProjectTasksSE3 = PickedUpProjectTasksSE3T{1,1};
        end
    %MaintTasks
        PickedUpMaintTasksSE3T = WorkByTypeSE3NoAdmin(find(WorkByTypeSE3NoAdmin.WorkType == [4]),"GroupCount");
        if isempty(PickedUpMaintTasksSE3T)
            PickedUpMaintTasksSE3 = 0;
        else
            PickedUpMaintTasksSE3 = PickedUpMaintTasksSE3T{1,1};
        end
    end
    AllTicketsPickedUpSE3 = PickedUpIncidentsSE3 + PickedUpRequestsSE3;
```

```
        AllTasksPickedUpSE3 = PickedUpProjectTasksSE3 +
           PickedUpMaintTasksSE3;
        AllWorkedPickedUpSE3 = AllTicketsPickedUpSE3 +
           AllTasksPickedUpSE3;

    %Total stopped work - all engineers
    AllTicketsStopped = AllTicketsStoppedE1 + AllTicketsStoppedSE1 +
       AllTicketsStoppedSE2 + AllTicketsStoppedSE3;
    AllWorkTasksStopped = AllWorkTasksStoppedE1 +
       AllWorkTasksStoppedSE1 + AllWorkTasksStoppedSE2 +
       AllWorkTasksStoppedSE3;
    AllWorkStopped = AllTicketsStopped + AllWorkTasksStopped;

    %Total picked up work - all engineers (excluding admin tasks)
    AllTicketsPickedUp = AllTicketsPickedUpE1 + AllTicketsPickedUpSE1 +
        AllTicketsPickedUpSE2 + AllTicketsPickedUpSE3;
    AllTasksPickedUp = AllTasksPickedUpE1 + AllTasksPickedUpSE1 +
       AllTasksPickedUpSE2 + AllTasksPickedUpSE3;
    AllWorkPickedUp = AllTicketsPickedUp + AllTasksPickedUp;

    %Generated work
        %Incidents
            if isempty(get(out.logsout,"IncGen").Values.IsAdmin.Data)
                IncidentsGenerated = 0;
            else
                IncidentsGenerated = height(timetable2table(
                   timeseries2timetable(get(out.logsout,"IncGen").
                   Values.IsAdmin)));
            end
```

```
        %Incidents from Change
            if isempty(get(out.logsout,"IncidentsfromReworkGen").Values
               .IsAdmin.Data)
                IncFromChgGenerated = 0;
            else
                IncFromChgGenerated = height(timetable2table(
                   timeseries2timetable(get(out.logsout,"
                   IncidentsfromReworkGen").Values.IsAdmin)));
            end
        %Requests
            if isempty(get(out.logsout,"ReqGen").Values.IsAdmin.Data)
                RequestsGenerated = 0;
            else
                RequestsGenerated = height(timetable2table(
                   timeseries2timetable(get(out.logsout,"ReqGen").
                   Values.IsAdmin)));
            end
        %Project Tasks
            if isempty(get(out.logsout,"ProjTaskGen").Values.IsAdmin.
               Data)
                ProjectTasksGenerated = 0;
            else
                ProjectTasksGenerated = height(timetable2table(
                   timeseries2timetable(get(out.logsout,"ProjTaskGen").
                   Values.IsAdmin)));
            end
        %Maintenance Tasks
            if isempty(get(out.logsout,"MaintTaskGen").Values.IsAdmin.
               Data)
```

```
                MaintenanceTasksGenerated = 0;
            else
                MaintenanceTasksGenerated = height(timetable2table(timeseries2timetable(get(out.logsout,"MaintTaskGen").Values.IsAdmin)));
            end
        %Admin Tasks
            if isempty(get(out.logsout,"AdminTaskGen").Values.IsAdmin.Data)
                AdminTasksGenerated = 0;
            else
                AdminTasksGenerated = height(timetable2table(timeseries2timetable(get(out.logsout,"AdminTaskGen").Values.IsAdmin)));
            end
        AllTicketsGenerated = RequestsGenerated + IncidentsGenerated;
        AllWorkTasksGenerated = ProjectTasksGenerated + IncFromChgGenerated + MaintenanceTasksGenerated;
        AllWorkGenerated = AllTicketsGenerated + AllWorkTasksGenerated;
        PercentTasks = AllWorkTasksGenerated./AllWorkGenerated;
        PercentTickets = AllTicketsGenerated./AllWorkGenerated;

    %Completed work
        %Incidents
            IncidentsCompletedTS = get(out.logsout,"IncComp").Values.IsAdmin;
            if isempty (IncidentsCompletedTS.Time)
                IncidentsCompletedTS = timeseries(0, 0);
            end
```

```
            IncidentsCompleted = height(timetable2table(
                timeseries2timetable(IncidentsCompletedTS)));
        %Requests
            RequestsCompletedTS = get(out.logsout,"ReqComp").Values.
                IsAdmin;
            if isempty (RequestsCompletedTS.Time)
                 RequestsCompletedTS = timeseries(0, 0);
            end
            RequestsCompleted = height(timetable2table(
                timeseries2timetable(RequestsCompletedTS)));
        %Project Tasks
            ProjectTasksCompletedTS = get(out.logsout,"ProjTaskComp").
                Values.IsAdmin;
            if isempty (ProjectTasksCompletedTS.Time)
                 ProjectTasksCompletedTS = timeseries(0, 0);
            end
            ProjectTasksCompleted = height(timetable2table(
                timeseries2timetable(ProjectTasksCompletedTS)));
        %Maintenance Tasks
            MaintenanceTasksCompletedTS = get(out.logsout,"
                MaintTaskComp").Values.IsAdmin;
            if isempty (MaintenanceTasksCompletedTS.Time)
                 MaintenanceTasksCompletedTS = timeseries(0, 0);
            end
            MaintenanceTasksCompleted = height(timetable2table(
                timeseries2timetable(MaintenanceTasksCompletedTS)));
        %Admin Tasks
            AdminTasksCompletedTS = get(out.logsout,"AdminWorkComp").
                Values.IsAdmin;
```

```
            if isempty (AdminTasksCompletedTS.Time)
                AdminTasksCompletedTS = timeseries(0, 0);
            end
            AdminTasksCompleted = height(timetable2table(timeseries2timetable(AdminTasksCompletedTS)));
        AllTicketsCompleted = RequestsCompleted + IncidentsCompleted;
        AllWorkTasksCompleted = ProjectTasksCompleted + MaintenanceTasksCompleted;
        AllWorkCompleted = AllTicketsCompleted + AllWorkTasksCompleted;

    %Rates

    TicketStoppageRate = AllTicketsStopped./AllTicketsCompleted;
    TicketsStoppedPerDay = AllTicketsStopped./IterationLength;
    TicketsCompletedPerDay = AllTicketsCompleted./IterationLength;
    TaskStoppageRate = AllWorkTasksStopped./AllWorkTasksCompleted;
    TasksStoppedPerDay = AllWorkTasksStopped./IterationLength;
    TasksCompletedPerDay = AllWorkTasksCompleted./IterationLength;
    ReworkPerDay = IncFromChgGenerated./IterationLength;
    TicketPickupRate = AllTicketsPickedUp./AllTicketsGenerated;
    TicketsPickedUpPerDay = AllTicketsPickedUp./IterationLength;
    TaskPickupRate = AllTasksPickedUp./AllWorkTasksGenerated;
    TasksPickedUpPerDay = AllTasksPickedUp./IterationLength;
    WorkPickedUpPerDay = AllWorkPickedUp./IterationLength;
%due to low overall counts, if any occur the statistic will be heavily skewed upwards...
%   ReworkRate = ReworkPerDay./WorkPickedUpPerDay; %percentage
    TotalWorkSessions = AllTasksPickedUp + AllTicketsPickedUp;
    WorkSessionsPerDay = TotalWorkSessions./IterationLength;
```

```
    ErrorsPerDay = AllWorkCompleted*ErrorRate;
    BaseReworkRate = ErrorsPerDay*PercentTasks;
    BaseIncFromChange = ErrorsPerDay*PercentTickets;

    %Analyze DES Utilization data

    IncrTime = seconds(0:0.0325:5);
    E1UtilTS = out.E1Utilization;
    if isempty(E1UtilTS.Time)
        E1UtilTS = timeseries(0,0);
    end
    E1UtilTT = timeseries2timetable(E1UtilTS);

    SE1UtilTS = out.SE1Utilization;
    if isempty(SE1UtilTS.Time)
        SE1UtilTS = timeseries(0,0);
    end
    SE1UtilTT = timeseries2timetable(SE1UtilTS);

    SE2UtilTS = out.SE2Utilization;
    if isempty(SE2UtilTS.Time)
        SE2UtilTS = timeseries(0,0);
    end
    SE2UtilTT = timeseries2timetable(SE2UtilTS);
    SE2UtilTT = timeseries2timetable(out.SE2Utilization);

    SE3UtilTS = out.SE3Utilization;
    if isempty(SE3UtilTS.Time)
        SE3UtilTS = timeseries(0,0);
```

```
    end
    SE3UtilTT = timeseries2timetable(SE3UtilTS);
    SE3UtilTT = timeseries2timetable(out.SE3Utilization);

    TTUtilsync = synchronize(E1UtilTT, SE1UtilTT, SE2UtilTT, SE3UtilTT,
        'union', 'previous');
    TTUtilsync.AvgUtil = mean(TTUtilsync{:,{'Data_E1UtilTT','
       Data_SE1UtilTT','Data_SE2UtilTT','Data_SE3UtilTT'}},2);

    util_pred = TTUtilsync.AvgUtil(end);

    %Analyze DES Total Time data
    IncTimeTS = get(out.logsout,"TotalTimeInc").Values;
    if isempty(IncTimeTS.Time)
        IncTimeTS = timeseries(0,0);
    end
    IncTimeTT = timeseries2timetable(IncTimeTS);
    IncTimeTTColName = IncTimeTT.Properties.VariableNames{1};
    if isempty(IncTimeTS.Time)
        IncTimePDFLambdaSim = 0;
    else
        IncTimeMean = mean(IncTimeTT.(IncTimeTTColName));
        IncTimePDFLambdaSim = 1/IncTimeMean;
    end

    ReqTimeTS = get(out.logsout,"TotalTimeReq").Values;
    if isempty(ReqTimeTS.Time)
        ReqTimeTS = timeseries(0,0);
    end
```

```
    ReqTimeTT = timeseries2timetable(ReqTimeTS);
    ReqTimeTTColName = ReqTimeTT.Properties.VariableNames{1};
    if isempty(ReqTimeTS.Time)
        ReqTimePDFLambdaSim = 0;
    else
        ReqTimeMean = mean(ReqTimeTT.(ReqTimeTTColName));
        ReqTimePDFLambdaSim = 1/ReqTimeMean;
    end

    %Analyze queue depth
    E1QueueTS = out.E1QueueLength;
    if isempty(E1QueueTS.Time)
        E1QueueTS = timeseries(0,0);
    end
    E1QueueTT = timeseries2timetable(E1QueueTS);

    SE1QueueTS = out.SE1QueueLength;
    if isempty(SE1QueueTS.Time)
        SE1QueueTS = timeseries(0,0);
    end
    SE1QueueTT = timeseries2timetable(SE1QueueTS);

    SE2QueueTS = out.SE2QueueLength;
    if isempty(SE2QueueTS.Time)
        SE2QueueTS = timeseries(0,0);
    end
    SE2QueueTT = timeseries2timetable(SE2QueueTS);

    SE3QueueTS = out.SE3QueueLength;
```

```
    if isempty(SE3QueueTS.Time)
        SE3QueueTS = timeseries(0,0);
    end
    SE3QueueTT = timeseries2timetable(SE3QueueTS);

    TTQueuesync = synchronize(E1QueueTT, SE1QueueTT, SE2QueueTT,
       SE3QueueTT, 'union', 'previous');
    TTQueuesync.AvgQueue = mean(TTQueuesync{:,{'Data_E1QueueTT','
       Data_SE1QueueTT','Data_SE2QueueTT','Data_SE3QueueTT'}},2);

    queue_pred = TTQueuesync.AvgQueue(end);

end
```

Acronyms

AI Artificial Intelligence *on page(s):* 1

API Application Programming Interface *on page(s):* 7, 9, 26, 27, 32, 57, 59, 61–67, 72, 73, 82, 85–89

BDD Block Definition Diagram *on page(s):* 60, 72

CASE Computer-Aided Software Engineering *on page(s):* 91, 92

CCTA Central Computing and Telecommunications Agency *on page(s):* 14

CDM Conceptual Data Model *on page(s):* 66

CI/CD Continuous Integration and Continuous Deployment *on page(s):* 27, 28, 71, 95

CIO Chief Information Officer *on page(s):* 59

CMDB Configuration Management Database *on page(s):* 70, 74

COTS Commercial Off-The-Shelf *on page(s):* 1, 2, 7, 60, 95

DBSE Document-Based Systems Engineering *on page(s):* 25

DES Discrete Event Simulation *on page(s):* 10, 19–21, 37, 41, 42, 44, 47, 50, 95

DNS Domain Name Service *on page(s):* 59, 70

ETL Extract, Transform and Load *on page(s):* 1, 60

IaaS Infrastructure as a Service *on page(s):* 7, 26, 31

IaC Infrastructure as Code *on page(s):* 2, 10, 60

IBD Internal Block Diagram *on page(s):* 73

IEEE Institute for Electrical and Electronics Engineers *on page(s):* 90

INCOSE International Council on Systems Engineering *on page(s):* 22, 28, 96

IoT Internet of Things *on page(s):* 1, 7

IPAM IP Address Management *on page(s):* 59, 70, 87

ITIL IT Information Library *on page(s):* 2, 6, 7, 14, 17, 34

LeSS Large-Scale Scrum *on page(s):* 13

LV Logical Viewpoint *on page(s):* 56, 72

MBE Model-Based Engineering *on page(s):* 91

MBSAP Model-Based System Architecture Process *on page(s):* 31, 56, 59, 64, 82

MBSE Model-Based Systems Engineering *on page(s):* ii, 25, 28, 29, 31, 82, 90–92, 96

MDSE Model-Driven Software Engineering *on page(s):* 91

NIST National Institute of Standards and Technology *on page(s):* 6, 25

OGC Office of Government Commerce *on page(s):* 14

OMG Object Management Group *on page(s):* 28

OV Operational Viewpoint *on page(s):* 56, 59, 64, 72

PaaS Platform as a Service *on page(s):* 7, 31

PMI Project Management Institute *on page(s):* 14

PRINCE2 PRojects IN Controlled Environments *on page(s):* 14

R&D Research and Development *on page(s):* 11

RA Reference Architecture *on page(s):* 31

RPA Robotic Process Automation *on page(s):* 54

RUP Rational Unified Process *on page(s):* 12

SaaS Software as a Service *on page(s):* 7, 31, 33

SAFe Scaled Agile *on page(s):* 13

SD System Dynamics *on page(s):* 10, 19–21, 33, 41, 42, 44, 50, 95

SDDC Software-Defined Data Centers *on page(s):* 10

SDI Software Defined Infrastructure *on page(s):* ii, iii, v, 2–4, 6, 8–10, 32, 55–62, 64–67, 69, 71–73, 77, 82–84, 87–91, 94–96

SDLC Software Development Lifecycle *on page(s):* 12, 14

SDN Software-Defined Networking *on page(s):* 8, 10, 26

SoI System of Interest *on page(s):* 31, 57

SysML Systems Modeling Language *on page(s):* 28, 29, 56, 82, 90, 96

UML Unified Modeling Language *on page(s):* 12, 28, 90–92

VM Virtual Machine *on page(s):* 69–71

Bibliography

[1] E. Simmon, “Evaluation of cloud computing services based on NIST SP 800-145,” National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 500-322, Feb. 2018, doi: 10.6028/NIST.SP.500-322.

[2] R. Stolletz and S. Helber, “Performance analysis of an inbound call center with skills-based routing,” *OR Spectrum*, vol. 26, no. 3, pp. 331–352, Jul. 2004, doi: 10.1007/s00291-004-0161-y.

[3] M. Hematian, M. M. Seyyed Esfahani, I. Mahdavi, N. Mahdavi-Amiri, and J. Rezaeian, “A multiobjective integrated multiproject scheduling and multi-skilled workforce assignment model considering learning effect under uncertainty,” *Computational Intelligence*, vol. 36, no. 1, pp. 276–296, 2020, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12260. doi: 10.1111/coin.12260.

[4] J. A. Mitchell, “Basic Principles of Information Technology Organization in Health Care Institutions,” *Journal of the American Medical Informatics Association*, vol. 4, no. 2, pp. s31–s35, 1997, url: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC61488/.

[5] H. I. H. I. . M. S. Society (HIMSS), *The Cphims Review Guide*. Taylor & Francis Group, Dec. 2021, google-Books-ID: IHJatwEACAAJ.

[6] J. H. Payne, “Management of multiple simultaneous projects: a state-of-the-art review,” *International Journal of Project Management*, vol. 13, no. 3, pp. 163–168, Jun. 1995, doi: 10.1016/0263-7863(94)00019-9.

[7] E. F. Franco, K. Hirama, and M. M. Carvalho, “Applying system dynamics approach in software and information system projects: A mapping study,” *Information and Software Technology*, vol. 93, pp. 58–73, Jan. 2018, doi: 10.1016/j.infsof.2017.08.013.

[8] A. Kossiakoff, S. Seymour, D. Flanigan, and S. Biemer, *Systems Engineering: Principles and Practices, 3rd Edition*, 3rd ed., ser. Wiley series in systems engineering and management. John Wiley & Sons, Inc, 2020.

[9] F. S. Glaiel, A. Moulton, and S. E. Madnick, "_USAgile Project Dynamics: A System Dynamics Investigation of Agile Software Development Methods," Massachusetts Institute of Technology. Engineering Systems Division, Working Paper, Oct. 2014, accepted: 2016-06-06T23:51:11Z. url: https://dspace.mit.edu/handle/1721.1/103024.

[10] "Rational Unified Process," Mar. 2022, page Version ID: 1078006369. url: https://en.wikipedia.org/w/index.php?title=Rational_Unified_Process&oldid=1078006369.

[11] L. Cao, R. Balasubramaniam, and T. Abdel-Hamid, "Modeling dynamics in agile software development | ACM Transactions on Management Information Systems," url: https://dl-acm-org.ezproxy2.library.colostate.edu/doi/abs/10.1145/1877725.1877730.

[12] B. Boehm, "Get ready for agile methods, with care," *Computer*, vol. 35, no. 1, pp. 64–69, Jan. 2002, conference Name: Computer. doi: 10.1109/2.976920.

[13] J. A. Buzacott, "Queueing models of Kanban and MRP controlled production systems," *Engineering Costs and Production Economics*, vol. 17, no. 1, pp. 3–20, Aug. 1989, doi: 10.1016/0167-188X(89)90050-5.

[14] M. Di Mascolo, Y. Frein, and Y. Dallery, "Queueing Network Modeling And Analysis Of Kanban Systems," in *Proceedings of the Third International Conference on Computer Integrated Manufacturing,*, May 1992, pp. 202–211, doi: 10.1109/CIM.1992.639073.

[15] *ITIL 4 Direct, Plan and Improve*. Stationery Office, Jan. 2020, google-Books-ID: otx-aygEACAAJ.

[16] J. R. San Cristóbal, L. Carral, E. Diaz, J. A. Fraguela, and G. Iglesias, "Complexity and Project Management: A General Overview," *Complexity*, vol. 2018, p. e4891286, Oct. 2018, publisher: Hindawi. doi: 10.1155/2018/4891286.

[17] D. Baccarini, “The concept of project complexity—a review,” *International Journal of Project Management*, vol. 14, no. 4, pp. 201–204, Aug. 1996, doi: 10.1016/0263-7863(95)00093-3.

[18] J. M. Lyneis, K. G. Cooper, and S. A. Els, “Strategic management of complex projects: a case study using system dynamics,” *System Dynamics Review*, vol. 17, no. 3, pp. 237–260, 2001, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sdr.213. doi: 10.1002/sdr.213.

[19] A. G. Rodrigues and T. M. Williams, “System dynamics in software project management: towards the development of a formal integrated framework,” *European Journal of Information Systems*, vol. 6, no. 1, pp. 51–66, Mar. 1997, publisher: Taylor & Francis _eprint: https://doi.org/10.1057/palgrave.ejis.3000256. doi: 10.1057/palgrave.ejis.3000256.

[20] A. Rodrigues and J. Bowers, “The role of system dynamics in project management,” *International Journal of Project Management*, vol. 14, no. 4, pp. 213–220, Aug. 1996, doi: 10.1016/0263-7863(95)00075-5.

[21] T. Abdel-Hamid and S. Madnick, “Lessons learned from modeling the dynamics of software development | Communications of the ACM,” *Communications of the ACM*, 1989, url: https://dl.acm.org/doi/abs/10.1145/76380.76383.

[22] J. M. Lyneis and D. N. Ford, “System dynamics applied to project management: a survey, assessment, and directions for future research,” *System Dynamics Review*, vol. 23, no. 2-3, pp. 157–189, 2007, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sdr.377. doi: 10.1002/sdr.377.

[23] D. N. Ford, J. M. Lyneis, and T. R. B. Taylor, “Project Controls to Minimize Cost and Schedule Overruns: A Model, Research Agenda, and Initial Results,” *2007 International System Dynamics Conference*, 2007.

[24] T. Abdel-Hamid, “The dynamics of software project staffing: a system dynamics based simulation approach,” *IEEE Transactions on Software Engineering*, vol. 15, no. 2, pp. 109–119, Feb. 1989, conference Name: IEEE Transactions on Software Engineering. doi: 10.1109/32.21738.

[25] D. N. Ford and J. D. Sterman, “Dynamic modeling of product development processes,” *System Dynamics Review*, vol. 14, no. 1, pp. 31–68, 1998, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291099-1727%28199821%2914%3A1%3C31%3A%3AAID-SDR141%3E3.0.CO%3B2-5. doi: 10.1002/(SICI)1099-1727(199821)14:1<31::AID-SDR141>3.0.CO;2-5.

[26] R. E. C. Ordoñez, M. Vanhoucke, J. Coelho, R. Anholon, and O. Novaski, “A Study of the Critical Chain Project Management Method Applied to a Multiproject System,” *Project Management Journal*, vol. 50, no. 3, pp. 322–334, Jun. 2019, publisher: SAGE Publications Inc. doi: 10.1177/8756972819832203.

[27] A. Platje and H. Seidel, “Breakthrough in multiproject management: how to escape the vicious circle of planning and control,” *International Journal of Project Management*, vol. 11, no. 4, pp. 209–213, Nov. 1993, doi: 10.1016/0263-7863(93)90037-N.

[28] A. P. Van Der Merwe, “Multi-project management—organizational structure and control,” *International Journal of Project Management*, vol. 15, no. 4, pp. 223–233, Aug. 1997, doi: 10.1016/S0263-7863(96)00075-0.

[29] C. Kang and Y. S. Hong, “Evaluation of Acceleration Effect of Dynamic Sequencing of Design Process in a Multiproject Environment,” *Journal of Mechanical Design*, vol. 131, no. 2, Jan. 2009, doi: 10.1115/1.3066599.

[30] K. O. Jensen, P. E. Barnsley, J. Tortolero, and N. Baxter, “Dynamic modelling of service delivery,” *BT Technology Journal*, vol. 24, no. 1, pp. 48–59, Jan. 2006, doi: 10.1007/s10550-006-0020-2.

[31] G. Antoniol, A. Cimitile, G. Di Lucca, and M. Di Penta, “Assessing staffing needs for a software maintenance project through queuing simulation,” *IEEE Transactions on Software Engineering*, vol. 30, no. 1, pp. 43–58, Jan. 2004, conference Name: IEEE Transactions on Software Engineering. doi: 10.1109/TSE.2004.1265735.

[32] P. Patanakul and D. Milosevic, “A competency model for effectiveness in managing multiple projects,” *The Journal of High Technology Management Research*, vol. 18, no. 2, pp. 118–131, Jan. 2008, doi: 10.1016/j.hitech.2007.12.006.

[33] S. Fricke and A. Shenbar, “Managing multiple engineering projects in a manufacturing support environment,” *IEEE Transactions on Engineering Management*, vol. 47, no. 2, pp. 258–268, May 2000, conference Name: IEEE Transactions on Engineering Management. doi: 10.1109/17.846792.

[34] Y. Diao, A. Heching, D. Northcutt, and G. Stark, “Modeling a complex global service delivery system,” *Proceedings - Winter Simulation Conference*, pp. 690–702, Dec. 2011, doi: 10.1109/WSC.2011.6147797.

[35] H. Rahmandad and D. M. Weiss, “Dynamics of concurrent software development,” *System Dynamics Review*, vol. 25, no. 3, pp. 224–249, 2009, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sdr.425. doi: 10.1002/sdr.425.

[36] C. Bartolini, C. Stefanelli, and M. Tortonesi, “SYMIAN: Analysis and performance improvement of the IT incident management process,” *IEEE Transactions on Network and Service Management*, vol. 7, no. 3, pp. 132–144, Sep. 2010, conference Name: IEEE Transactions on Network and Service Management. doi: 10.1109/TNSM.2010.1009.I9P0321.

[37] G. Antoniol, G. Casazza, G. Di Lucca, M. Di Penta, and F. Rago, “A queue theory-based approach to staff software maintenance centers,” in *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, Nov. 2001, pp. 510–519, iSSN: 1063-6773. doi: 10.1109/ICSM.2001.972764.

[38] *ITIL Foundation, ITIL.* TSO, The Stationary Office, 2019, google-Books-ID: HmsY-wQEACAAJ.

[39] J. Voyer, A. Cahill, K. Laustsen, and B. Philbrick, “System Dynamics Modeling of an IT Major Incident Resolution Process,” 2015, url: https://proceedings.systemdynamics.org/2015/sessions-thread.html.

[40] J. Wiik and K.-P. Kossakowski, “Dynamics of Incident Response.” FiRST, Apr. 2017, url: https://www.first.org/resources/papers/2005.

[41] G. Fenner, A. Lima, N. de Souza, A. Moura, and R. Andrade, “A system dynamics model for managing service desk capacity,” in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 1424–1427, iSSN: 1573-0077. doi: 10.1109/INM.2015.7140506.

[42] Y. Li and K. Katircioglu, “Measuring and Applying Service Request Effort Data in Application Management Services,” in *2013 IEEE International Conference on Services Computing*, Jun. 2013, pp. 352–359, doi: 10.1109/SCC.2013.64.

[43] R. Oliva, “A dynamic theory of service delivery - implications for managing service quality,” Ph.D. dissertation, Sloan School of Management, MIT, Cambridge, MA, Jun. 1996, url: https://people.tamu.edu/~roliva/research/pdfs/thesis.pdf.

[44] X. Li, Z. Zhan, S. Guo, and L. Zhang, “It incident assign algorithm based on the difference between support groups,” in *2010 International Conference on Advanced Intelligence and Awarenss Internet (AIAI 2010)*, Oct. 2010, pp. 319–323, doi: 10.1049/cp.2010.0778.

[45] R. E. i. C. Cloutier, “The Guide to the Systems Engineering Body of Knowledge (SEBoK),” Hoboken, NJ, US, 2023, bKCASE is managed and maintained by the Stevens Institute of Technology Systems Engineering Research Center, the International Council on Systems Engineering, and the Institute of Electrical and Electronics Engineers Systems Council. url: www.sebokwiki.org.

[46] P. Anderson, “Complexity Theory and Organization Science on JSTOR,” *Organization Science*, vol. Vol. 10, No. 3, no. Special Issue: Application of Complexity Theory to Organization Science, pp. 216–232, 1999, url: https://www-jstor-org.ezproxy2.library.colostate.edu/stable/2640328?sid=primo.

[47] J. W. Forrester, *Principles of Systems*. Productivity Press, 1990, google-Books-ID: oMASAQAAMAAJ.

[48] C. Caulfield and S. Maj, “A case for systems thinking and system dynamics,” in *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, vol. 5, Oct. 2001, pp. 2793–2798 vol.5, iSSN: 1062-922X. doi: 10.1109/ICSMC.2001.971932.

[49] R. Jonkers and K. E. Shahroudi, “Connecting Systems Science, Thinking and Engineering to Systems Practice for Managing Complexity,” *Journal of the International Society for the Systems Sciences*, vol. 65, no. 1, 2021, number: 1. url: https://journals.isss.org/index.php/jisss/article/view/3875.

[50] T. K. Abdel-Hamid, “The dynamics of software development project management : an integrative system dynamics perspective,” Thesis, Massachusetts Institute of Technology, 1984, accepted: 2007-08-03T18:17:24Z. url: https://dspace.mit.edu/handle/1721.1/38235.

[51] F. Groundstroem and S. Juhola, “Using systems thinking and causal loop diagrams to identify cascading climate change impacts on bioenergy supply systems,” *Mitigation and Adaptation Strategies for Global Change*, vol. 26, no. 7, p. 29, Aug. 2021, doi: 10.1007/s11027-021-09967-0.

[52] J. B. Homer and G. B. Hirsch, “System Dynamics Modeling for Public Health: Background and Opportunities,” *American Journal of Public Health*, vol. 96, no. 3, pp. 452–458, Mar. 2006, publisher: American Public Health Association. doi: 10.2105/AJPH.2005.062059.

[53] B. K. Choi, D. Kang, and B. K. Choi, *Modeling and Simulation of Discrete Event Systems*. Somerset, UNITED STATES: John Wiley & Sons, Incorporated, 2013, url: http://ebookcentral.proquest.com/lib/csu/detail.action?docID=1402564.

[54] S. Brailsford and N. Hilton, "A Comparison of Discrete Event Simulation and System Dynamics for Modelling Healthcare Systems," in *Planning for the Future, Health, Service Quality and Emergency Accessibility : Proceedings from ORAHS 2000*. Glasgow Caledonian University, 2001.

[55] K. Chahal, T. Eldabi, and T. Young, "A conceptual framework for hybrid system dynamics and discrete event simulation for healthcare," *Journal of Enterprise Information Management*, vol. 26, no. 1/2, pp. 50–74, Jan. 2013, publisher: Emerald Group Publishing Limited. doi: 10.1108/17410391311289541.

[56] E. D. Gönül-Sezer and Z. Ocak, "Comparison of System Dynamics and Discrete Event Simulation Approaches," in *Simulation and Modeling Methodologies, Technologies and Applications*, ser. Advances in Intelligent Systems and Computing, M. S. Obaidat, J. Kacprzyk, T. Ören, and J. Filipe, Eds. Cham: Springer International Publishing, 2016, pp. 69–81, doi: 10.1007/978-3-319-31295-8_5.

[57] A. Greasley, *Simulating Business Processes for Descriptive, Predictive, and Prescriptive Analytics*. Walter de Gruyter GmbH & Co KG, Oct. 2019, google-Books-ID: VjrEDwAAQBAJ.

[58] J. Viana, S. C. Brailsford, V. Harindra, and P. R. Harper, "Combining discrete-event simulation and system dynamics in a healthcare setting: A composite model for Chlamydia infection," *European Journal of Operational Research*, vol. 237, no. 1, pp. 196–206, Aug. 2014, doi: 10.1016/j.ejor.2014.02.052.

[59] J. Morgan, S. Howick, and V. Belton, "Designs for the complementary use of System Dynamics and Discrete-Event Simulation," in *Proceedings of the 2011 Winter Simulation Con-*

ference (WSC), Dec. 2011, pp. 2710–2722, iSSN: 1558-4305. doi: 10.1109/WSC.2011.6147977.

[60] A. Borshchev, *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*. AnyLogic North America, 2013, google-Books-ID: 2c5FnwEACAAJ.

[61] E. J. Oughton, W. Usher, P. Tyler, and J. W. Hall, "Infrastructure as a Complex Adaptive System," *Complexity*, vol. 2018, p. e3427826, Nov. 2018, publisher: Hindawi. doi: 10.1155/2018/3427826.

[62] "Definition of IT Infrastructure - Gartner Information Technology Glossary," url: https://www.gartner.com/en/information-technology/glossary/it-infrastructure.

[63] O. Hanseth and K. Lyytinen, "Theorizing about the Design of Information Infrastructures: Design Kernel Theories and Principles," *All Sprouts Content*, vol. 4, no. 12, Apr. 2008, url: https://aisel.aisnet.org/sprouts_all/68.

[64] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998, doi: 10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D.

[65] University of Michigan, P. Edwards, G. Bowker, University of Pittsburgh, S. Jackson, University of Michigan, R. Williams, and University of Edinburgh, "Introduction: An Agenda for Infrastructure Studies," *Journal of the Association for Information Systems*, vol. 10, no. 5, pp. 364–374, May 2009, doi: 10.17705/1jais.00200.

[66] M. Grisot, O. Hanseth, and A. Thorseng, "Innovation Of, In, On Infrastructures: Articulating the Role of Architecture in Information Infrastructure Evolution," *Journal of the Association for Information Systems*, vol. 15, no. 4, Apr. 2014, doi: 10.17705/1jais.00357.

[67] D. Ribes and T. A. Finholt, "_USThe Long Now of Infrastructure: Articulating Tensions in Development," May 2009, accepted: 2013-03-25T17:50:36Z Publisher: Journal of the

Association for Information Systems (JAIS). url: https://repository.library.georgetown.edu/handle/10822/557392.

[68] M. Morgan, "Synergizing Model-Based Systems Engineering, Modularity, and Software Container Concepts to Mitigate Obsolescence," 2021.

[69] M. Farwick, C. Schweda, R. Breu, and I. Hanschke, "A situational method for semi-automated Enterprise Architecture Documentation: Software & Systems Modeling," *Software & Systems Modeling*, vol. 15, no. 2, pp. 397–426, May 2016, publisher: Springer Nature. doi: 10.1007/s10270-014-0407-3.

[70] "About the Zachman Framework," url: https://zachman-feac.com/zachman/about-the-zachman-framework.

[71] "TOGAF | www.opengroup.org," url: https://www.opengroup.org/togaf.

[72] "Predicts 2024: Generative AI Will Transform IT Infrastructure and Operations," url: https://www.gartner.com/en.

[73] S. Malgas, "Cloud Architect: How to build architectural diagrams of Google Cloud Platform(GCP)," Feb. 2018, url: https://medium.com/\spacefactor\@m{}sphe.malgas/cloud-architect-how-to-build-architectural-diagrams-of-google-cloud-platform-gcp-67ff7662f9ec.

[74] "SAFe 6.0 Framework," url: https://scaledagileframework.com/.

[75] "Featured Visio templates and diagrams - Microsoft Support," url: https://support.microsoft.com/en-us/office/featured-visio-templates-and-diagrams-27d4274b-5fc2-4f5c-8190-35ff1db34aa5.

[76] D. R. Call and D. R. Herber, "Applicability of the diffusion of innovation theory to accelerate model-based systems engineering adoption," *Systems Engineering*, vol. 25, no. 6, pp. 574–583, 2022, doi: 10.1002/sys.21638.

[77] S. Kotusev, “Different Approaches to Enterprise Architecture,” *Journal of Enterprise Architecture*, vol. 12, pp. 9–16, Feb. 2017.

[78] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, Dec. 2009, doi: 10.1145/1496091.1496100.

[79] “SysML FAQ: What is SysML?” url: https://sysml.org/sysml-faq//sysml-faq/what-is-sysml.html.

[80] M. Hause, J. Buitelaar, M. van de Ven, and E. Burgers, “The Use of MBSE in Infrastructure Projects – An MBSE Challenge Team Paper,” *INCOSE International Symposium*, vol. 25, no. 1, pp. 371–387, 2015, doi: 10.1002/j.2334-5837.2015.00069.x.

[81] A. Poller, “Exploring and managing the complexity of large infrastructure projects with network theory and model-based systems engineering—The example of radioactive waste disposal,” *Systems Engineering*, vol. 23, no. 4, pp. 443–459, 2020, doi: 10.1002/sys.21537.

[82] M. Nikolaidou, A. Tsadimas, and D. Anagnostopoulos, “Model-Based Enterprise Information System Design: A SysML-based approach.”

[83] A. Tsadimas, “Model-based enterprise information system architectural design with SysML,” in *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, May 2015, pp. 492–497, iSSN: 2151-1357. doi: 10.1109/RCIS.2015.7128911.

[84] S. Izukura, K. Yanoo, T. Osaki, H. Sakaki, D. Kimura, and J. Xiang, “Applying a Model-Based Approach to IT Systems Development Using SysML Extension,” in *Model Driven Engineering Languages and Systems*, J. Whittle, T. Clark, and T. Kühne, Eds. Berlin, Heidelberg: Springer, 2011, pp. 563–577, doi: 10.1007/978-3-642-24485-8_41.

[85] A. Tsadimas, M. Nikolaidou, and D. Anagnostopoulos, “Evaluating software architecture in a model-based approach for enterprise information system design,” in *Proceedings of the*

2010 ICSE Workshop on Sharing and Reusing Architectural Knowledge, ser. SHARK '10. New York, NY, USA: Association for Computing Machinery, May 2010, pp. 72–79, doi: 10.1145/1833335.1833346.

[86] "MBSE Initiative," url: https://www.incose.org/communities/working-groups-initiatives/mbse-initiative.

[87] R. Clouthier and I. Obiako, "Model-Based Systems Engineering Adoption Trends 2009-2018," url: https://sebokwiki.org/wiki/Model-Based_Systems_Engineering_Adoption_Trends_2009-2018.

[88] M. Chami and J.-M. Bruel, "A Survey on MBSE Adoption Challenges." Wiley Interscience Publications, 2018, pp. 1–16, url: https://oatao.univ-toulouse.fr/22637/.

[89] K. Henderson and A. Salado, "Value and benefits of model-based systems engineering (MBSE): Evidence from the literature," *Systems Engineering*, vol. 24, no. 1, pp. 51–66, 2021, doi: 10.1002/sys.21566.

[90] K. X. Campo, T. Teper, C. E. Eaton, A. M. Shipman, G. Bhatia, and B. Mesmer, "Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature," *Systems Engineering*, vol. 26, no. 1, pp. 104–129, 2023, _eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21644. doi: 10.1002/sys.21644.

[91] T. Huldt and I. Stenius, "State-of-practice survey of model-based systems engineering," *Systems Engineering*, vol. 22, no. 2, pp. 134–145, 2019, doi: 10.1002/sys.21466.

[92] "IEEE/ISO/IEC International Standard for Software, systems and enterprise–Architecture description," *ISO/IEC/IEEE 42010:2022(E)*, pp. 1–74, Nov. 2022, conference Name: ISO/IEC/IEEE 42010:2022(E). doi: 10.1109/IEEESTD.2022.9938446.

[93] J. M. Borky and T. H. Bradley, *Effective Model-Based Systems Engineering*. Springer, Sep. 2018.

[94] E. Soares Palma, E. Yumi Nakagawa, D. M. Barroso Paiva, and M. Istela Cagnin, “Evolving Reference Architecture Description: Guidelines based on ISO/IEC/IEEE 42010,” Sep. 2022, publication Title: arXiv e-prints ADS Bibcode: 2022arXiv220914714S. doi: 10.48550/arXiv.2209.14714.

[95] S. Gudenkauf, M. Josefiok, A. Göring, and O. Norkus, “A Reference Architecture for Cloud Service Offers,” in *2013 17th IEEE International Enterprise Distributed Object Computing Conference*, Sep. 2013, pp. 227–236, iSSN: 1541-7719. doi: 10.1109/EDOC.2013.33.

[96] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, “NIST Cloud Computing Reference Architecture.”

[97] “Service-Oriented Cloud Computing Infrastructure (SOCCI) Framework,” url: https://www.opengroup.org/soa/source-book/socci/index.htm.

[98] A. Krishnaswamy, “Cloud Reference Architecture,” 2012.

[99] M. Bartock, D. Dodson, M. Souppaya, D. Carroll, R. Masten, G. Scinta, P. Massis, H. Prafullchandra, J. Malnar, H. Singh, R. Ghandi, L. Storey, R. Yeluri, T. Shea, M. Dalton, R. Weber, K. Scarfone, A. Dukes, J. Haskins, C. Phoenix, and B. Swarts, “Trusted Cloud: Security Practice Guide for VMware Hybrid Cloud Infrastructure as a Service (IaaS) Environments,” National Institute of Standards and Technology, Tech. Rep., Apr. 2022, doi: 10.6028/NIST.SP.1800-19.

[100] R. Shue, “IBM Cloud Computing Reference Architecture,” 2009.

[101] R. D. Zota and I. Petre, “An Overview of the Most Important Reference Architectures for Cloud Computing,” *Informatica Economica*, vol. 18, pp. 26–39, Dec. 2014, doi: 10.12948/issn14531305/18.4.2014.03.

[102] M. Badger, T. Grance, R. Patt-Corner, and J. Voas, “Cloud Computing Synopsis and Recommendations,” National Institute of Standards and Technology, Tech. Rep. NIST Special Publication (SP) 800-146, May 2012, doi: 10.6028/NIST.SP.800-146.

[103] L. Youseff, M. Butrico, and D. Da Silva, "Toward a Unified Ontology of Cloud Computing," in *2008 Grid Computing Environments Workshop*, Nov. 2008, pp. 1–10, iSSN: 2152-1093. doi: 10.1109/GCE.2008.4738443.

[104] M. Torkashvan and H. Haghighi, "A service oriented framework for cloud computing," in *Proceedings of the 3rd International Conference on Information and Communication Systems*, ser. ICICS '12. New York, NY, USA: Association for Computing Machinery, Apr. 2012, pp. 1–5, doi: 10.1145/2222444.2222469.

[105] R. R. Rhinehart, *Engineering Optimization: Applications, Methods and Analysis*. John Wiley & Sons, Mar. 2018, google-Books-ID: 3FpTDwAAQBAJ.

[106] G. Kim, K. Behr, and G. Spafford, *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*. BizBook Lab, Oct. 2014, google-Books-ID: mqXomAEACAAJ.

[107] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice: Second Edition*. Morgan & Claypool Publishers, Mar. 2017.

[108] A. Rodrigues da Silva, "Model-driven engineering: A survey supported by the unified conceptual model," *Computer Languages, Systems & Structures*, vol. 43, pp. 139–155, Oct. 2015, doi: 10.1016/j.cl.2015.06.001.

[109] V. C. Gu, Q. Cao, and W. Duan, "Unified Modeling Language (UML) IT adoption — A holistic model of organizational capabilities perspective," *Decision Support Systems*, vol. 54, no. 1, pp. 257–269, Dec. 2012, doi: 10.1016/j.dss.2012.05.034.