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

Summer 2025

Doctoral Committee:

Advisor: Daniel R. Herber

Steven A. Conrad
Kamran Eftekhari Shahroudi
Erika E. Gallegos
Ravi Mangal

ABSTRACT

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

This dissertation 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 are 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. Results from model simulation 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 to support the automation of the use cases identified with the help of the hybrid process simulation. The reference architecture is defined in a product- and tool-agnostic manner, elaborating a subset of the architecture in support of a common IT process: that of provisioning server capacity.

Finally, the paper proposes a roadmap and business case framework for IT leaders to follow, using the insights provided by the reference architecture to estimate costs and those of the hybrid simulation to estimate benefits. Rather than wholesale adoption of automation and DevOps processes, the paper recommends an incremental approach to establishing an SDI and the associated processes, tools, and automation code as needed to enable the priority use cases. The paper ends by discussing key limitations of the existing research and recommendations for future work.

ACKNOWLEDGEMENTS

To be completed.

DEDICATION

To be completed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AI Artificial Intelligence *on page(s):* 1, 116

AI/ML Artificial Intelligence and Machine Learning *on page(s):* 119

API Application Programming Interface *on page(s):* 8, 10, 30, 31, 33, 63, 66, 68, 70, 72–75, 77, 80–82, 84, 86, 91, 93, 96, 101, 102, 104, 107, 109, 115

BDD Block Definition Diagram *on page(s):* 70, 84, 86

BPML Business Process Modeling Language *on page(s):* 75

CCTA Central Computing and Telecommunications Agency *on page(s):* 17

CDM Conceptual Data Model *on page(s):* 65, 77

CI/CD Continuous Integration and Continuous Deployment *on page(s):* 22, 83, 121

CIO Chief Information Officer *on page(s):* 70

CMDB Configuration Management Database *on page(s):* 81, 88

COTS Commercial Off-The-Shelf *on page(s):* 1, 2, 8, 70, 112, 115, 121

DBSE Document-Based Systems Engineering *on page(s):* 28

DES Discrete Event Simulation *on page(s):* 4, 6, 7, 12, 13, 23–25, 38, 39, 41, 44, 46–48, 50, 53, 59, 102, 112–114, 117–119, 121, 140

DNS Domain Name Service *on page(s):* 68, 81, 92

EDI Electronic Data Interchange *on page(s):* 35

ETL Extract, Transform and Load *on page(s):* 1, 35, 70

IaaS Infrastructure as a Service *on page(s):* 8, 30, 32

IaC Infrastructure as Code *on page(s):* 2, 13, 70

IBD Internal Block Diagram *on page(s):* 84, 86

INCOSE International Council on Systems Engineering *on page(s):* 26, 34, 122

IoT Internet of Things *on page(s):* 1, 9

IPAM IP Address Management *on page(s):* 68, 81, 107

ITIL IT Information Library *on page(s):* 2, 6, 9, 17, 20, 115

LDM Logical Data Model *on page(s):* 65, 77, 90, 91

LeSS Large-Scale Scrum *on page(s):* 16

LV Logical Viewpoint *on page(s):* 65, 84, 113

MBSAP Model-Based System Architecture Process *on page(s):* 5, 7, 11, 31, 32, 65, 68, 75, 77, 90, 96, 113

MBSE Model-Based Systems Engineering *on page(s):* ii, 5, 7, 28, 31, 33–35, 94, 96, 112, 113, 115, 122

NIST National Institute of Standards and Technology *on page(s):* 7, 29, 32

NPV Net Present Value *on page(s):* 104

OGC Office of Government Commerce *on page(s):* 17

OMG Object Management Group *on page(s):* 33

OS Operating System *on page(s):* 81, 82

OV Operational Viewpoint *on page(s):* 65, 68, 75, 84

PaaS Platform as a Service *on page(s):* 8, 32

PAM Privileged Access Management *on page(s):* 106

PDF Probability Density Function *on page(s):* 52

PMBOK Project Management Body of Knowledge *on page(s):* 16, 17

PMI Project Management Institute *on page(s):* 17

PRINCE2 PRojects IN Controlled Environments *on page(s):* 17

R&D Research and Development *on page(s):* 14

RA Reference Architecture *on page(s):* 5, 11, 31, 32, 115

REST Representational State Transfer *on page(s):* 93

RPA Robotic Process Automation *on page(s):* 63

RUP Rational Unified Process *on page(s):* 15

SaaS Software as a Service *on page(s):* 8, 32, 36

SAFe Scaled Agile *on page(s):* 16

SD System Dynamics *on page(s):* 4, 6, 12, 13, 23–25, 36, 38, 39, 47, 48, 50, 59, 103, 112–114, 117, 118, 120, 140

SDDC Software-Defined Data Centers *on page(s):* 13, 33

SDI Software Defined Infrastructure *on page(s):* ii, iii, ix, x, 2–13, 33, 64–68, 70, 72, 74–77, 79, 80, 82–84, 89, 90, 93, 95–97, 99–101, 107–110, 113, 115, 116, 119, 121, 122

SDLC Software Development Lifecycle *on page(s):* 15, 17

SDN Software-Defined Networking *on page(s):* 10, 13, 30, 112

SLA Service Level Agreement *on page(s):* 29

SoI System of Interest *on page(s):* 31, 66

SoS System of Systems *on page(s):* 112

SysML Systems Modeling Language *on page(s):* 33, 34, 65, 96, 112, 113, 115, 122

UML Unified Modeling Language *on page(s):* 15, 33

VM Virtual Machine *on page(s):* 80–83

Chapter 1

Introduction

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.

1.1 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 Artificial Intelligence (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 emergent 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 system-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 IT Information Library (ITIL) (especially 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 Infrastructure as Code (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 ITIL. The leverage of Software Defined Infrastructure (SDI) by an enterprise IT organization, in fact, requires the creation of a new system for infrastructure management and allows the 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 SDI 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 SDI. As the old story relates, the cobbler's children have no shoes.

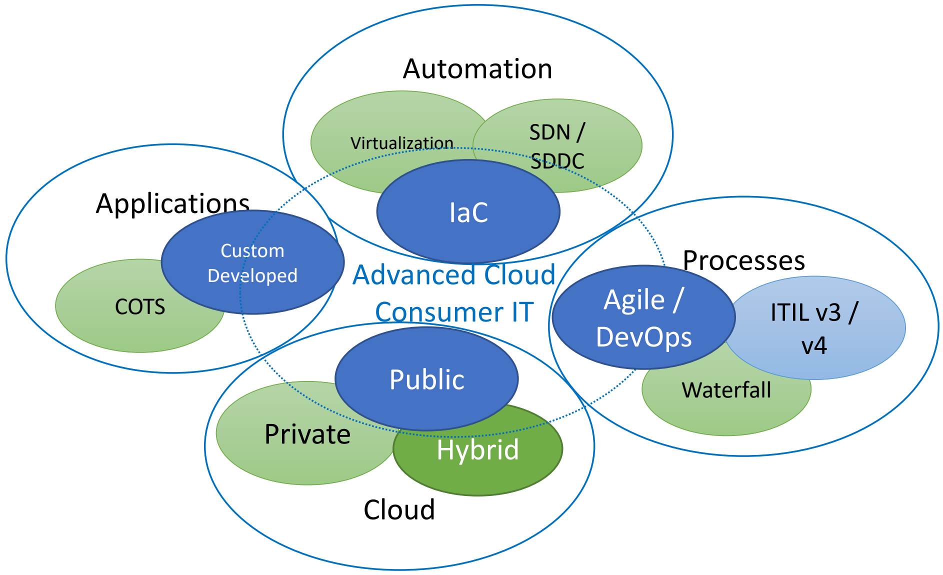

Figure 1.1: "New School" technologies and processes.

1.2 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 provider business units are actively investigating digital transformation to increase efficiency and quality, healthcare provider IT leaders are not clear 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.

The key research questions are as follows.

- Research Question 1: What are the basic components and capabilities of an on-premises SDI system?
- Research Question 2: Under what conditions should healthcare providers implement in-house SDI?
- Research Question 3: How should provider organizations proceed to implement on-premises SDI, if at all?

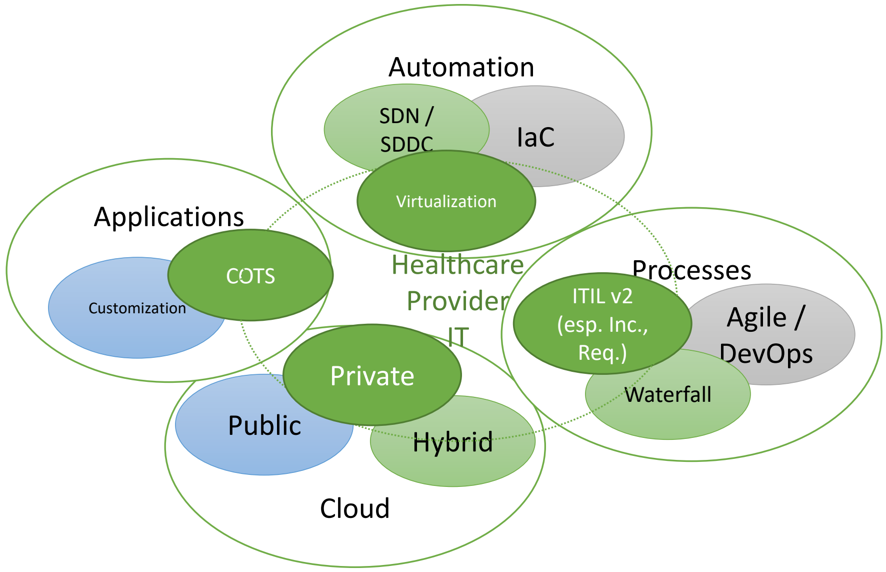

Figure 1.2: Healthcare provider IT technologies and processes.

1.3 Design of the Study

To conduct this research, I propose an explanatory sequential mixed-method approach to data collection and analysis.

To determine what should be automated, the research will start with the development of simulation models of the work performed by a large representative healthcare IT organization that will be used as the case study organization throughout the research. This is intended to identify potential areas of leverage of an SDI system, as well as provide predictions on the impact of automation on the outcomes of service delivery. These models are based on both queuing theory (in the form of a Discrete Event Simulation (DES) model) to address the fundamental management of work in the case study organization, and System Dynamics (SD) (in the form of a stock-and-flow model with feedback loops) to address other influencing factors. These models are built based on the guidance of prior academic research and then simulated using data culled from work management systems in the case study organization, augmented with estimates made by key stakeholders in the organization. This use of a hybrid SD and DES model in application to a skill-based organization responsible for project and operational work is novel, based on the literature review, and partially answers RQ2.

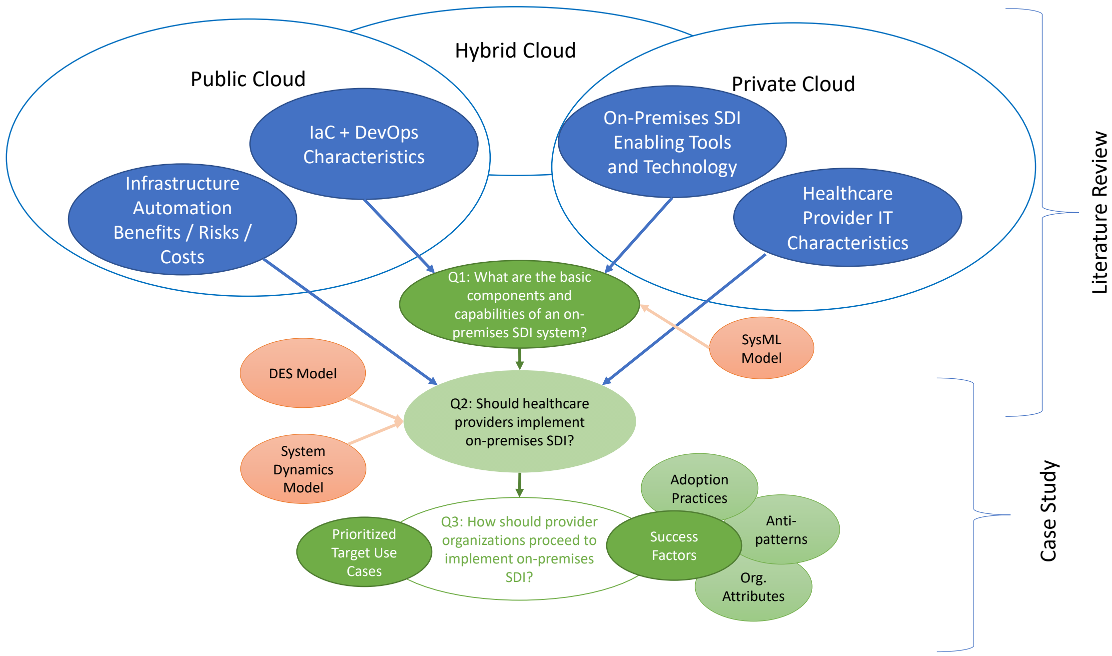

Figure 1.3: Overall flow of research.

This will be followed with the development of a Reference Architecture (RA) for an SDI solution to serve as the basis of process automation, leveraging the Model-Based System Architecture Process (MBSAP) to decompose the architecture and Model-Based Systems Engineering (MBSE) tooling to document it. The architecture will highlight a specific use case leveraged by the case study organization. The purpose of this research section is to define what needs to be built to enable SDI. This is also novel, based on a review of the available literature, which fully answers RQ1 and partially answers RQ3.

The results of these two sections of research are finally combined into an analysis of the value and cost of SDI, proposing a business case and implementation roadmap for use by IT decision makers to determine whether to implement SDI, and if so, how. This section partially answers RQ2 and RQ3.

The general flow of the research is shown in Figure 1.3.

1.4 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 indicate 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 – 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" in 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 semi-automated management system the most appropriate option, and if so, is focused on which use cases?

This research contributes to Systems Engineering practices by developing and validating a novel hybrid modeling approach SD and DES, to manage and optimize complex workflows in skill-based healthcare IT teams – and by extension to similarly-organized teams in other industries. This hybrid model further contributes to the Systems Engineering body of knowledge by explicitly addressing the challenges of balancing operational and project tasksin a single team under realistic constraints, such as limited skilled resources and dynamic priorities. The DES component of the hybrid model specifically provides quantitative comparisons of different operational strategies, thus enabling evidence-based decisions around resource allocation, prioritization, and process automation. This represents a practical advancement in systems engineering methodologies and applicability to IT infrastructure management. The explicit integration of stochastic and dynamic influences enables a deeper understanding of system behavior, supporting better-targeted automation investments. This research further contributes to Systems Engineering practice by proposing a vendor- and industry-agnostic reference architecture for SDI, utilizing the MBSAP and MBSE. This reference architecture facilitates standardization and integration across diverse IT infrastructure components, promoting reusable solutions and more efficient infrastructure management.

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 de-provision infrastructure capacity significantly enhances the first and fourth characteristics above and are critical to the development of agility and scalability common in DevOps environments.

1.5 Assumptions and Limitations

This study will focus on the applicability of SDI technologies and practices to large, for-profit healthcare providers in the United States, which are generally assumed to share the characteristics below. These characteristics explain in part the industry's preference for on-premises, COTS applications.

- Relatively low margins with increasing erosion due to evolving industry dynamics driving high rates of mergers, acquisitions, and divestitures, as well as increased ownership of provider organizations by private equity firms.
  - Significant commitment of IT staff to integrate after merger and acquisition activity, which drains available resources from other activities.
  - Increased likelihood of non-standard applications and infrastructure following this activity, often persisting until the obsolescence of critical systems forces the budget of capital to refresh those systems and enable standardization.
- Earnings-driven incentive systems that 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, by patching the infrastructure and updating the applications).
  - 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 depth and breadth of IT personnel generally – and especially of software development skills.

    - 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 the 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.

These characteristics are not unique to the healthcare provider industry and could, in fact, be applied to many organizations or industries where margin pressure and growth through acquisition are common. The conclusions drawn from this research may have a broader applicability to organizations or industry segments with similar characteristics, including finance and telecommunications.

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 the 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 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 SDI than others.

This study assumes that for a variety of reasons, 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 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.

1.6 Organization of the Remainder of the Dissertation

Chapter 2 consists of a qualitative review of the literature of the related areas that underlie SDI and the tools used throughout the investigation. Following this, Chapter 3 uses simulation models to explore whether and in what areas SDI is most suited to healthcare provider organizations, highlighting a specific use case within the case study organization. With the motivation for and targets of SDI established, Chapter 4 develops the architectural framework for an on-premises

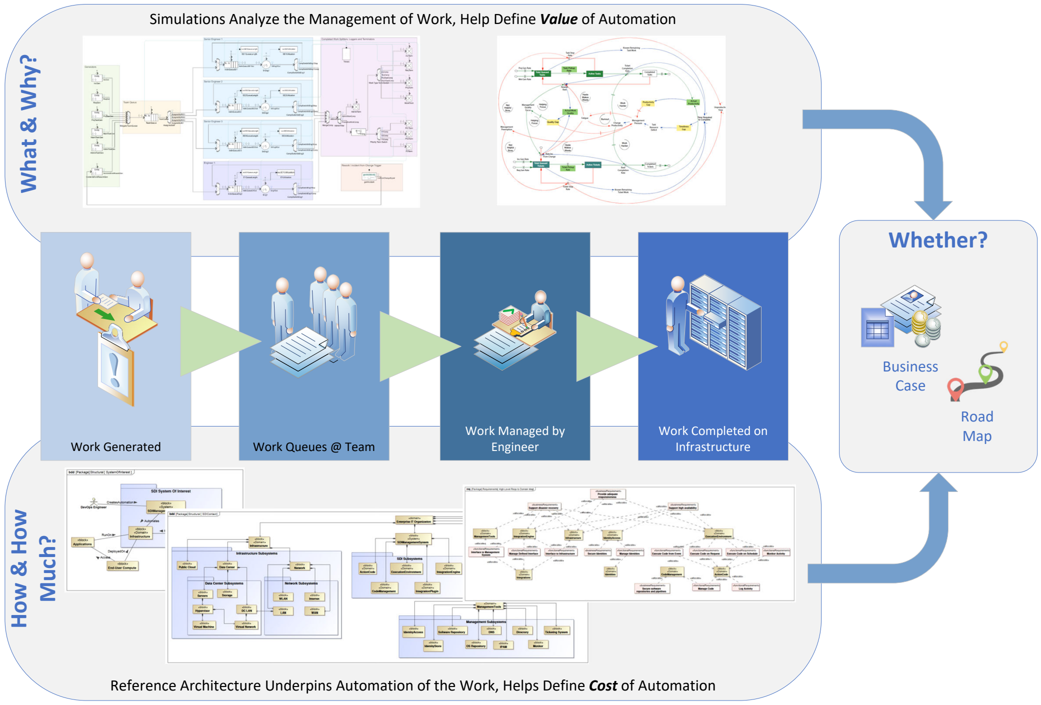

Figure 1.4: Threads between core areas of this dissertation.

SDI using the MBSAP. The results of these two sections are combined in Chapter 5 to provide a business case and implementation roadmap to guide IT decision makers and answer the research questions. Finally, Chapter 6 offers a summary and implications of this research and suggests areas for future research. Figure 1.4 summarizes how the simulation models, the RA, the business case, and the implementation roadmap demonstrated in this study work together to allow IT leaders to determine whether to implement SDI.

Chapter 2

Literature Review[1]

2.1 Introduction and Organization

This section outlines previous academic research relevant to this study, broken down into several sections.

- Relevant work management processes for unscheduled (operational tickets) and scheduled (project tasks) work.
- Modeling and simulation techniques for both forms of work.
- The characteristics of the IT infrastructure to be automated to enable better work management.
- Automation techniques and examples of implementation.
- Architectural descriptions and reference architectures, with emphasis on data center and public cloud environments

To conduct the literature review, I used an iterative mixed-method "citation chaining" approach (also known as "snowball sampling"). The process began with targeted searching in Google Scholar, Connected Papers, and the university library to identify key articles. I then performed citation searches both backward (checking references cited by those papers) and forward (looking for newer works that cite those papers, primarily using Connected Papers) to systematically expand the set of relevant literature. This approach uncovered 807 relevant papers, presentations, websites, blogs, and books. Although this is not a strictly systematic review, it provides a robust and iterative process to uncover additional studies and build a comprehensive body of research [2, 3].

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 pub-

[1]Note that a subset of the content in this chapter has been submitted for publication in the journal IEEE *Access*, and separate sections have been included in the journal IEEE *IT Professional* publication.

lic 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.

Prior research in the next two sections informs how best to formally discuss the management of work within a team in the case study organization, with the characteristics of being skill-based and responsible for both unscheduled (operational) and scheduled (project) work. This background is used to develop the simulation models explored in Chapter 3. The research outlined in the final three sections of this chapter informs the development of the SDI reference architecture in Chapter 4. In general, while the body of previous research addresses aspects of the problems addressed in this study and provides a great deal of guidance about the tools and techniques useful to do so, there is no single source that directly addresses the research questions outlined in Chapter 1. Also note that while the references in this chapter apply to the case study organization specifically and the healthcare provider industry more widely, they are not in any way specific to this industry and have broad applicability to organizations in many industries.

2.2 Work Management Processes

There is significant research in the modeling of project and operational work management systems and processes, including the use of DES and SD in understanding complex processes and predicting the benefits of their improvement. 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 with 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 for 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 [4] it is common across IT and in other functions such as R&D [5]. 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 [6] and the CPHIMS Review Guide [7] 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 [8]. Franco et al. [9] 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* [10]. 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 [11]. 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) [12], which was developed by Rational Software as a productized software development process and is highly 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 [13]. 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 system architecture planning [14]. 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 [15,16]. 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 Project Management

Body of Knowledge (PMBOK) 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. IT Information Library (ITIL) 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* [17] 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 [18, 19]. There is a rich body of research on the applicability of system dynamics as applied to the project management of single projects [20–23]. Lyneis and Ford, in particular, developed this model depicting the management of scheduled work through several iterations [24, 25]. 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 [26]. 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 [27]. 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 [22].

Ordonez et al. [28] 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 [29] emphasize the complexity of balancing costs, resource allocations, and completion times in these scenarios, while Van Der Merwe [30] 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 [8]. These characteristics are seen in the case study organization in Chapter 3.

Kang and Hong [31] 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 [32]. 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 [33].

Patanakul and Milosevic [34] 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 [35]. Diao and Hecheng acknowledge similar management overhead in the context of coordinating operational tickets between teams [36]. Platje and Seidel [29] 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 [37] 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. [32] 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 the project and the 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 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 [38] and treated by other researchers [39]. 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 queuing 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” [40].

Voyer et al. [41] 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 [42] 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. [43], and treatment of these issues resulting in ticket re-routing / reassignment is addressed by Li et al. [44].

Oliva developed a request management model that can be used as a basis for the second major type of unscheduled work [45]. 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. [46] and can also be fruitfully extended to project tasks.

2.2.4 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) requires cloud technologies that deliver capacity that is programmable, elastic, and on demand. These technologies are then heavily automated to accomplish common tasks, such as the promotion of changes between environments, or the mass-recreation of development or testing environments. These could be provided through any variant of cloud services deployments – public, private, or hybrid [47, 48]. 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).

2.3 Modeling and Simulation

2.3.1 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" [49]. Organizations are well researched as complex adaptive systems [50], 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 [51]. Systems thinking is a collection of methodologies that allow for consideration of the "whole" system, including its constituent parts and interactions [52]. 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 [53–56]. 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.

2.3.2 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) [57].

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 [58]. Another element that sets DES apart from SD is that it is inherently stochastic in assigning key elements of the model [59]. 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.

2.3.3 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 [59, 60]. 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 [61]. The conceptual "metamodel" for the integration of the two models closely follows the approach discussed by Viana et al. [62], 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. [59] 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 [59] 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. [63] 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 [64] 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 3.5.

2.4 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 [65], 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) [66, 67]. 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 [68].

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 [67, 69–71].

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 [67]. 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 [72].

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) [70].

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) [73]. 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 [74] and TOGAF [75] explicitly or implicitly rely on the production of document artifacts and viewpoints. These information repositories are not integrated, and therefore are 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 [76].

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 [77]) 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 [78]) 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.
- Visio diagrams and those from other drawing-only programs are ubiquitous, but these are point-in-time documents [79]. 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 [80] elaborate the practical differences between Document-Based Systems Engineering (DBSE) and MBSE, summarized clearly in Figure 2.1. Kotusev [81] 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.

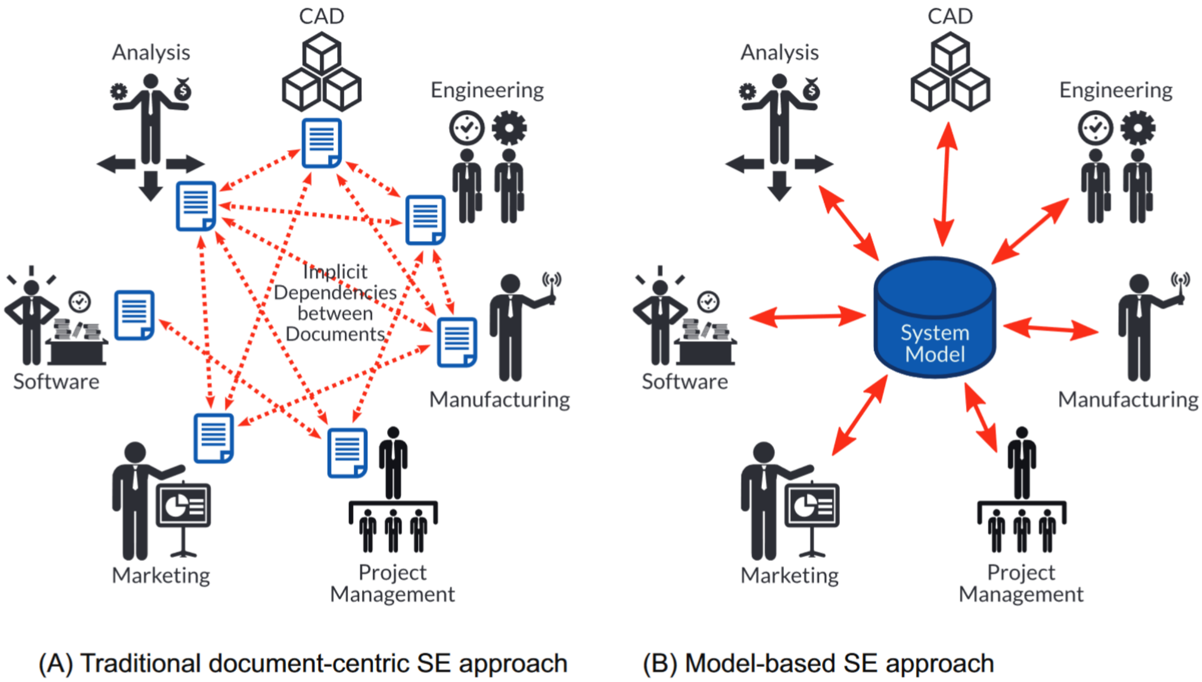

Figure 2.1: Document-based or traditional approach to SE compared to MBSE.

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" [82].

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.

Cloud Adoption

There has been research done in this space that covers some of the criteria that decision makers should use to assess whether cloud technologies are appropriate and the most appropriate deployment model (public, private or hybrid) [83–86]. 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.

2.5 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 [87, 88]. 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.

Infrastructure Automation and Infrastructure-as-Code

To date, research in this space has focused on 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 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 [89–91]. In addition, infrastructure automation is shown to rely on a high level of system standardization, based on the common analogy that systems (especially servers) are managed as "cattle" (larger numbers but essentially indistinguishable) as opposed to "pets" (individually unique) [92].

2.6 Architecture Descriptions and Reference Architectures

Architecture Description

ISO/IEC/IEEE 42010:2022(E) 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, its behavior and structure, and principles governing its design, use, operation, and evolution [93].

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) [94] applies the methods of MBSE to successively elaborate a system architecture and will be followed in Section 4.

Reference Architecture

Borky and Bradley define Reference Architecture (RA) 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 [95]. 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 [96, 97]. 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 [98].

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 (e.g., VMware, Oracle, Cisco, IBM, etc.) [99, 100]. All are designed to eliminate barriers to purchase and help IT properly implement the supporting infrastructure, and in some cases point to 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 a 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, composed of an Event Control Agent and a Service Execution Agent, to enable the programmatic response to events that could occur in a cloud environment [104]. Fung discusses this in the context of an on-premises Software-Defined Data Centers (SDDC) as the "Infrastructure Orchestration, Automation & Management" layer. 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.

2.6.1 A Note on Platforms

The concept of an IT "platform" occurs frequently in the literature, especially in the context of cloud services and reference architectures. Unfortunately, the term is used imprecisely, referring to different concepts in different contexts [105, 106]. For the purposes of this research, I will use the technical definition by Zeamari and Laurier [107]: "an extensible digital core (hardware and software) that provides core functionality shared by interoperating modules and interfaces".

2.6.2 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" [108]. 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 [109] and evolutionary systems-of-systems [110]. In addition, there are some academic papers that outline the use of SysML for the purpose of designing individual enterprise IT systems [111–114]. 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" [115]. 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 [116]. 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 [117]. Note that MBSE has been successfully adopted in traditional waterfall and Agile design environments.

SysML and MBSE are complex in themselves, so the cost and effort to adopt and maintain them is only justified when perceived value outweighs barriers to adoption [80]. Henderson and Salado reviewed the existing literature on the benefits of MBSE in 2021 [118] and found only two papers that claimed robustly *measured* benefits and a larger number citing *observed* benefits, while most benefits claims have no or highly subjective evidence. A more recent analysis by Campo et al. [119] supports this and further discusses the barriers claimed — increased cost, time, effort, and complexity during adoption — noting that these are better justified than benefits. According to a survey conducted in 2018, the benefits are realized most often during the early stages of the design process [120]. Broadly summarized, the *measured* and *observed* benefits are improved quality (specifically in error reduction and design consistency) and efficiency of the engineering process (especially traceability and collaboration). Also, benefits are generally realized post-adoption, which front-loads costs and back-loads benefits. This combination implies that MBSE can benefit long-term architecture more than short-lived initiatives, although all can eventually benefit once widely adopted.

The combination of a complex system-of-systems with a high degree of cross-departmental and interdependent involvement makes the 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 Electronic Data Interchange (EDI) or HL7), as well as bulk data transfers (Extract, Transform and Load (ETL)), replication, and aggregation to support 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 design, upgrade, migration, and recovery of 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.

Chapter 3

Simulation Modeling of the Work Environment[2]

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

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.

As stated in Section 1.5, 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.
- Earnings-driven incentive systems that discourage operational costs in favor of capital investments.

[2]Note that a subset of the content in this chapter has been published in the proceedings of the 2024 IEEE International Systems Conference

- High and increasing operational and clinical dependence on IT system and data performance and availability.

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 conflicting needs of IT and the constraints on it have fueled the need to explore options to automate wherever possible, either to secure immediate cost savings or to attenuate cost growth. Of course, the ability to provide higher levels of service, in terms of availability and performance, while doing so is always a benefit.

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 Developing the Models

As briefly mentioned in Chapter 1, the process for developing the hybrid model is as follows.

- Define the SD model for a single team in phases following the recommendations in Chapter 3, starting with the base stock-and-flow diagram and expanding to accommodate various causal loops.
- Add data inputs, equations, and constraints to determine the dynamics of the model and refine until the simulation functions independently with a reasonable output.
- Define the base DES model for a single team, again based on the recommendations in Chapter 3, focused on correct wiring of generators, queues, and terminators, and then elaborated the model by adding the impact of work stoppages and error generation.
- Add data inputs and equations to drive the dynamics of the model and refine as above. Add monitors as needed to extract the output from the model.
- Define the integration between the two models, in terms of both process and data elements. Align inputs and outputs in each model to support repeating iterations.
- Generate the code to manage iterations and data consolidation and analysis, and refine all elements until the combined / hybrid model can iterate multiple times while remaining in control and producing reasonable results.
- Generate the code to validate the results of the data, including regression and sensitivity analysis.

Note that the original intent was to generate a multi-team model in addition to the single-team model; however, due to the time available and increased complexity, this was deferred to future research.

3.3 Modeling the Work Environment of a Single Team

The methodology for modeling work management in the case study organization begins with a single team for simplicity, and specifically with the SD model in order to fully characterize the high-level flow of work and the influencing factors on it. The DES model is then defined in order to capture the detail of how the team leader and the staff resources approach their work. Following this, the models are simulated individually and refined as necessary to accommodate the available data from the organization's work management tools; these data are augmented with estimates from the organization's stakeholders where necessary. Finally, the simulations are coupled to iterate between the queuing and dynamic influences of the system. Although the expansion of the models to accommodate multiple teams was initially envisioned, the level of complexity rises significantly as teams are added and new dynamics arise; this is left to future research.

3.3.1 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.

3.3.2 Full Single-Team Model

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

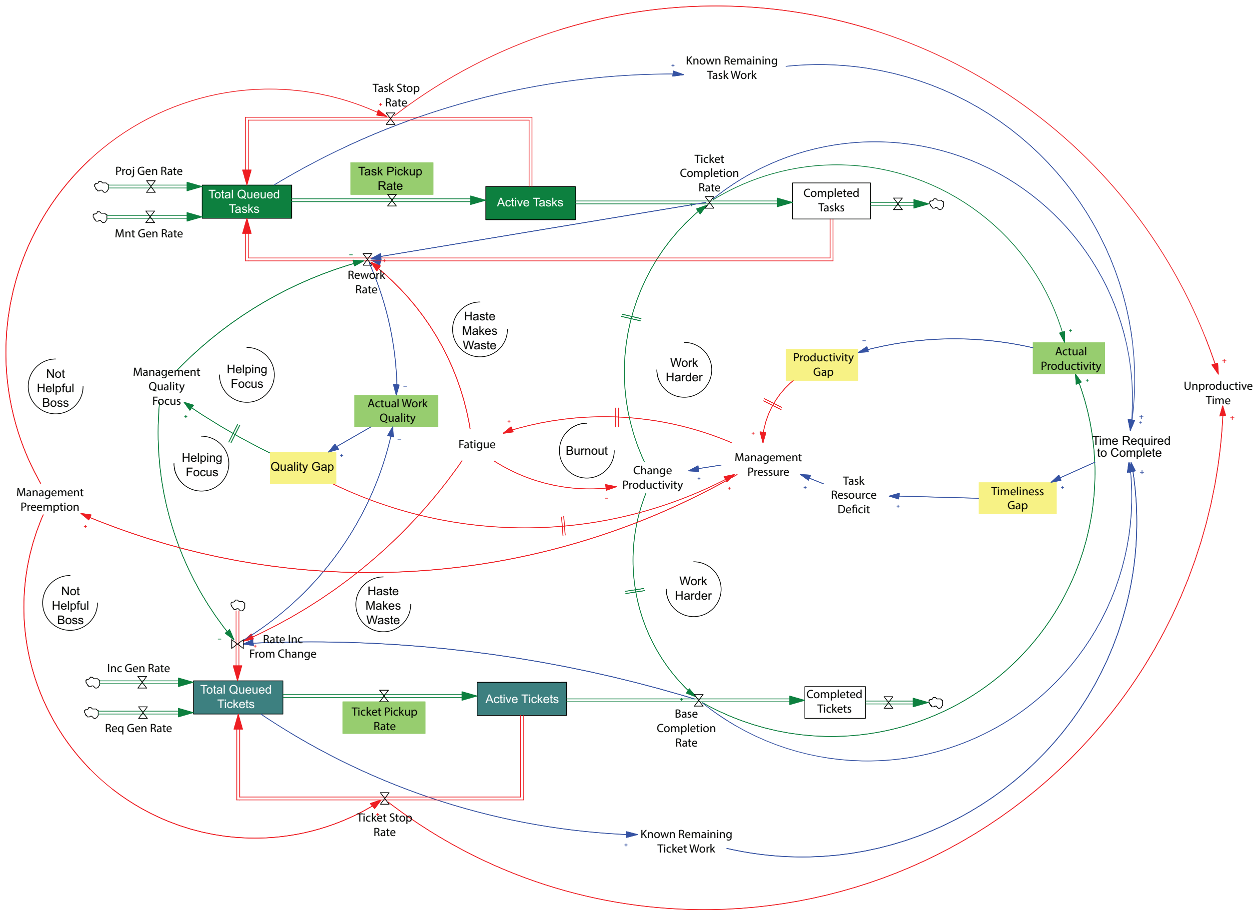

Figure 3.1: Vensim system dynamics model of a single team reflecting both project and operational workflows.

- 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.

3.3.3 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 [61]. 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 interarrival, 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 dis-

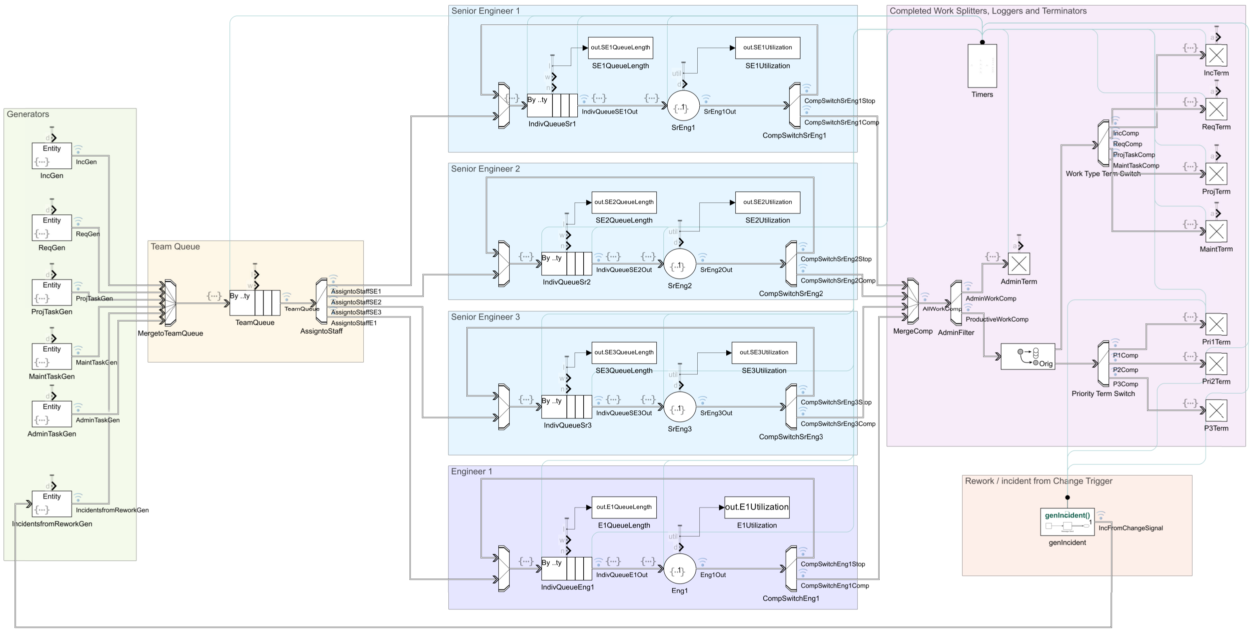

Figure 3.2: Simevents queuing model depicting entity generators for each work type feeding four engineers with individual queues.

cussion in the literature distinguishing the interarrival 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 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. Some sources argue that the interarrival time of project tasks cannot be treated essentially as Poisson, unlike operational tickets. This is because the identification and assignment of project tasks tend to be “bursty” – tasks are identified in large groups during planning phases based on organizational project management processes. Note that “identification” and “assignment” dates are distinct from the required “due date”. That said, this would tend to invalidate the “memoryless” requirement for Poisson arrival, arguing for another distribution, such as Weibull. This becomes relevant in the modeling of project task completion and in the overall validity of functional managers managing project tasks similarly to operational tickets.

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 service team is perceived as essentially random within many teams in the organization as long as the planning cycles for the different projects are not synchronized, due to the effect of the Palm-Khintchine theorem [121].
- 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.

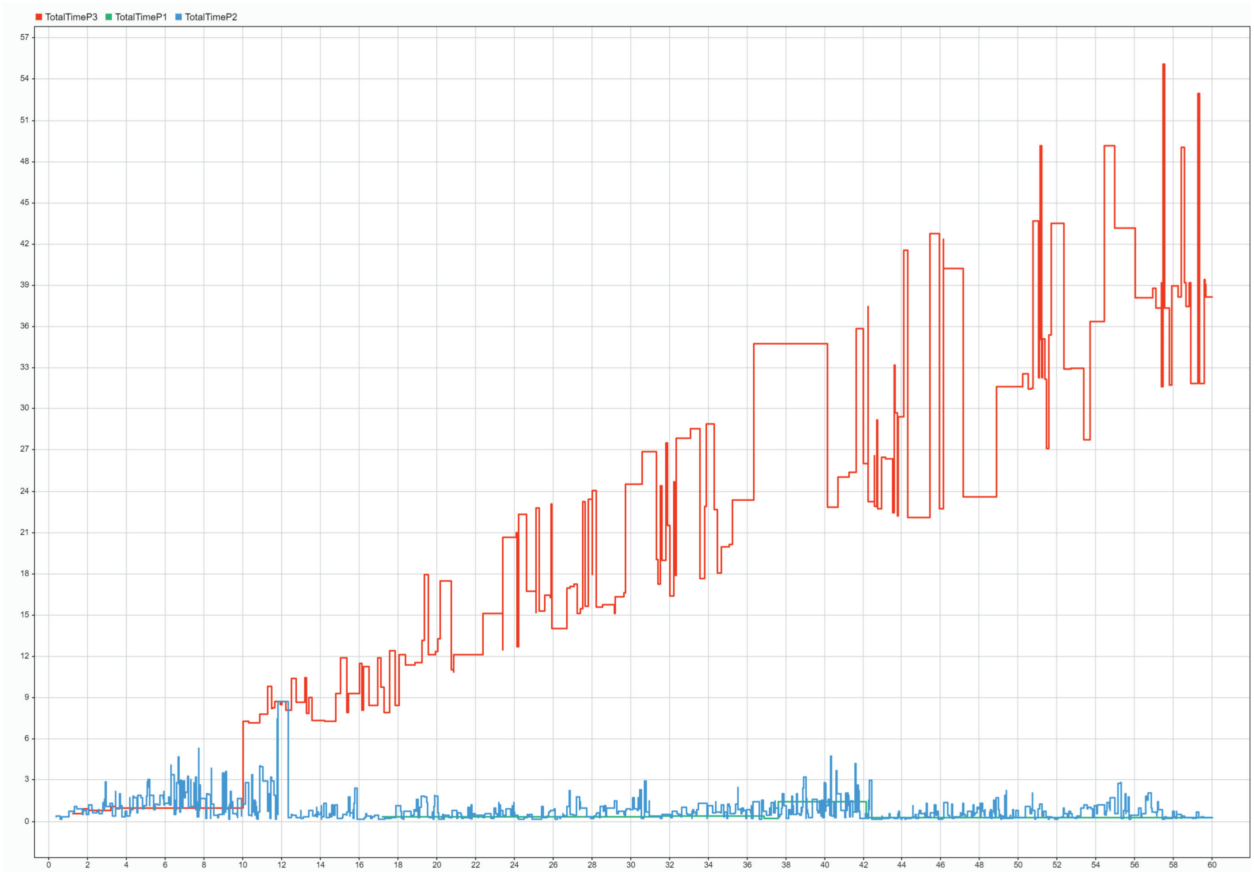

Figure 3.3: SimEvents Results using initial estimates for independent variables.

Running the DES model independently for 60 working days (roughly 12 weeks with 5 day weeks), returns the results shown in Figure 3.3. Note that high-priority work items (in green) are addressed very quickly, medium-priority items (in blue) are usually addressed within 3 days (and in many cases the day received), but that low-priority items (in red) begin to queue at the 10-day mark, and by the end of the simulation take many weeks to complete.

3.3.4 Interaction Points Between Models

As recommended by [61], 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 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.

3.4 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 error generation and stop rates has a strong impact on subsequent long iterations of the SimEvents model [122]. 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 Figure 3.4 and Figure 3.5 – the graphs show the increase in days to complete work over time between the baseline and the next iteration, with the simulation timeline on the horizontal axis (in days) and the difference in completion times on the vertical axis (also in days).

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.

3.5 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, and 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

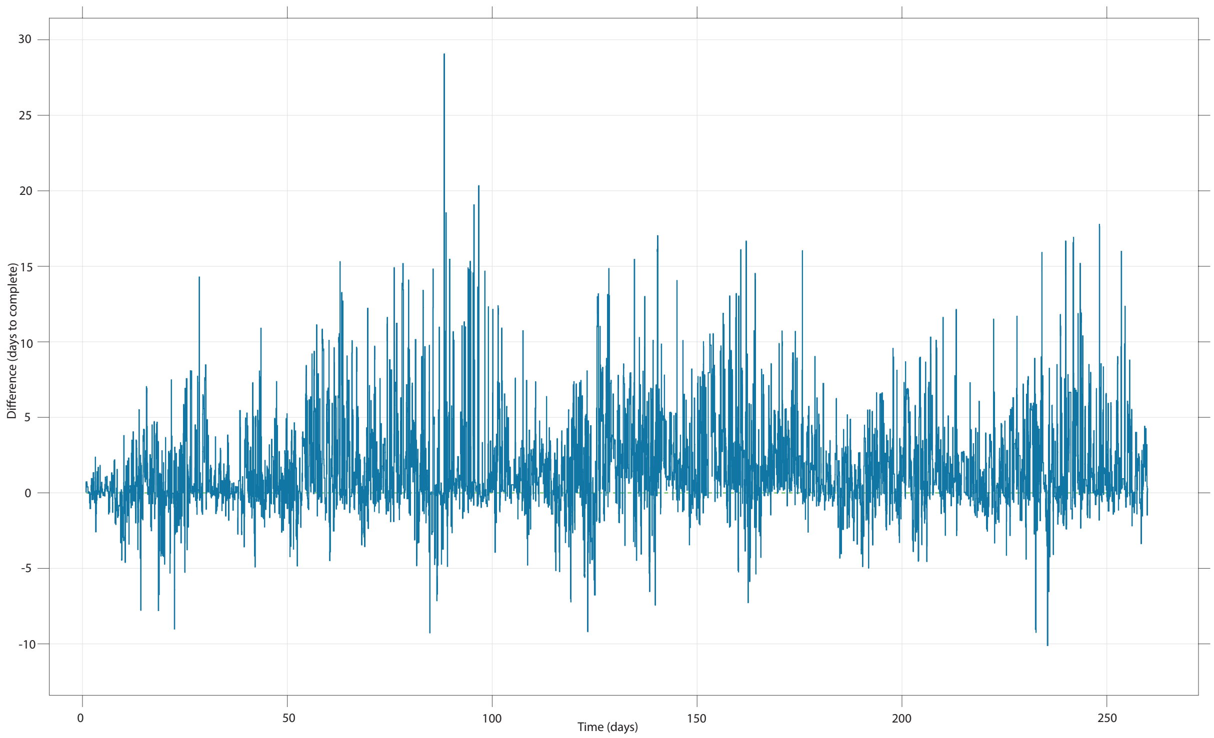

Figure 3.4: Comparison between SimEvents results before and after dynamic influences from Vensim - Medium-Priority Work.

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 Vensim input and Vensim output driven through Microsoft Excel utilizing 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.

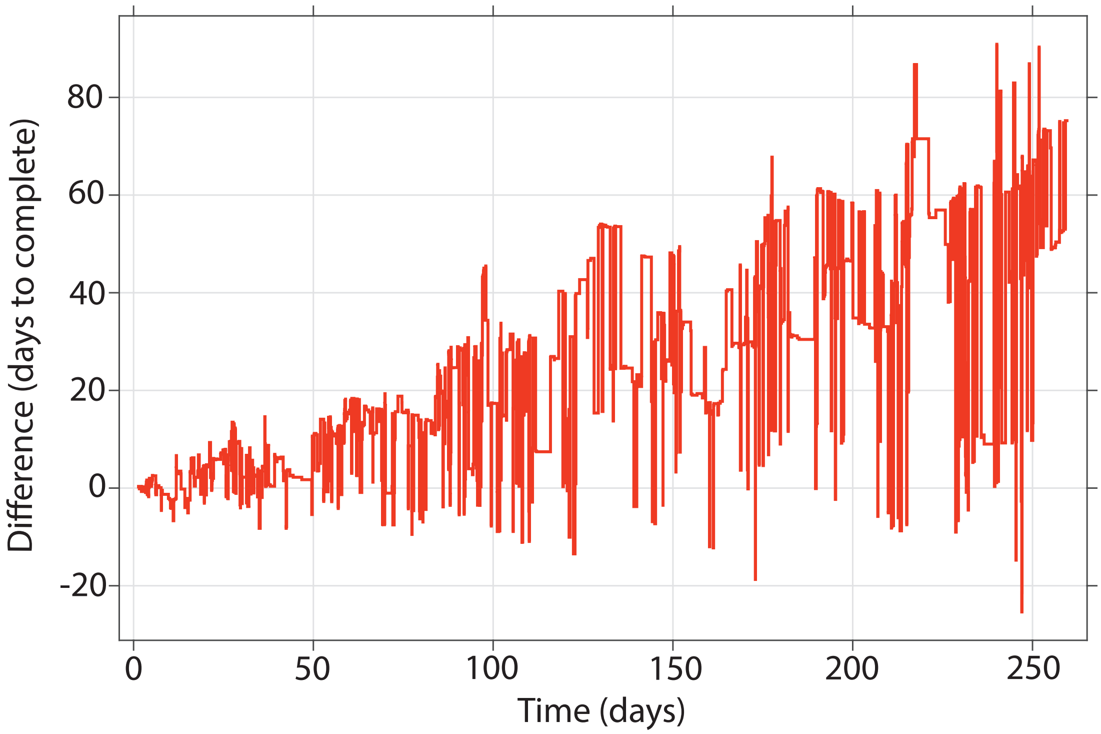

Figure 3.5: Comparison between SimEvents results before and after dynamic influences from Vensim - Low-Priority Work.

- Analyze the results of the model and generate statistics.
- Output the 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 of each iteration of both simulation models are maintained in Matlab throughout the exercise.

Integrating and Iterating the Models

Due to its 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 3.6 illustrates the integration and iteration logic.

Reducing Iteration Length

As noted in Section 2.3.3, 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.

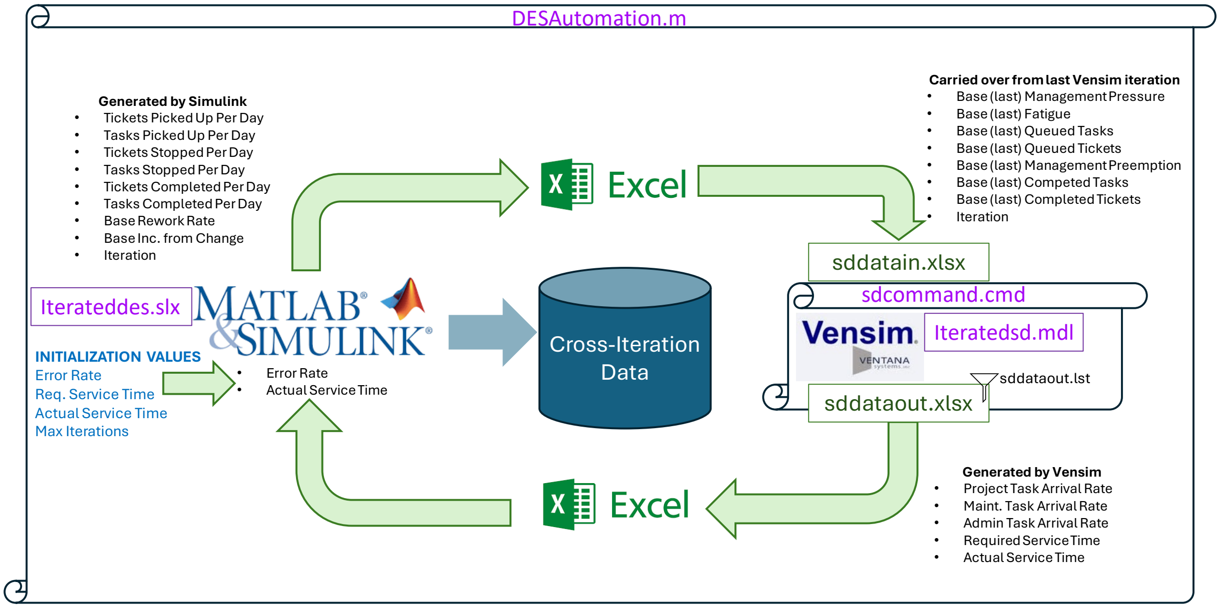

Figure 3.6: Scripting, parameters, and logic flow of iterations.

- 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 end-ing 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 do not 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 the 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.

3.6 Uncertainties in the Models

3.6.1 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 (interarrival 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.

- Currently, time accounting is 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 (service time) spent 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 were 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.

3.6.2 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 PDF coefficients used in the SimEvents model.

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

Table 3.1: Managerial Estimates for DES Inputs

Dependent Variables	LB	UB	X0	Comments
Required Service Time	0.060	0.255	0.255	Between 30 min and ~2 hours (115 min) required, on average, per task (event)
Available Service Time	0.060	0.240	0.123	Between 30 min and ~2 hours (115 min) available, on average, per engineer (server)
Interarrival rate of admin tasks	0.064	0.255	0.233	Between 4 and 16 tasks per day for the team, on average
Interarrival rate of project tasks	0.021	0.342	0.342	Between 3 and 46 tasks per day for the team, on average
Interarrival rate of maint tasks	0.064	1.027	1.027	Between 1 and 16 tasks per day for the team, on average

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

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.

3.6.3 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 3.7. 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.

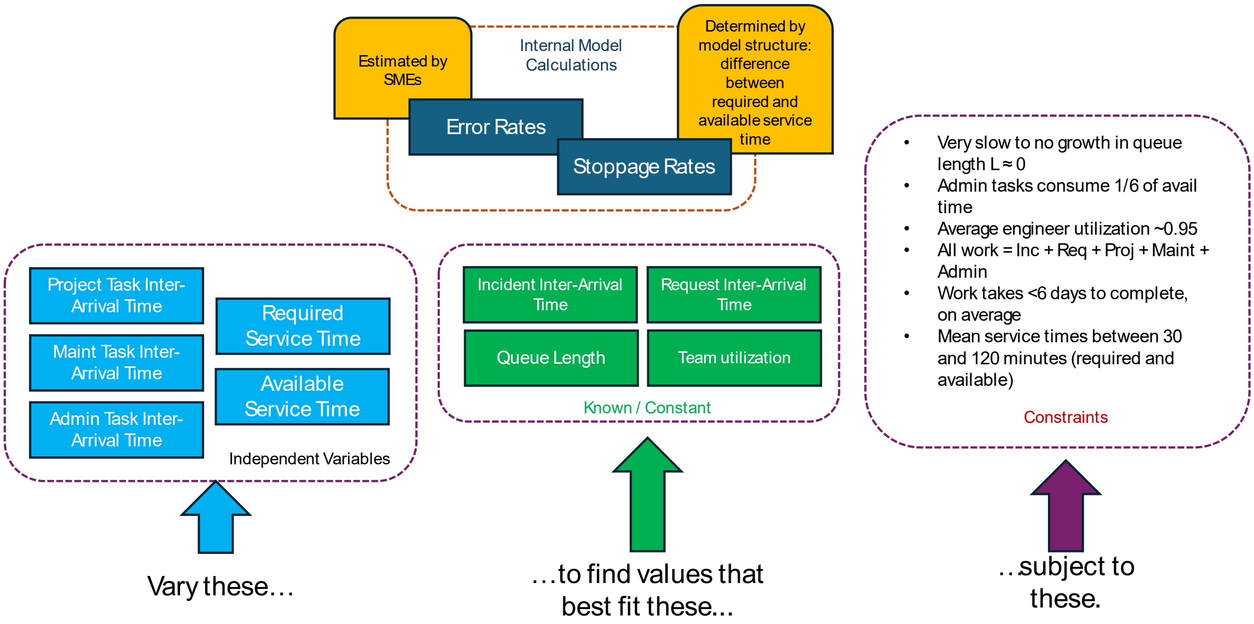

Figure 3.7: Schematic of regression analysis.

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 resulting residuals and optimal values for the dependent variables never stabilized. According to Rheinhart (Section 1.7) [123], 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 interarrival 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

Table 3.3: Optimal Variable Results from Direct Search

	Pattern Search	Particle Swarm
Sum of Least Squares	3.011	1.968
Available Service Time	0.196	0.342
Required Service Time	0.372	0.599
Interarrival rate of project tasks	0.234	0.241
Interarrival rate of maint tasks	0.117	0.154
Interarrival rate of admin tasks	0.230	0.171

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 used. The results of both direct methods are shown in Table 3.3.

These results imply a total work item 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. Furthermore, 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.

More concernedly, these results indicate that each work item will, on average, take almost twice as long to complete (288 minutes, or 4.8 hours) than is available to an engineer to work without stopping (164 minutes, or just over 2 hours). Further, at a rate of roughly 21 work items per day, this implies that roughly 756 hours (6,048 minutes) of work arrive daily when a total of 32 hours is available for a team of four engineers – 23 times more than is available. This

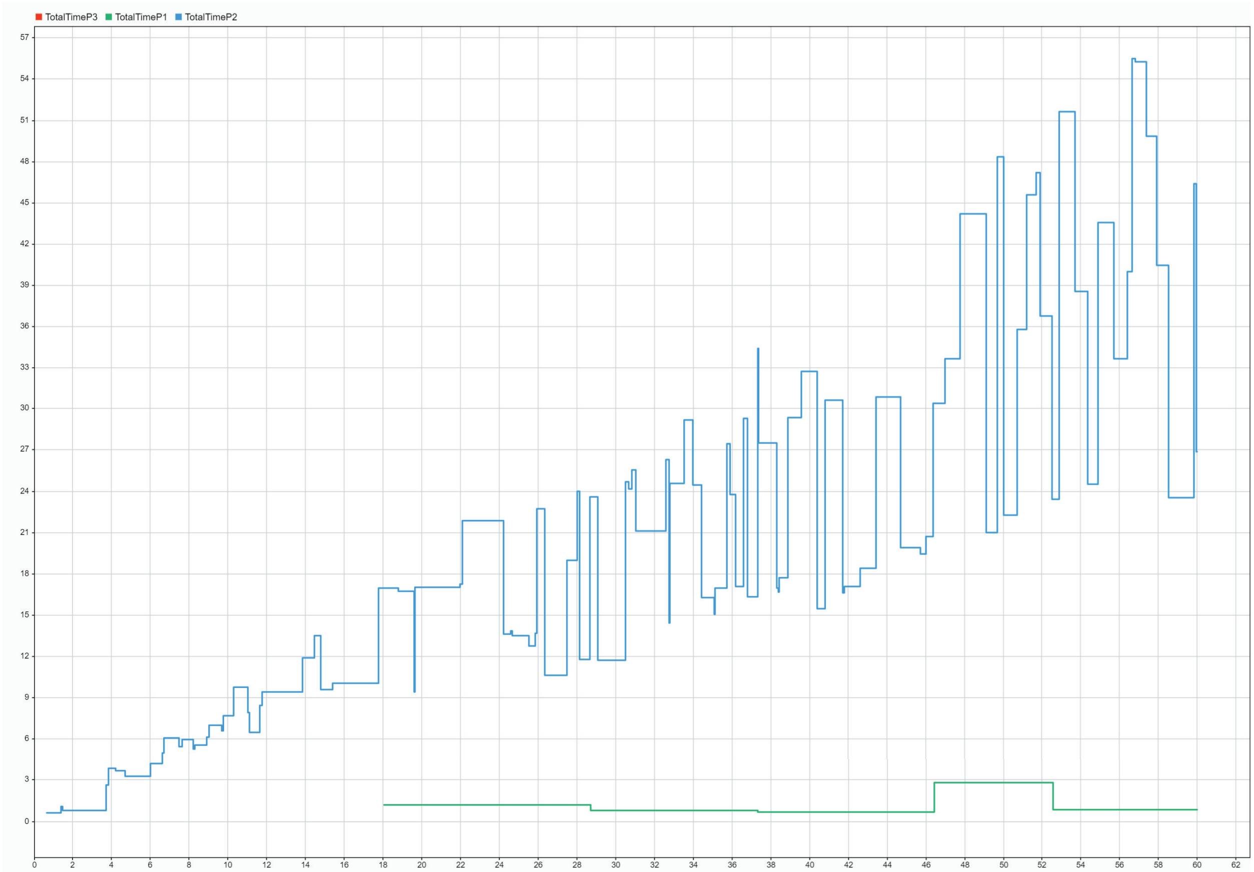

Figure 3.8: SimEvents Results using regression results for independent variables.

of course means that queueing will begin immediately, and that most work items will never be touched (especially low priority items). This does not match actual results within the case study organization. Figure 3.8 shows the impact of this on the queue times for high- (green) and medium-priority (blue) work items. With these parameters low-priority items remain queued and are never worked or completed.

3.7 Sensitivity Analysis

Using the results of the particle swarm regression as the "best" values for the five unknown variables, a sensitivity analysis shows the following impact on queue depth at the end of an iteration (holding the team utilization constant at 1). The impact of the two service time variables (available

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	0.400	2.501	55%	0.409	2.446	54%
Interarrival rate of Request Tickets	0.502	1.994	186%	0.599	1.671	155%
Interarrival rate of all tickets	0.223	4.495	80%	0.243	4.116	73%
Interarrival rate of all tasks	0.058	17.169		0.061	16.467	
Interarrival rate of all work	0.046	21.664		0.049	20.582	
Actual Values						
Interarrival rate of Incident Tickets	0.220	4.540				
Interarrival rate of Request Tickets	0.9307	1.074				
Interarrival rate of all tickets	0.1781	5.615				
Implied "True" Values						
Interarrival rate of all tasks	0.0623	16.049		0.067	14.968	
Assumed admin task interarrival rate	0.2770	3.61		0.292	3.430	
Implied project + maint task interarrival rate	0.080	12.439		0.087	11.537	

and required) on the queue depth is minor, while as expected, an increase in the arrival rates for the three task types is much more strongly correlated with an increase in queue depth.

- As Actual Service times increase from roughly 2.5 to 3 hours, the queue depth increases very slightly – roughly 0.16 additional of a work item in the queue over 10 days, or an extra 1 work item every 90 days. See Table 3.5.
- The impact of an increase in Required Service times is similar. As the time required to complete a work item increases from 4.3 to 5.3 hours, the queue depth increases by roughly 0.14 additional work items, or less than 1 work item every 90 days. See Table 3.6.
- As the Project Task Arrival Rate increases from 3.76 to 4.6, the queue depth increases by 0.57 additional work items in the queue, or almost 3.5 every 90 days. See Table 3.7.
- As the Maintenance Task Arrival Rate increases from 5.9 to 7.2, the queue depth increases by almost 1.1 additional work items in the queue, or over 6.5 every 90 days. See Table 3.8.

- As the Administrative Task Arrival Rate increases from 5.3 to 6.5, the queue depth increases by 0.8 additional work items in the queue, or over 4.8 every 90 days. See Table 3.9.

Table 3.5: Sensitivity Analysis – Available Service Time

Available Service Time	Avg. Hours	Queue Depth
0.308	2.47	14.11
0.317	2.53	13.89
0.325	2.60	13.49
0.334	2.67	13.50
0.342 (base)	2.74	13.96
0.351	2.81	13.93
0.359	2.88	13.92
0.368	2.94	13.94
0.377	3.01	13.96

Table 3.6: Sensitivity Analysis – Required Service Time

Required Service Time	Avg. Hours	Queue Depth
0.539	4.32	13.784
0.554	4.44	13.82
0.569	4.56	13.99
0.584	4.68	13.89
0.599 (base)	4.80	13.96
0.614	4.92	14.24
0.629	5.03	14.34
0.644	5.15	13.86
0.659	5.27	13.92

Table 3.7: Sensitivity Analysis – Project Task Arrival Rate

Project Task Arrival	Tasks / Day	Queue Depth
0.217	4.61	14.31
0.223	4.48	14.31
0.229	4.37	14.25
0.235	4.25	14.35
0.241 (base)	4.15	13.92
0.247	4.05	14.04
0.253	3.95	14.00
0.259	3.86	13.89
0.265	3.77	13.74

Table 3.8: Sensitivity Analysis – Maintenance Task Arrival Rate

Maint. Task Arrival	Tasks / Day	Queue Depth
0.139	7.20	14.42
0.143	7.01	14.13
0.147	6.82	13.91
0.150	6.65	13.53
0.154 (base)	6.48	13.743
0.158	6.32	13.89
0.162	6.17	13.39
0.166	6.03	13.46
0.170	5.89	13.32

Table 3.9: Sensitivity Analysis – Administration Task Arrival Rate

Admin. Task Arrival	Tasks / Day	Queue Depth
0.154	6.49	13.79
0.158	6.31	14.03
0.163	6.14	13.45
0.167	5.99	13.38
0.171 (base)	5.84	13.32
0.176	5.70	13.39
0.180	5.56	13.46
0.184	5.43	13.46
0.188	5.31	12.98

3.8 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 the 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 are 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.

3.9 Improvement Focus Areas

The models reinforce several common sense improvement targets, such as reducing *completion (sojourn) 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, as well as between different functional leaders. Given the complexity involved in large organizations, this can quickly become impractical at scale, where the volume of work and the number of teams due to technical specialization combine rapidly.
- Generally, limit the amount and manage the priority of all types of work flowing into the system, including operational tickets, project tasks, and administrative work.
  - Limit the number of projects in the process through a governance and prioritization function, that is, portfolio management.
  - Manage the operational tickets assigned to a team. Improve the accuracy of the triage and assignment of tickets to the appropriate team and staff member based on skill type and skill level required and available.
  - Reduce the amount of administrative work assigned to technical resources.
  - Improve the quality of work execution in order to reduce the amount of rework and / or the number of incidents resulting from planned changes.
  - Develop a process to prioritize both project tasks and operational tickets. Typically, tasks are prioritized within the tool used to manage them, where tickets are prioritized separately within a different tool. For teams involved in completing both types of work, functional managers need a way to prioritize between both.
  - 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 (especially 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 re-

quires 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.

- Continuously improve the skill level of resources through documentation and training to reduce error rates. In addition, improve the skill types of resources to 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 more highly skilled and cross-trained staff are highly recruited externally and must be actively retained.
- Create cross-functional teams to prevent work transfers between department queues, either permanently or on an ad hoc basis. This is more routinely seen in project-driven areas and organizations, and particularly in larger projects, but it can also be in response to critical complex issues. 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 the 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 that represent targets for automation, summarized here:

1. Reduce overall completion times by significantly reducing or eliminating queue times.
   - Automate response to incidents or requests to begin work more quickly. This can also improve responsiveness after hours, where skilled staffing is often reduced or is primarily on call.
   - Reduce switching costs and cross-team coordination by partial or full automation of complex cross-team processes (such as server provisioning).
2. 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).
3. Reduce the time required service time for completion of a work item (ticket or task) through partial automation of associated tasks. This can be accomplished through programmatic use of APIs or through Robotic Process Automation (RPA) to automate work that requires the sequential use of several tools / applications by the assigned resource to complete repetitive tasks.
4. Completely eliminate specific work activities from a team's queue through fully automated responses and execution of the work, without human supervision or approval.

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.

Of course, there are potential “soft” benefits to be derived from automation. As the work best suited to automation is often both high-volume and predictable, it also tends to be tedious for the engineering staff. Using automation for these work items allows staff to focus more of their available time on new or complex work (commensurate with their level of skill). This leads not only to faster completion of those tasks, but also to higher levels of work satisfaction and morale.

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* [124]. 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 a driving example that determines the sequence in which tools and infrastructure components are incorporated into the SDI.

Chapter 4

Developing an SDI Architectural Framework with MBSE[3]

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 [94] 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 2.6 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 (CDMs and LDMs).
- 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. This architecture is the basis of the one ultimately built by the case study organization to automate the server provisioning process outlined below.

[3]Note that a significant subset of the content in this section has been submitted for publication in the journal IEEE *Access*. Furthermore, a small subset of the information in this chapter is included in an article in the journal IEEE *IT Professional*.

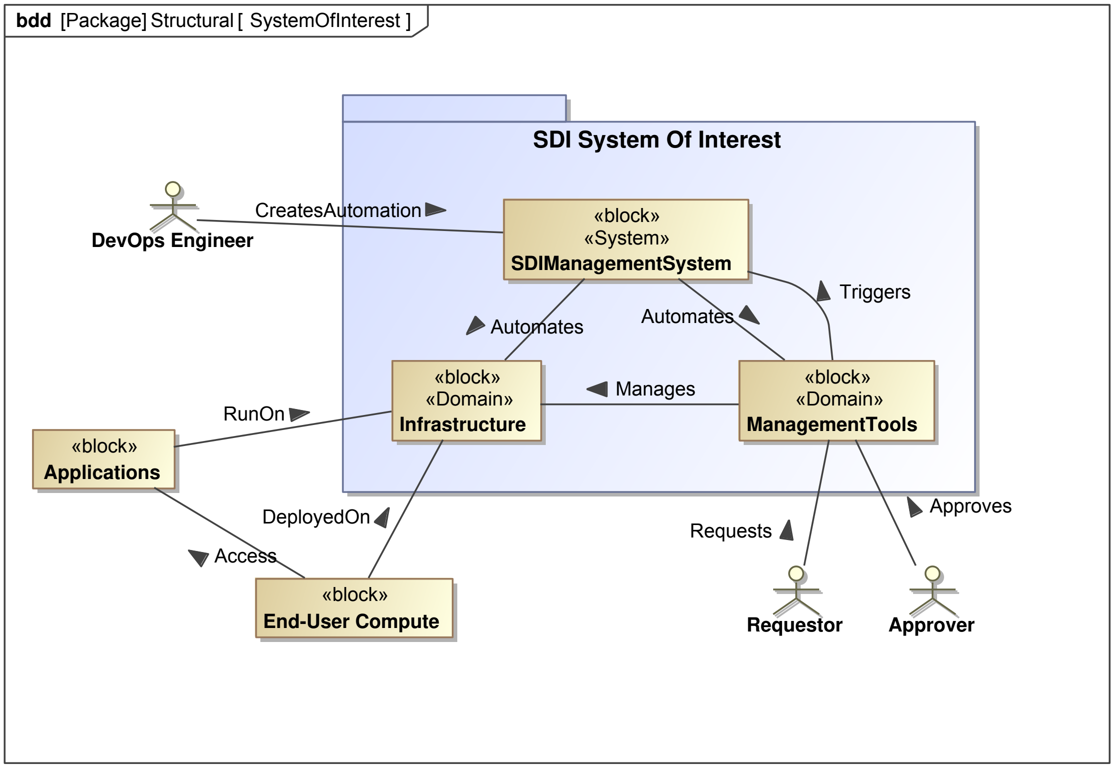

Figure 4.1: SDI System of Interest.

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 SDI Management System in combination represent the System of Interest (SoI) as shown in Figure 4.1 and are further refined in Figure 4.6. 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. Note that the *Applications* block, while often included in the automation system by DevOps engineers in product development-focused organizations (for example, in code deployment pipelines or for use in "A/B testing", is excluded from the SoI – although they can certainly be layered on top of the SDI in more mature organizations.

△ 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 Solution Requirements

A functional SDI actually represents a *platform* for automation – interesting, but not particularly useful in and of itself as a platform is simply a construct on which to run useful code. What an organization decides to automate on top of that platform is the actual source of value and depends on the analysis of where improvements are most required, the basis of 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 of the SDI solution, and Figure 4.3 represents the highest level of non-functional requirements (business and performance). Functionally, the SDI must provide the ability to execute code, manage that code, interface between component subsystems (and provide the ability to log into them), and finally monitor all of its activities. The system must perform these functions securely and must also be as resilient and performant as necessary to support the use cases being automated.

4.2.2 Example System Provisioning Requirements

The use cases for an SDI 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, such as writing and maintaining the execution code, or support the underpinning services such as backup or monitoring.
- That of the use-case stakeholders (e.g., requesters) that are automated by those users to meet business needs.

For the purposes of this chapter, the focus is on the latter, primarily because these are foundational to the research questions and also because they illustrate the structure and function of the SDI. 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, addresses many of the opportunities identified in the simulation models, and represents a solid technical platform on which to build automation to address subsequent use cases and expand as needed. The infrastructure provisioning requirements have been derived / expanded in Figure 4.4 and are mapped to the high-level SDI requirements in Figure 4.5.

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 sys-

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.

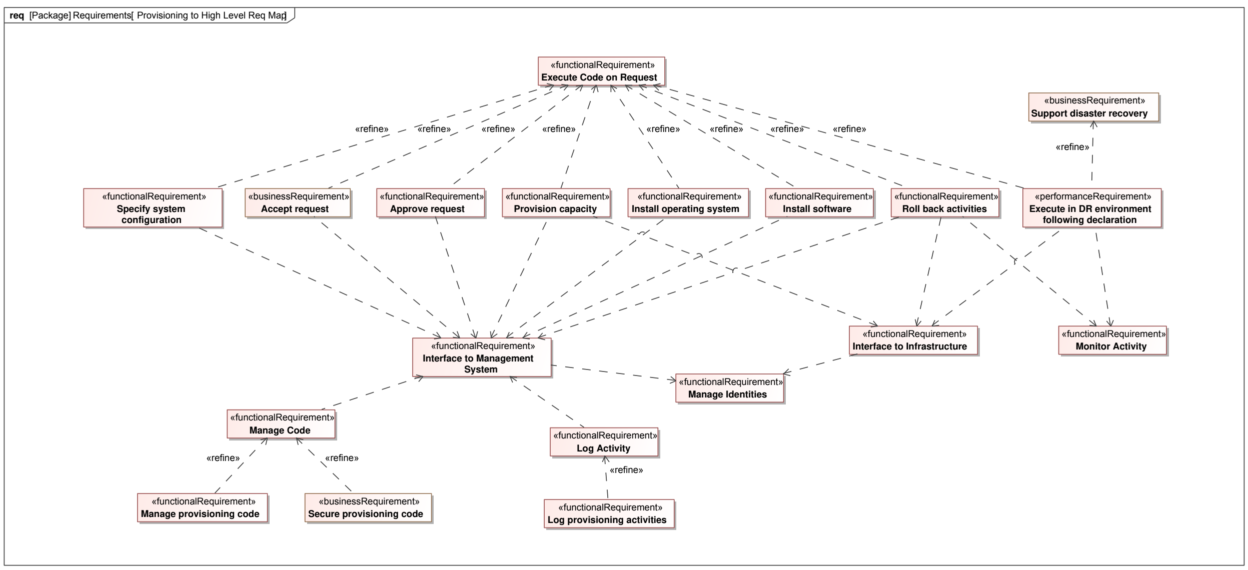

Figure 4.5: Mapping of provisioning requirements to high-level SDI requirements.

tem 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 (or equivalent head of IT), and the key stakeholders are the teams that manually deploy and support these systems today. This particular use case is (usually) not time-critical; however, the system must ultimately be capable of both request-based and event-driven / near 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 in Chapter 1 many organizations (in particular healthcare providers) 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 that are custom-developed, either on-premises or in the cloud. Historically, COTS applications and their underlying components have been provisioned and built "by hand", with IT engineers with various skills who manually deploy servers, assign storage, and install operating systems, databases, and other application components. Enterprise IT is increasingly shifting towards SDI to manage these environments in development-focused organizations, which is the combination of public, private, or hybrid cloud technologies with automation-based management. SDI and its subtopics of software-defined networking, DevOps, infrastructure automation, and IaC are well-studied in the context of off-premise cloud service providers, but there is limited application of these concepts to the on-premise IT infrastructure (e.g., private and hybrid clouds) on which COTS applications are deployed.

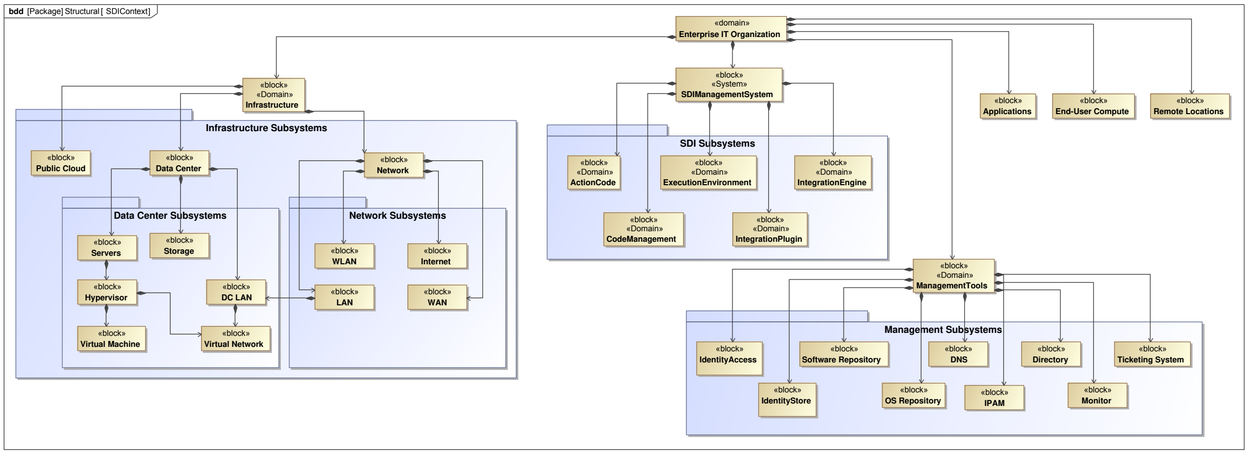

Figure 4.6: SDI Management System in the context of existing technical infrastructure and management tools.

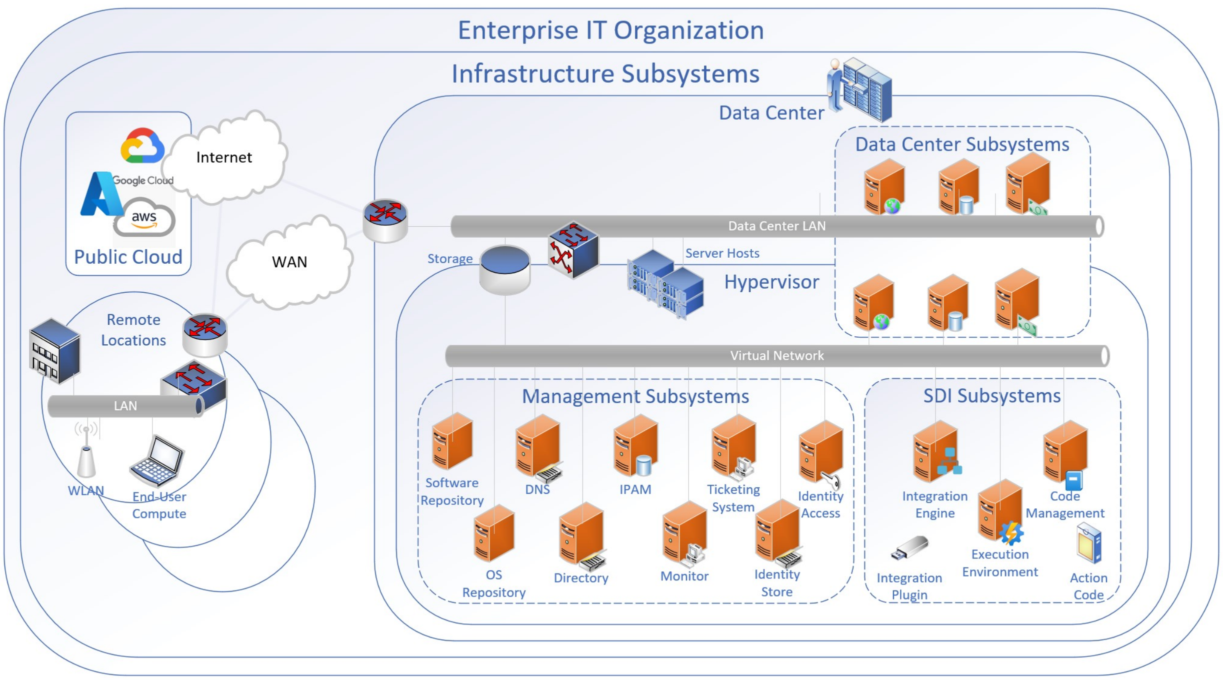

Figure 4.7: Equivalent Visio Diagram of SDI Management System in the context of existing environment.

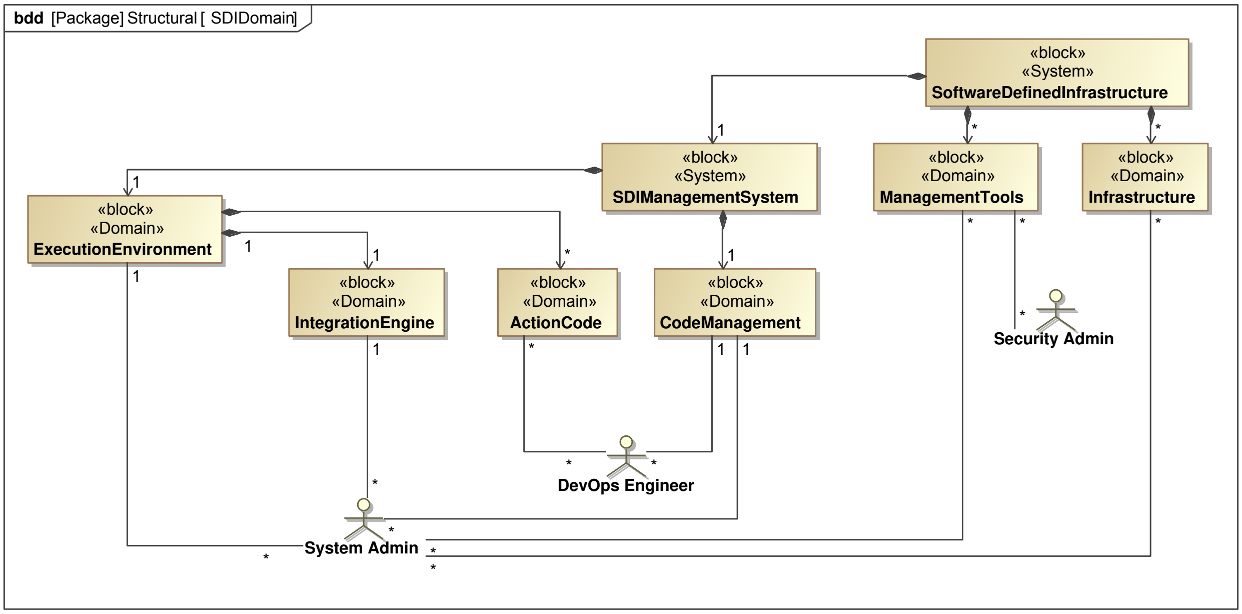

Figure 4.8: High-level domain diagram of the SDI.

4.3.1 Structural Perspective

Domain Definition

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. 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 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.

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 APIs to *Infrastructure* and *ManagementTools* 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: examples include *IdentityAccess* and *ExecutionEnvironment*, and can be either developed against exposed APIs or realized through a "plugin" architecture that enables predefined configuration components obtained from the various product vendors.
  - Allocated Requirements: functional requirements 5, 6, 7; non-functional requirements 3, 4, and 5.
- *Infrastructure* and *ManagementTools*
  - Owner: Customer system administrators.
  - Description: represents the external infrastructure and management systems that are managed by the system (such as the ticketing system or a hypervisor environment).
  - Operations: varies according to the APIs exposed by the vendors of these systems.
  - Data: varies based on the function of the interfaced systems
  - Interfaces: defined in the APIs defined by the vendors, or via pre-defined "plugins".
  - Allocated Requirements: functional requirements 5, 6; non-functional requirements 3, 4, and 5.

- *ExecutionEnvironment*
  - Owner: DevOps engineers.
  - Description: this is the environment in which the *ActionCode* runs and includes the Orchestration Engine (which 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: functional requirements 1, 2, 3, 9 directly, and 8 via the *ActionCode*; non-functional requirements 3, 4, and 5.
- *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 according to the intent of the *ActionCode* (i.e., the automated use case) and support of APIs exposed by *InterfacedSystems*. Implements logging of its activities.
  - Data: code repository, code version, code status.
  - Interfaces: varies depending on the intent of the *ActionCode*.
  - Allocated Requirements: functional requirements 1, 2, 3, 4, 8, 9.

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

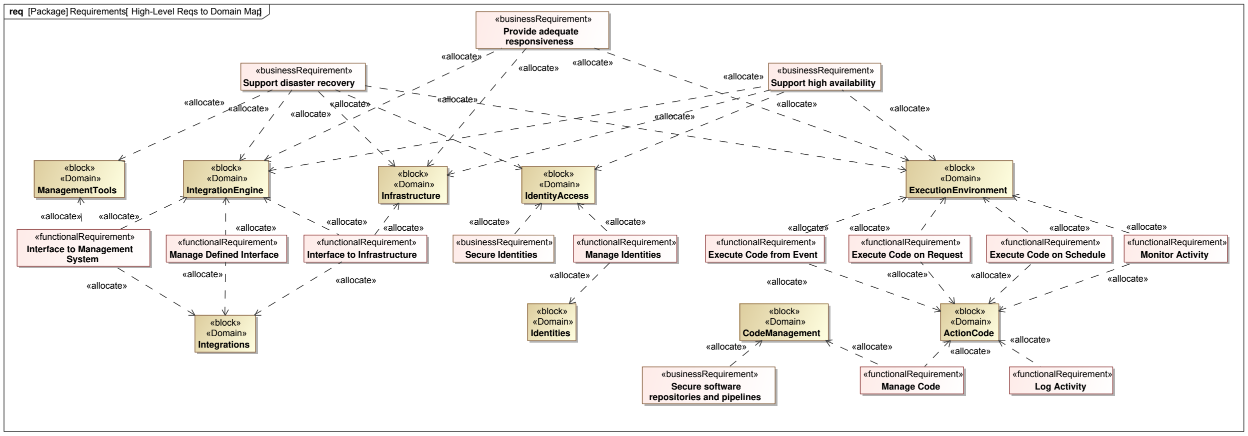

Figure 4.9: Allocation of high-level requirements to SDI domains.

4.3.2 Behavioral Perspective

According to MBSAP the Behavioral Perspective uses use case, activity, and sequence diagrams (and others if needed, such as BPML) 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.

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

Figure 4.11 outlines the activities that could occur during the use case "Automatically Respond 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. Note that this activity diagram also depicts the use case "Process Request" (although the addition of an Approver

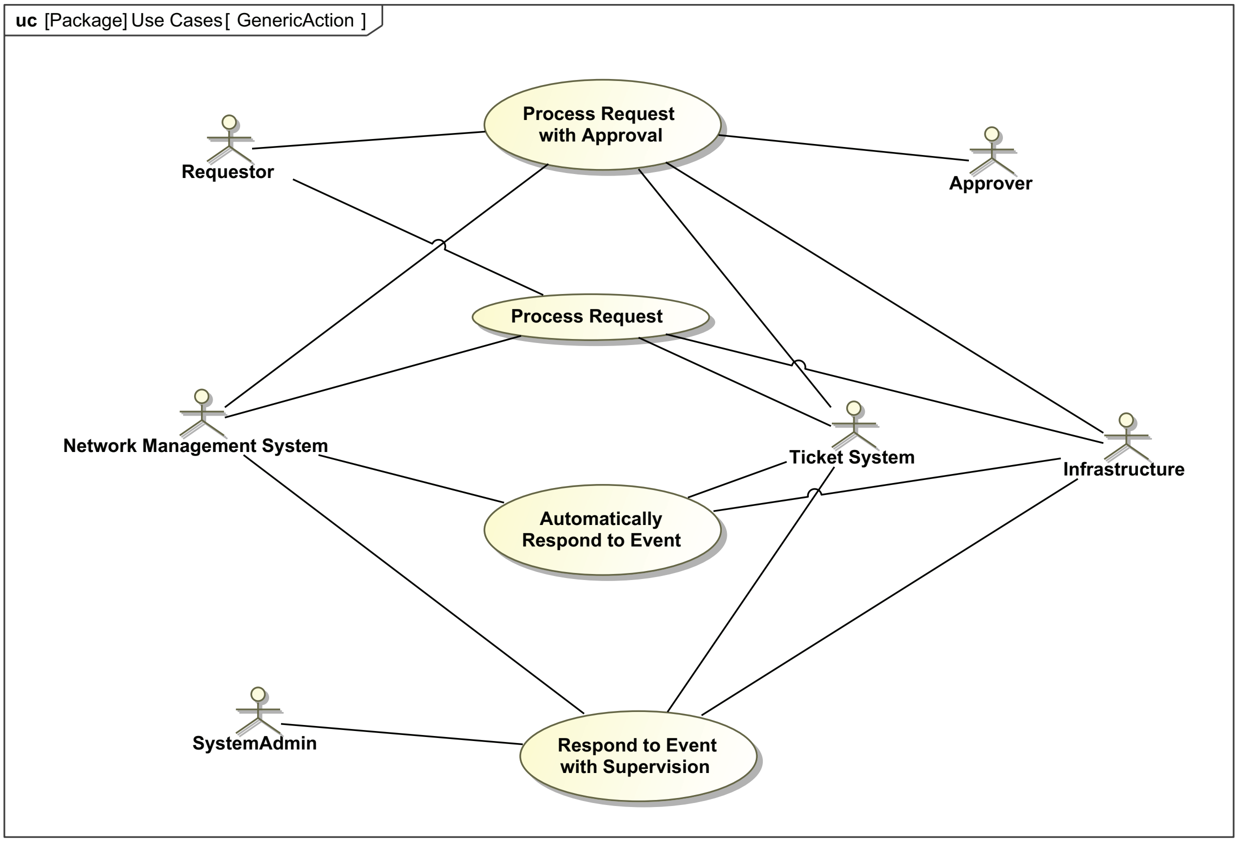

Figure 4.10: Use case depicting the response of the SDI to a request or event.

may be appropriate to some cases), while the remaining two use cases would require the addition of an Approver or SystemAdmin swim lane to the diagram.

Sequence Diagrams

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

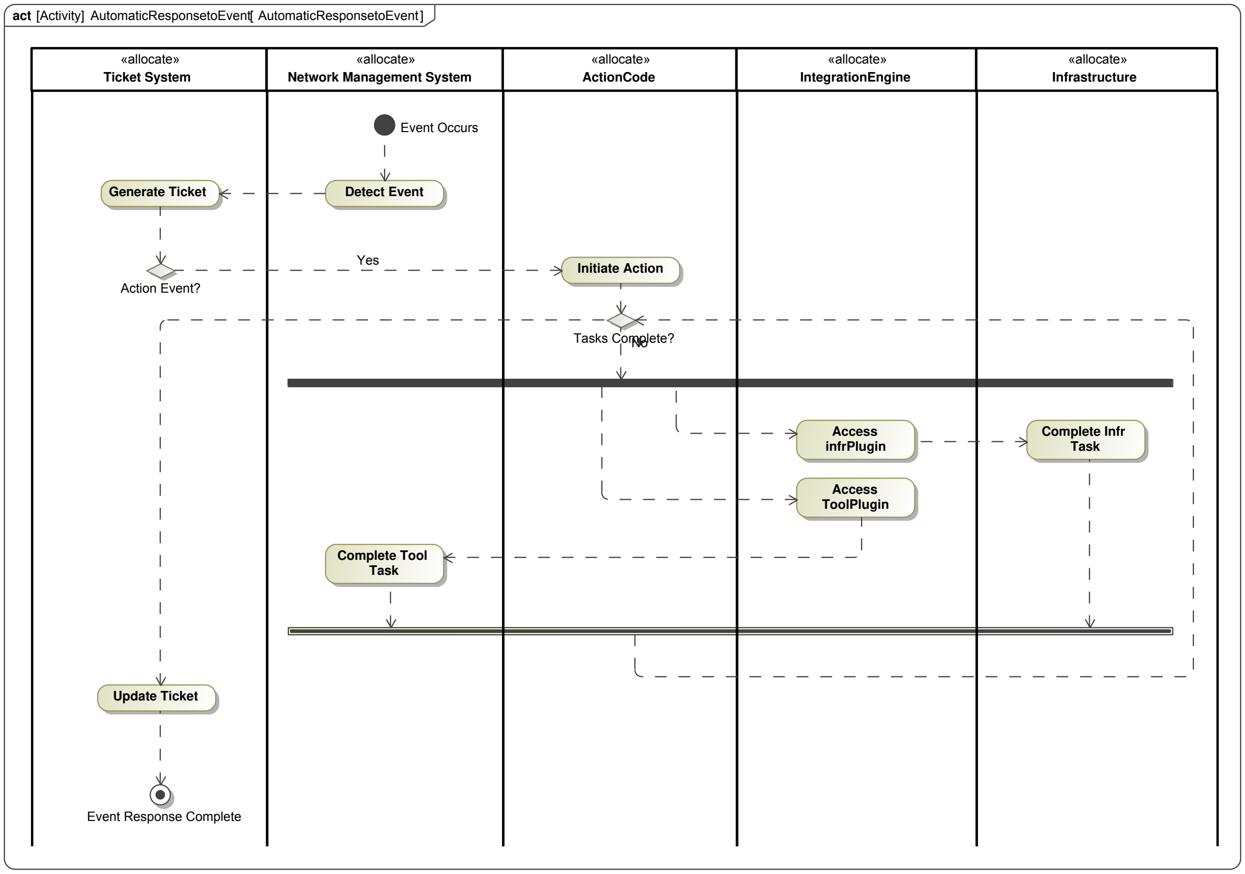

Figure 4.11: Activity diagram outlining the SDI response to a generic event.

4.3.3 Data Perspective

The Conceptual Data Model (CDM) in the MBSAP captures the general information categories of a system. A high-level CDM for the SDI is shown in Figure 4.13, The specific APIs leveraged via the code and the subsystems acting by 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) and would be further elaborated in the Logical Data Model (LDM). 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.

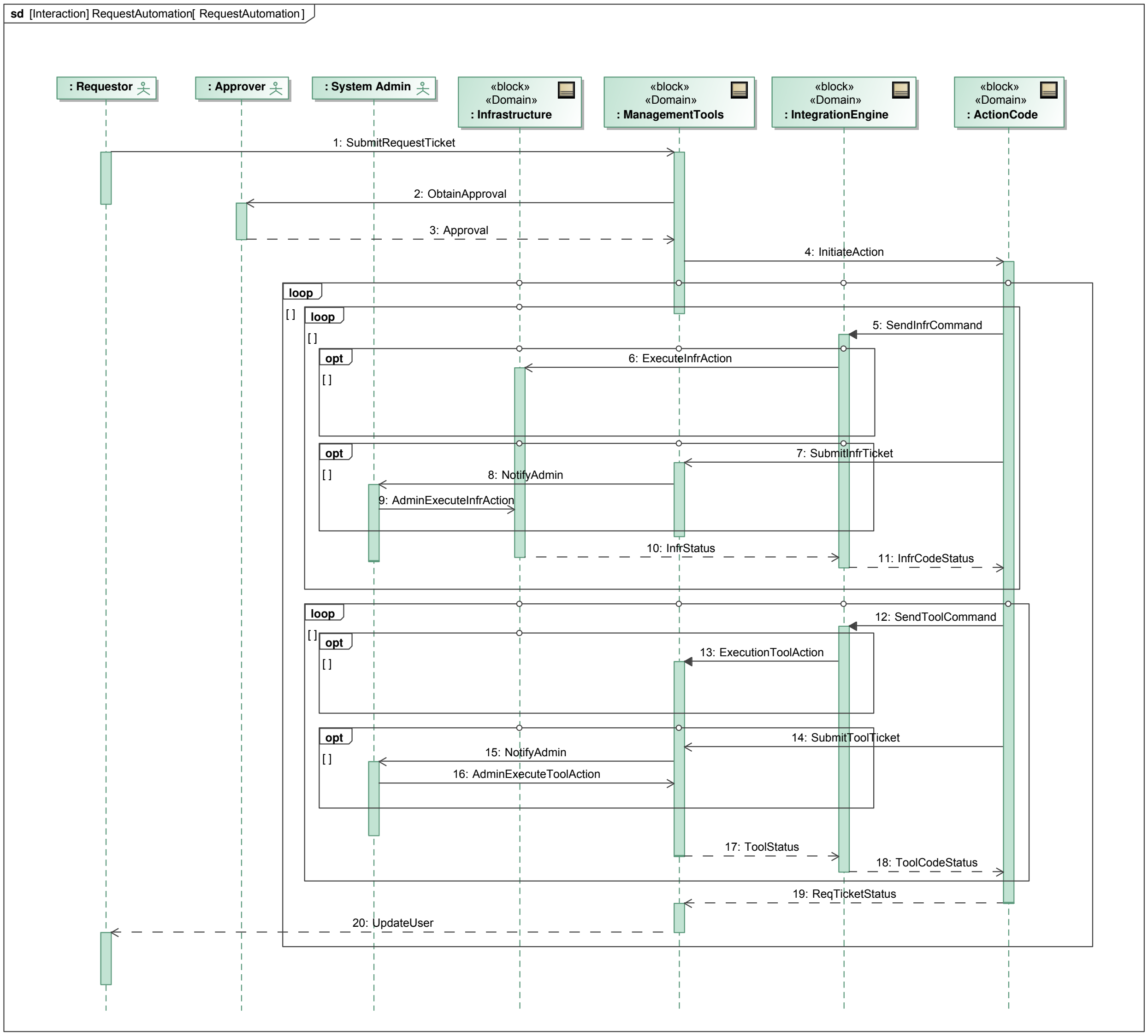

Figure 4.12: Sequence of activities between system modules in the provisioning process.

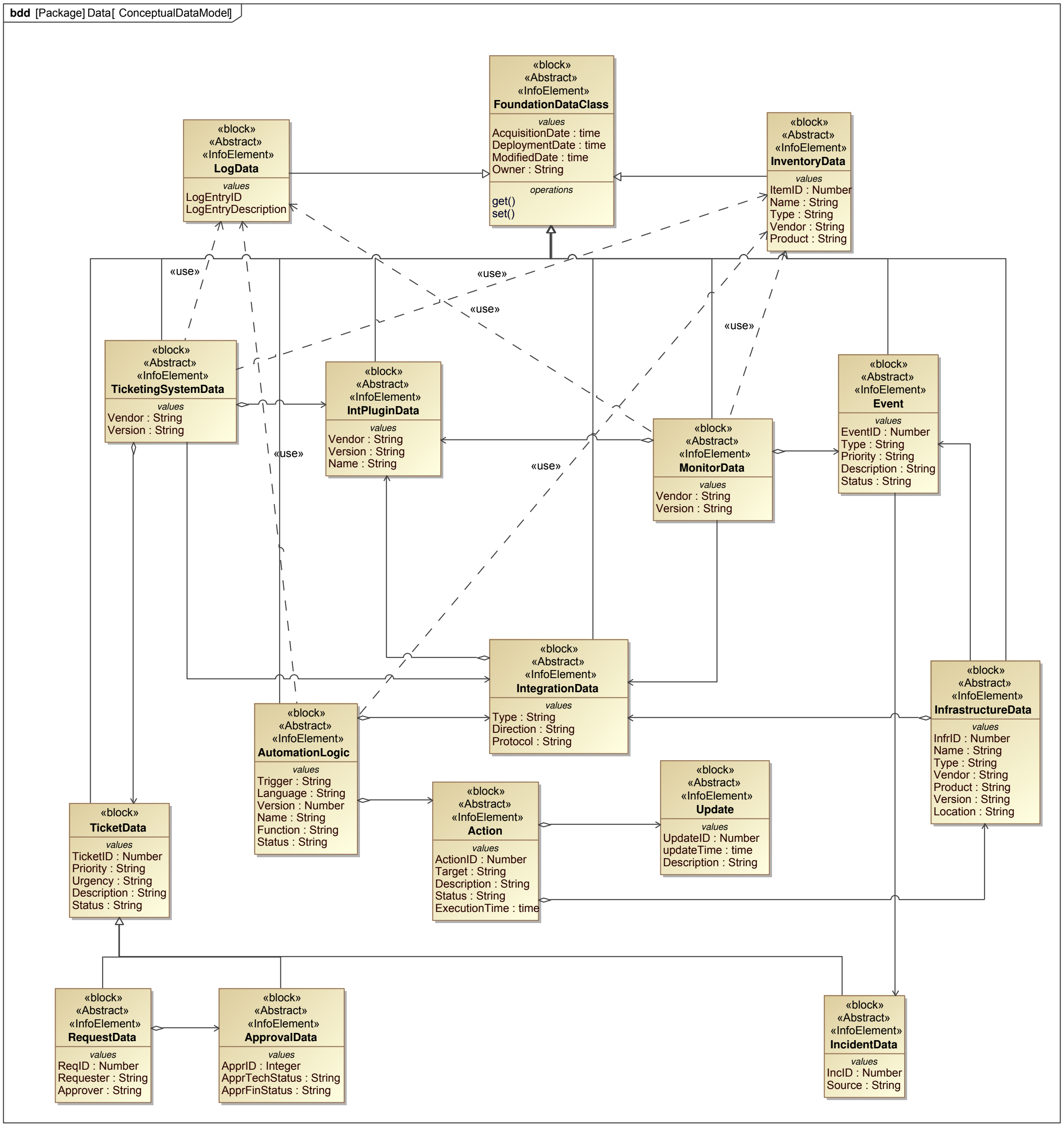

Figure 4.13: Conceptual Data Model for the SDI.

4.3.4 Services Perspective

Many services are supplied by the systems within the three domains and exposed via APIs to the *AutomationCode*. Any which are not exposed directly by the products within the domains, or that require significant coordination through code, are candidates for the creation of custom-defined services. These could include, but are not limited to, the domain-level services leveraged by the provisioning function listed below.

- *SDIManagementSystem* domain services are those which would ideally be provided by the product used as the core of the solution (for example, Terraform).
  - *ConfigureFunction*: provides the ability to register a new automated function (executed by a specific instance of *ActionCode*) in the SDI system.
  - *MonitorRequests*: provides the ability to recognize that a new request for a configured automated function has been received in the integrated ticketing system.
  - *MonitorEvents*: provides the ability to recognize that a new event of interest 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.
  - *RemoveFunction*: provides the ability to de-register an existing automated function in the SDI system.
- *Infrastructure* domain services are those that would be expected to be exposed by the product vendors through defined *api*s.
  - *ConfigureInterface*: provides the ability to configure a network interface on a network device or physical server host.
  - *CreateVirtualMachine*: provides the ability to create a new Virtual Machine within a hypervisor environment, with parameters specified for memory, processor, network in-

terfaces, and operating system. Additional services for stopping, starting, and deleting virtual machines should also be available.

  - *CloneVirtualMachine*: provides the ability to copy an existing VM (and its parameters) within a hypervisor environment.
  - *ConfigureStorage*: provides the ability to provision (or de-provision) storage of a specific type and speed, and associate it with an existing VM. This can be further refined as either "block" (raw) storage or "file" (formatted) storage.
  - *AttachImage*: provides the ability to boot a VM from a specific OS image, in the event that the VM is not "cloned" from an existing VM. 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 are, as above, those that the product vendors would expect to expose through defined APIs.
  - *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.

- *ManageBackup*: provides the ability to register, update, or remove a server or storage location for scheduled data or configuration backups. Additional services should also be available to manage individual backups.
- *ManageAlert*: provides the ability to establish, update, or delete attributes and thresholds for monitoring and alerting on a configured infrastructure component (such as a VM or storage).
- *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 sample 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 API.

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 six 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. In comparison, leading development organizations are known to provision entire application environments frequently throughout the day.

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, including disaster recovery environments), 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.

The key functional requirements for provisioning servers are shown in Figure 4.14. Note that additional requirements should be defined to modify or delete a provisioned VM – and to create, update, or delete storage, network, backup, or monitoring configurations, among other activities.

#	Name	Text
1	16 Automate system provisioning on request	Enable users to request systems for provisioning, consisting of compute, storage, network and software resources
2	16.1 Specify system configuration	Enable collection of system specifications
3	16.1.1 Specifiy application name	Select application name linked to the application inventory, or create a new application name in the inventory
4	16.1.2 Specify compute configuration	Specify memory and processor or select from standard options
5	16.1.3 Specify storage configuration	Specify storage type and capacity or select from standard options
6	16.1.4 Specify network configuration	Specify network attributes
7	16.1.5 Specify operating system	Select standard operating system image or leave blank
8	16.1.6 Specify software to install	Specify standard software to install or leave blank
9	16.2 Accept request	Accept request from ticketing system for execution
10	16.2.1 Forward request for approval	Submit new requests for technical and financial approval within 10 minutes
11	16.3 Approve request	Submit request to approver and obtain approval / denial decision, with conditions for selective automatic approval
12	16.3.1 Forward approved request for provisioning	Submit approved request to the provisioning system within 10 minutes of approval
13	16.4 Provision capacity	Provision compute, storage and network capacity following approval in the appropriate order
14	16.5 Install operating system	Install selected operating system onto each provisioned server
15	16.6 Install software	Install automatic and optionally selected software onto each provisioned server
16	16.6.1 Install standard software	Install required software such as patching, security, and accounting packages
17	16.6.2 Install optional software	Install other software on specified servers if required and packages exist in software repository
18	16.7 Roll back activities	Roll back provisioning activities if comopute, storage or network capacity not available
19	16.8 Log provisioning activities	Log all actions taken by the provisioning process, including successes and failures
20	16.9 Complete provisioning activities	Complete provisioning activities within 2 hours of approved request being submitted to system
21	16.10 Execute in DR environment following declaration	Ensure availability of system and execution code in DR environment to enable automated provisioning there after failover
22	16.11 Manage provisioning code	Create, update and delete code components as needed
23	16.12 Secure provisioning code	System must secure code and configuration information remain confidential and are not altered except to authorized users / processes
24	16.18.1 Secure software repository	Ensure confidentiality and integrity of software packages and images, and enforce change control processes
25	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
26	16.18.3 Secure provisioning pipeline	Ensure integrity of code and execution environments through defined development processes

Figure 4.14: Server provisioning functional requirements.

4.4.1 Structural Perspective

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

The BDD in Figure 4.15 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.

The Internal Block Diagram (IBD) shown in Figure 4.16 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.

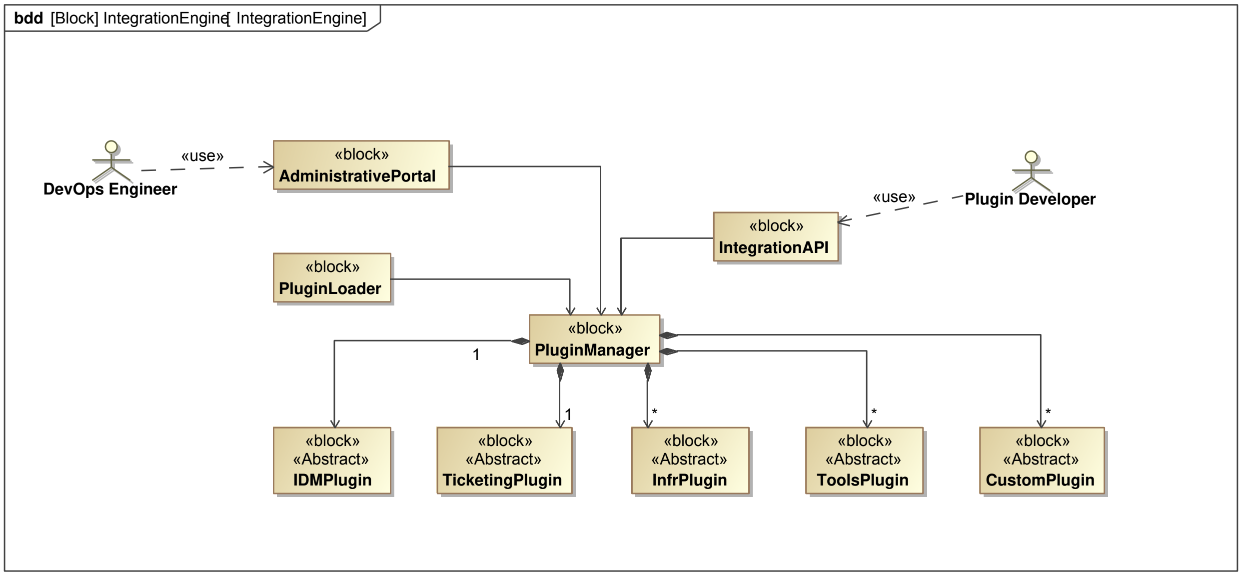

Figure 4.15: Block Definition Diagram of the integration engine.

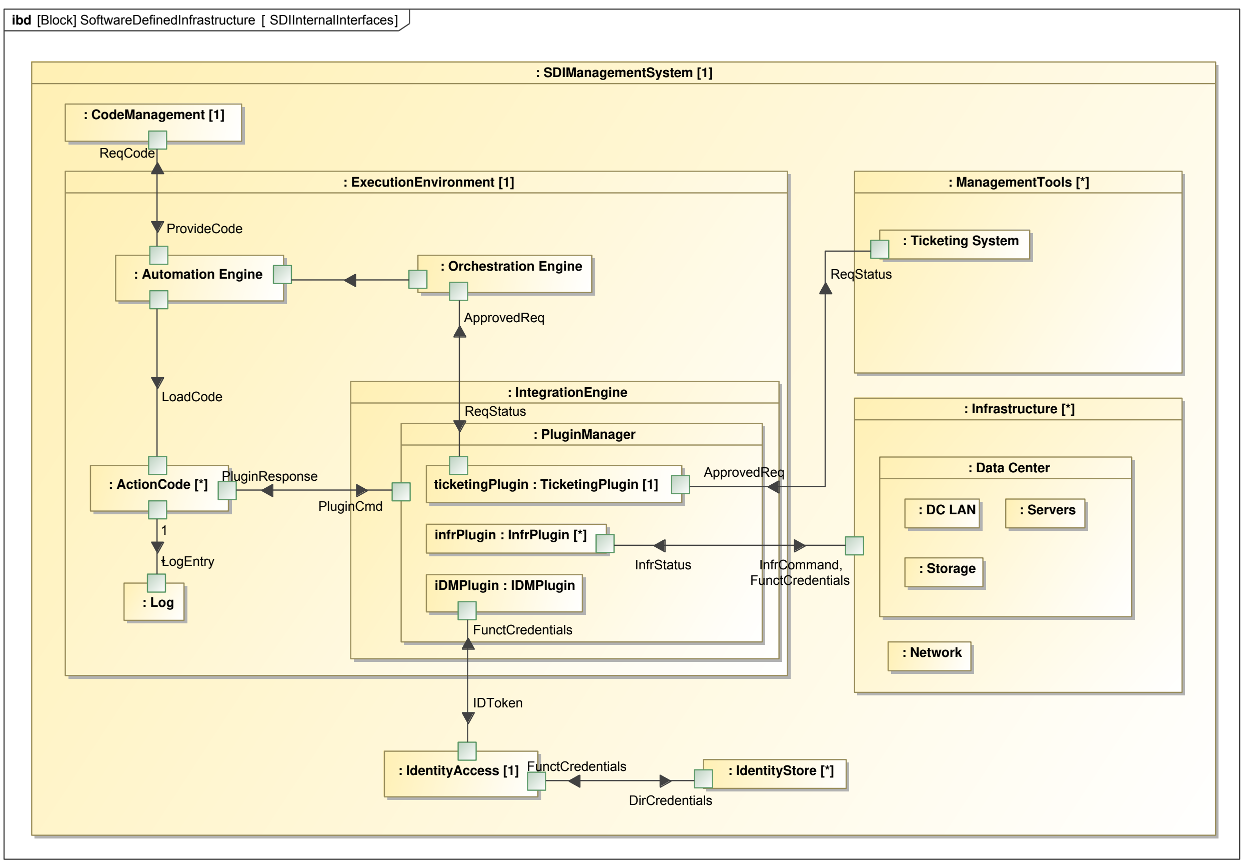

Figure 4.16: IBD showing interfaces between the SDI domains.

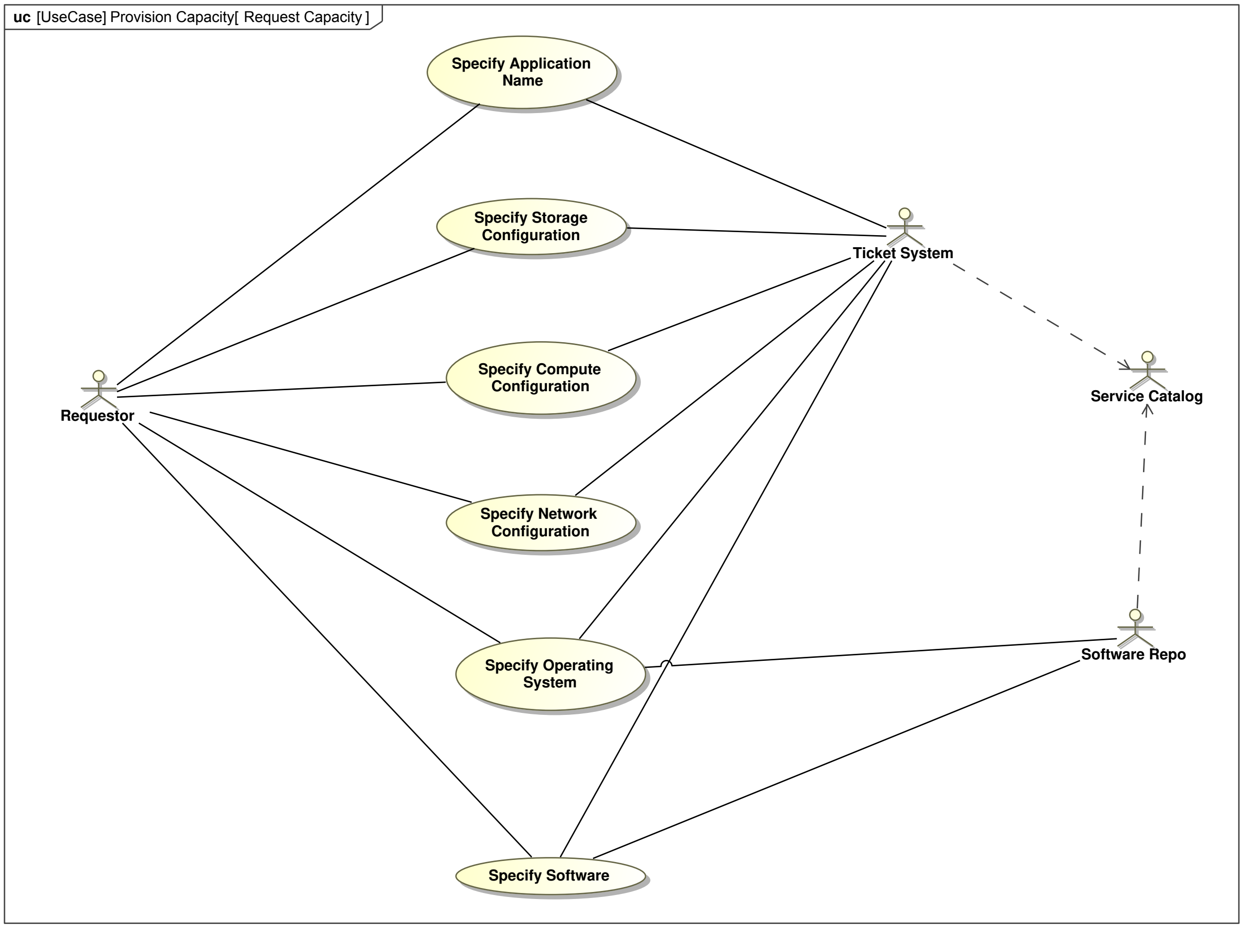

Figure 4.17: Use case for requesting server and storage capacity.

BDDs and IBDs can be developed for the *ExecutionEnvironmnet*, *CodeManagement* and *ManagementTools* domains.

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 4.17. The Provision Capacity use cases are shown below 4.18.

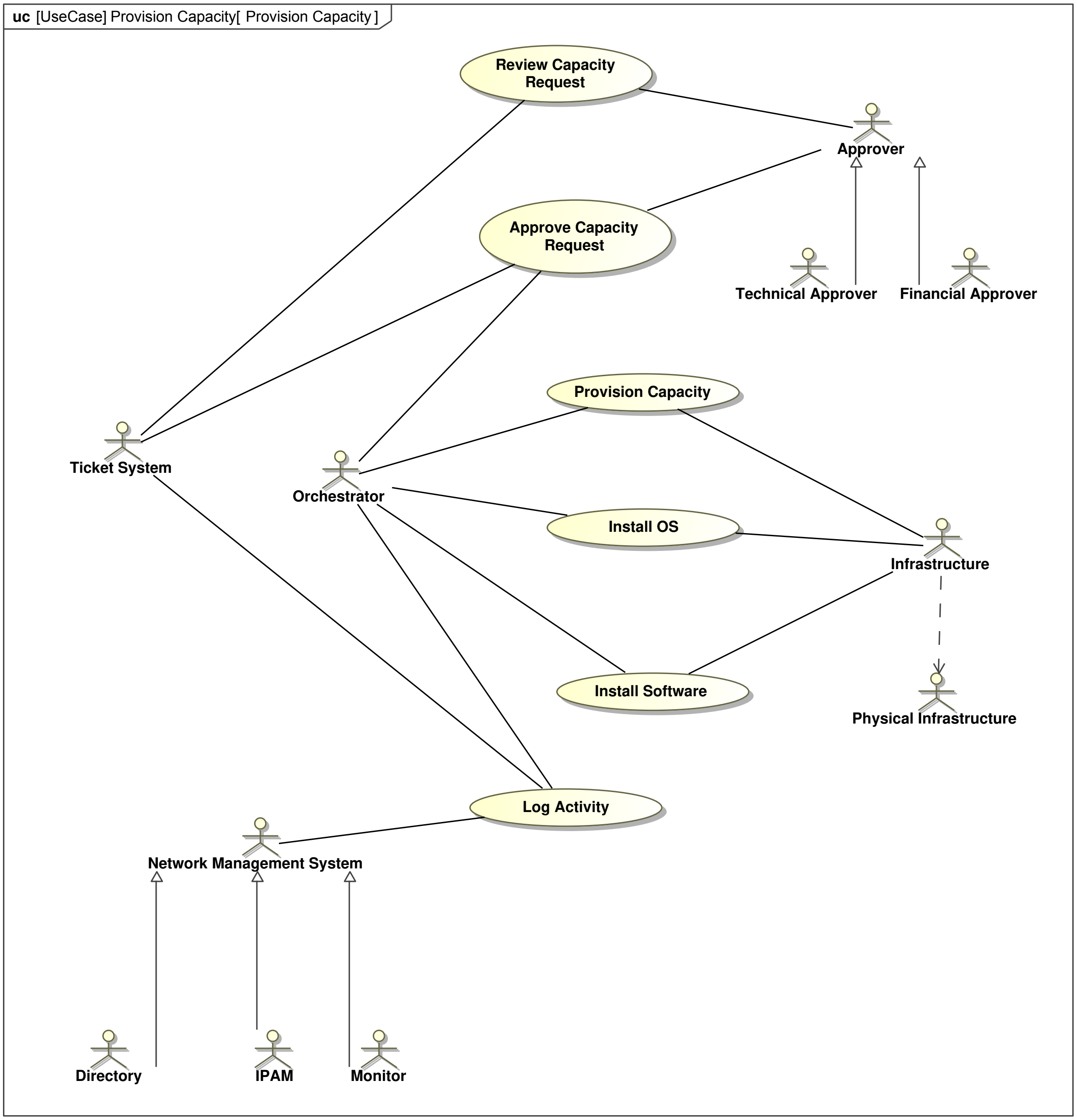

Figure 4.18: Use case for provisioning server and storage capacity.

Use Case Specifications

The following are specifications for three selected use cases; all others should be defined but are not shown.

- Specify the Compute Configuration (from Figure 4.17).
  - Owner – system administrators for the individual infrastructure domains.
  - 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 – the application must already be approved by the management and defined in CMDB.
  - Trigger – a new application has been approved, and the project to implement it has begun.
  - Post-conditions – all server configurations for the application are defined.
  - User roles – requester.
  - Data Objects – ServerData.
  - Primary scenario.
    - For each server needed:
      - The requester selects from the preconfigured server options (for example, "basic" or "high performance").
      - The system determines the cost of each configuration.
  - Secondary scenario(s).
    - For each server needed:
      - The requester defines custom values for the memory, processor, and local disk.
      - Steps 2-3 remain the same.
  - Allocated requirements: 16.1.1 – 16.1.6.
- Install OS (from Figure 4.18).
  - 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) is configured, connectivity established to OS repository.
- Trigger – on approval and deployment of server capacity
- Post-conditions – current OS patches deployed to each OS instance.
- User Roles – automated administrator identity.
- Data Objects – OSData .
- 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 – 16.5 Install the operating system.

• Log Activity (from Figure 4.18).
  - 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.
  - Post-conditions – N/A.
  - User roles – automated administrator identity.
  - Data Objects – LogData.
  - Primary Scenario – record made of each action completed.
  - Secondary scenario(s) – record made of each action failed / rolled back.
  - Allocated requirements – 16.8 Log provisioning activities.

State Machine Diagrams

The diagram in Figure 4.19 depicts the various states the request to provision the SDI can take, from the initial drafting by the requester to complete provisioning or cancelation of the request.

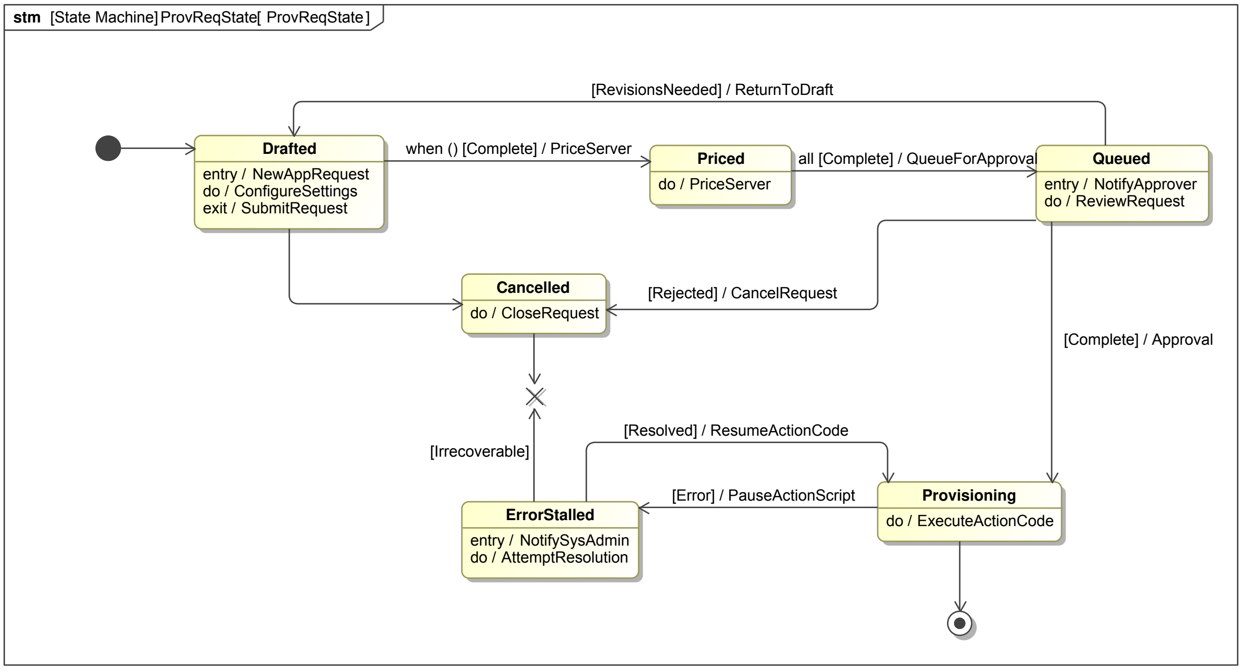

Figure 4.19: Possible states of a provisioning request.

The "Drafted", "Queued", and "Canceled" states are managed through interaction of the *ActionCode* with the ticketing system, while "Provisioning" and "ErrorStalled" are managed through interaction with *Infrastructure* and *ManagementTools*. The "Priced" state could be implemented within the *ActionCode* or in an externally interfaced system. If "Provisioning" is successful, the system is completely deployed, and the process ends (ideally with appropriate logging and notification to the requester). If it is unsuccessful (or canceled), the process terminates with logging and notification, and any actions taken, such as provisioning of storage, would be rolled back.

4.4.3 Data Perspective

Logical Data Model

The Logical Data Model (LDM) in the MBSAP captures the general information categories of a system. A partial LDM for the SDI is shown in Figure 4.20, focused on the main elements involved in the server provisioning use case, as a complete LDM covering all potential automation use cases would be extremely large and complicated, potentially touching on every subsystem in the IT environment. For example, the representation of the concept of location, assigned here to the

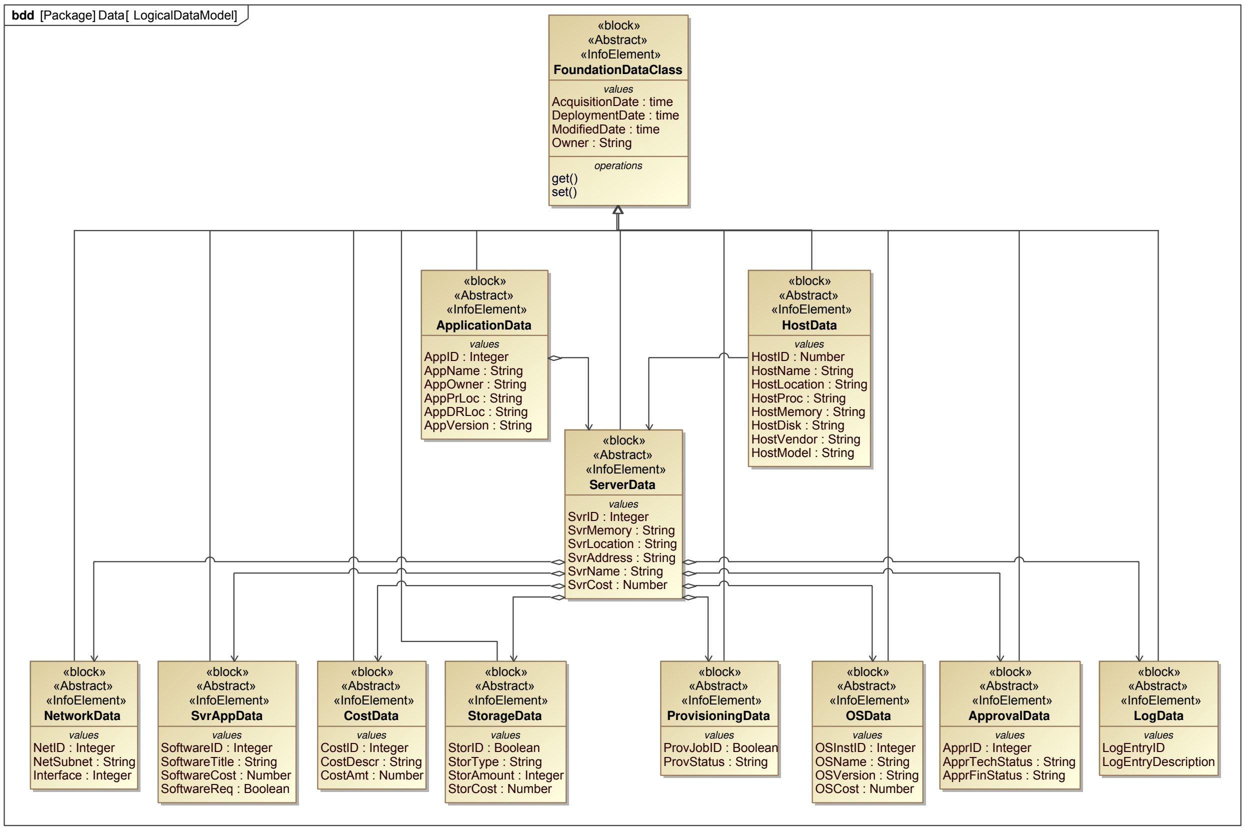

Figure 4.20: Logical Data Model for the server provisioning use cases.

host object, could be expanded to include a nested set of additional objects (rack units in a rack in a room in a data center, etc.). The specific APIs leveraged via the code and the subsystems acting on by the code will also 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 LDM shown is specific to the server provisioning process and would need to be elaborated to address other use cases.

4.4.4 Services Perspective

Rather than implementing complex tasks directly within *ActionCode*, commonly accessed functions can be extracted into services that can be managed separately and reused in different use

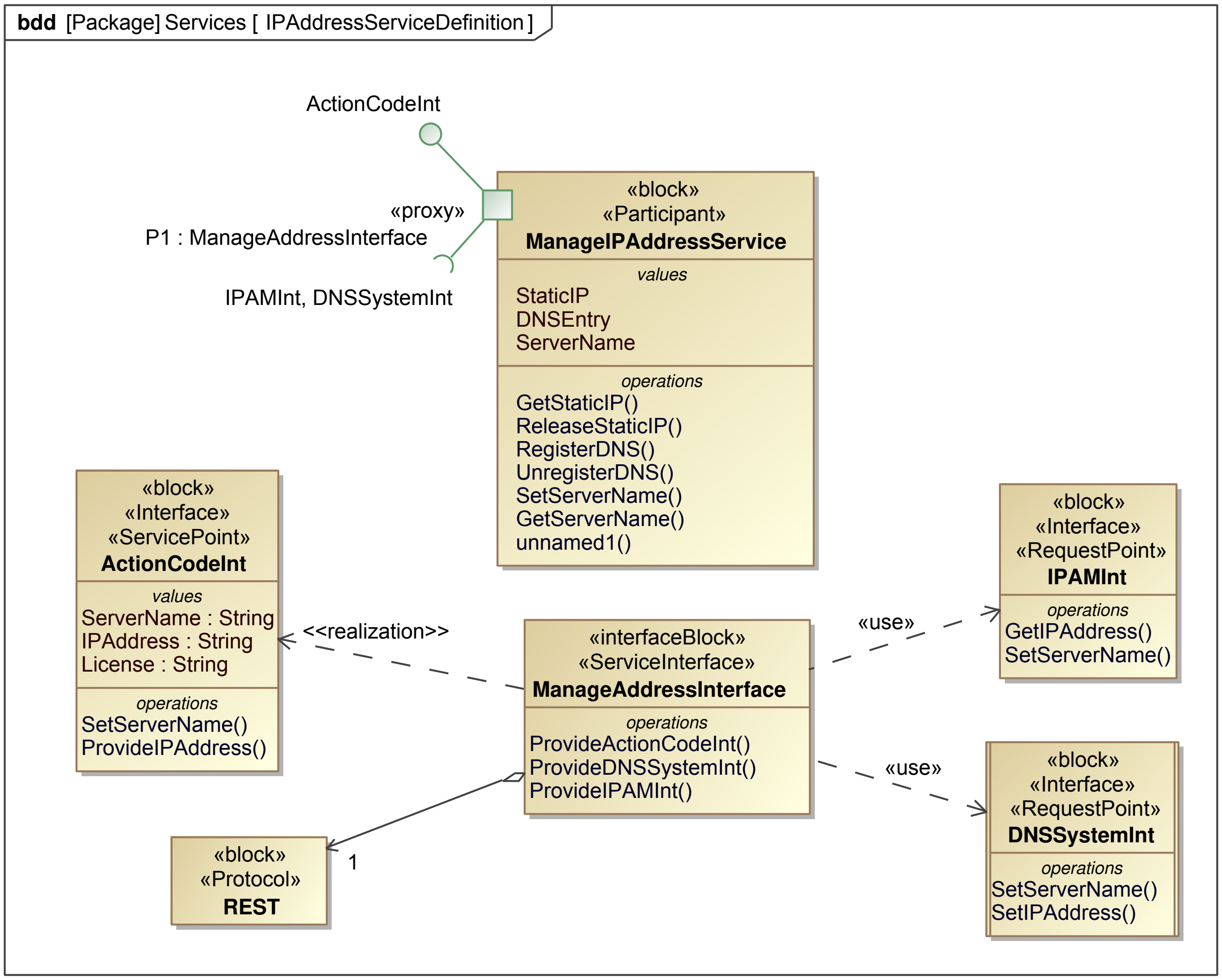

Figure 4.21: Service for the management of IP addresses, used by the Provisioning use case among others.

cases. For example, many actions involving servers including provisioning and de-provisioning, but also moving them between data centers and other events like disaster recovery, involve the management of network IP addresses. This includes the process of acquiring an IP address, but also updating any systems that need to know the IP address has changed, in particular DNS. Although the product vendors in question may expose specific services from their individual tools, the coordination of functions across tools makes this a potential candidate for the development of a custom service. A possible service definition for this function is shown in Figure 4.21.

Figure 4.22 shows the coordination of multiple services (including the one above) with a service contract established.

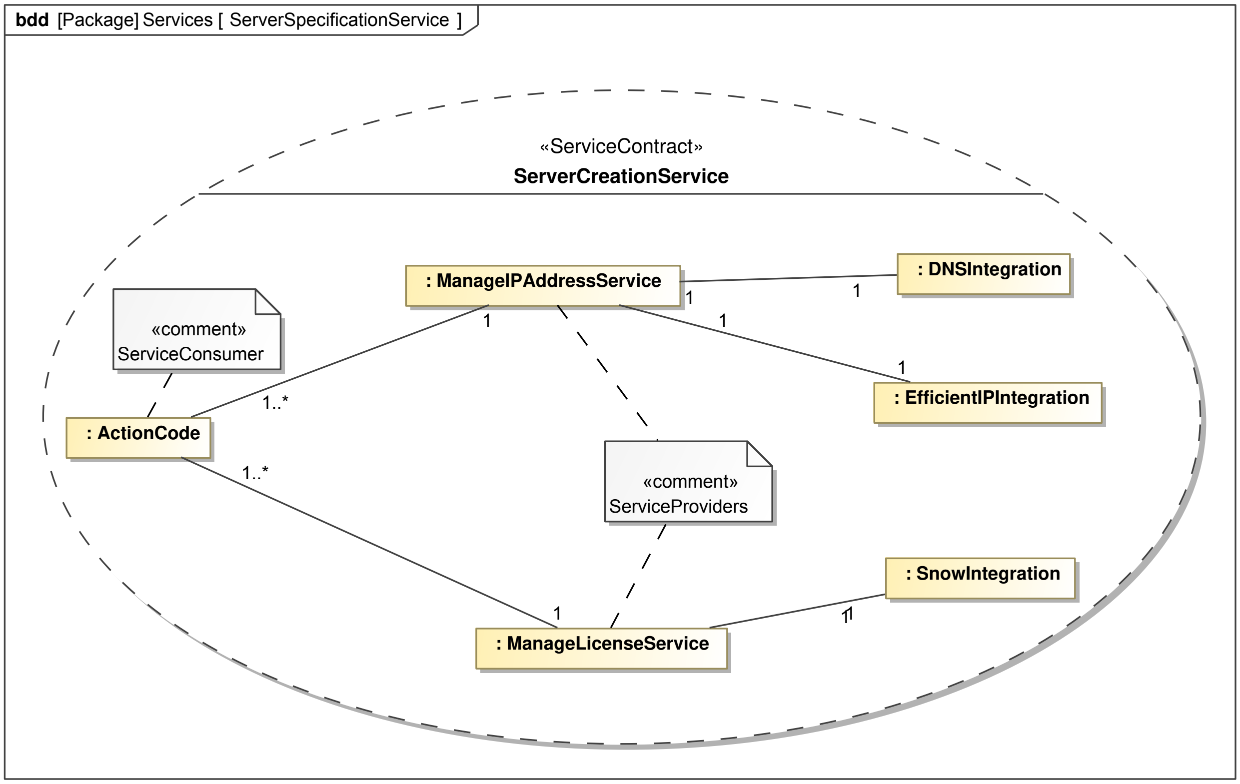

Figure 4.22: Service for the management of IP addresses, used by the Provisioning use case among others.

4.4.5 Contextual Perspective

The architectural layers for the SDI are shown in Figure 4.23. The foundation of all IT environments is the logical and virtual infrastructure, which is what the *ActionCode* takes action on. The triggers for those actions are shown at the top of the conceptual model, with the core components of the SDI Management System shown in the center.

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 Representational State Transfer (REST) code stored under version control in the organization's GitLab environment as the "action code". Additionally, the modeler could define the

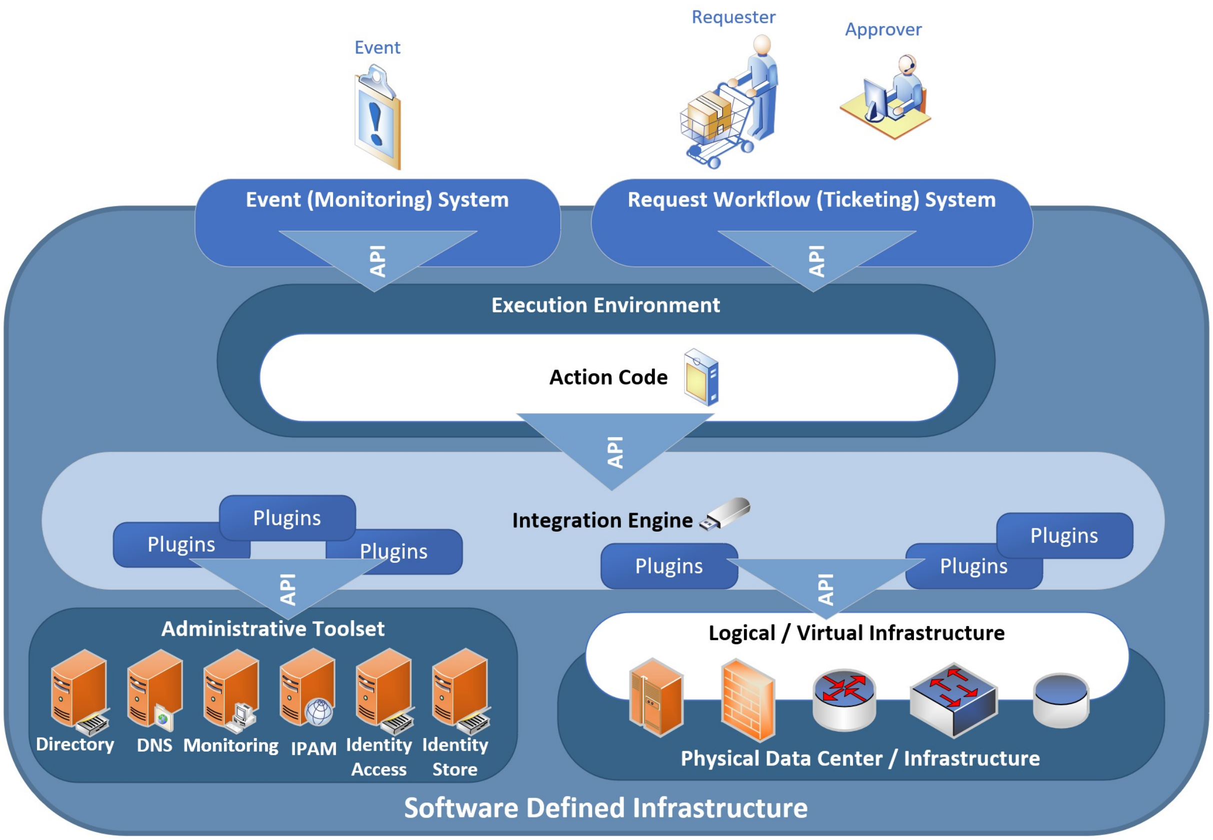

Figure 4.23: SDI architectural layers.

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 4.24, which highlights the ability to continue to develop systems from conceptual models into detailed designs suitable for engineers to build.

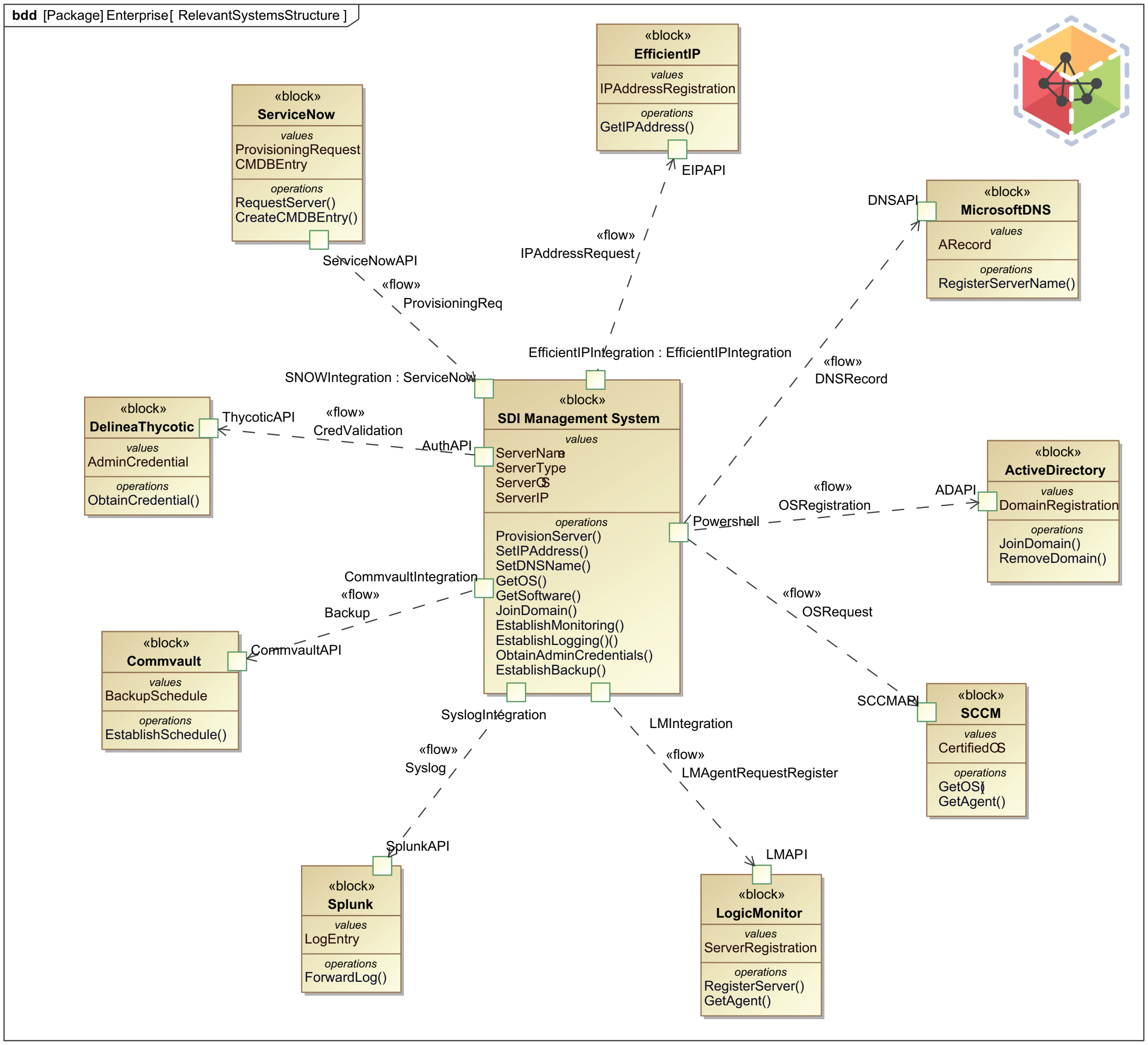

Figure 4.24: Product-specific design example for the SDI Management System.

Chapter 5

Discussion

5.1 Introduction

In trying to answer the original research questions, it becomes apparent that there are really two fundamental issues that need to be addressed to determine whether to implement SDI:

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

Chapter 3 addresses how IT leaders should begin to identify and prioritize activities that could be automated to the extent possible. From a baseline point of view, this includes work items that occur *frequently* enough to justify the investment in automation and that are *repeatable* enough that the execution sequence is predictable and, therefore, 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 work where the effort to automate can be justified. For example, even a unique task that has to be implemented on hundreds or thousands of devices can be worth automating even once to ensure consistency and accuracy of the 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, as well as the scalability of the existing staff.

Chapter 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 binds them together through APIs and provides the environment in which the automation code can execute.

These will be combined in the format of a business case to justify SDI and a road map to execute it in the discussion below.

5.2 Model Validity

A useful point of analysis for the hybrid simulation model would be the steady-state point reflecting the inputs that lead to near-full utilization with no queue growth. At this point, any increase in workload will cause queuing of work to grow, while below this point, there is still slack in the system. From this point on, the impact of the specific use case automation is much clearer to demonstrate.

Unfortunately, gradient-based methods are completely unable to find a minimum to the objective function in Section 3.6.3. Similarly, derivative-free approaches (pattern search and particle swarm) were unable to find an optimized solution that meets the constraints of known total duration times, near full utilization, and no change in queue depth. This can be due to one or more of the following:

- **Flat and multi-valued objective landscape**. It is possible that many different parameter combinations can yield very similar performance. This phenomenon of equifinality creates an almost "flat" region in the objective function. In such regions, small changes in interarrival rates or service times produce negligible changes in the observed performance metrics. As a result, gradient-based methods lack a clear slope to follow, while direct search methods may struggle to detect a meaningful improvement from one combination to another.
- **Nonlinearity and ill-conditioning**. Queuing systems with feedback loops (like rework and stop / start returns to queues) and multiple interacting stochastic processes tend to have highly non-linear mappings from input parameters to output. Non-linearities can produce multiple local optima or plateaus in the search space. When gradients vanish or become unreliable due to such non-linearity, optimization methods have trouble identifying the direction or region where an optimum exists.

- **Stochastic noise and simulation variability**. The stochastic variability of interarrival and service times means that each simulation run can yield different performance outcomes, even for the same set of parameters. This noise can obscure the relationship between parameters and output, making the objective function "noisy" and further flattening the optimization landscape. This noise can mask small but significant differences in performance, leading to difficulty in distinguishing between nearly equivalent solutions.
- **Conflicting or overdetermined constraints**. The constraints used (matching the known total duration times, maintaining near-full utilization, and keeping queue depths constant) may impose conflicting requirements on the system. If the constraints are too strict or if the underlying model is over-parameterized relative to the available data, the feasible solution space might be very narrow or even inconsistent. In such cases, the optimization algorithm may not find a unique, well-defined optimum that satisfies all constraints simultaneously.

Based on the theoretical foundation and the practical experience of the stakeholders in the team, the simulation model itself appears to be structured appropriately. The difficulty lies in calibrating certain parameters when the system is tuned to operate at nearly full utilization and maintain stable queue lengths. In such conditions, many different combinations of parameters could produce very similar aggregate outputs. This indicates an identifiability issue rather than a flaw in the simulation structure. The problem of determining the "best" parameter values under your specific constraints is likely not well-formed in the sense that the mapping from the dependent variables to the performance outcomes is not one-to-one. Instead, there exists a region in the parameter space where many combinations yield similar output values. This makes it difficult to find a unique solution using standard optimization techniques.

As a result, rather than using the simulation model to predict the threshold *inputs* to obtain an equilibrium *output*, the only way to address the validity of the model is to use it to predict the impact on *outputs* of automating a specific use case (based on a predicted change of *inputs*) and then compare the predictions to the results of real-world implementation of this automation. For example, assume that a team identifies based on an assessment of volumes to complete a certain work item:

- Partial automation of that function is possible that reduces the required service time by 50%.
- This item of work represents 10% of the volume of request tickets and project tasks within the team.
- Therefore, in order to predict the outcome of the model, the two associated work generators should be reconfigured to use two different formulas to determine the required service time, with 10% of the entities created that reduced the required service time.

The simulation can then be run to predict the overall impact on the team's utilization and queue depth. Based on the results of the sensitivity analysis in Chapter 3, one would expect the impact of this single automation activity on the queue depth to be fairly small (reducing the Required Service time for a small percentage of work); however, in combination with other automation efforts, the aggregate effect can certainly accumulate to a substantial reduction in queuing (or utilization).

Following automation implementation, measurements of these metrics can be compared to prediction to determine how closely the model represents the actual system. Of course, any deviation of actual resulting outputs from predicted could be a result of several factors aside from the validity of the model (e.g., faulty estimation of the change in inputs from the automation or poor implementation). Therefore, the model can only be fully validated by repeated iteration of this "predict-implement-assess" cycle.

5.3 Example Business Case for SDI

The focus of this analysis will be on the creation of the initial SDI framework to enable the canonical example of server provisioning used throughout this investigation.

5.3.1 Objectives and Scope

For each use case identified as the initial driver, 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 at least six separate technical teams, several of which touched on the request more than once as the request was routed to completion. From Section 3.6.3, each task can take 3 to 4.5 hours on average to complete, implying 18 to 27 hours of effort. This seemed high to the stakeholders for the tasks involved, so a lower number of 10 hours was used for the 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 from any technical team, with completion in 8 hours (assuming adequate server and storage capacity is 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.

5.3.2 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 (GitLab was chosen), as well as the creation of processes for the testing and review of automation code, and 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 of different types, for example, web servers, application servers, and a database), include registering the servers in monitoring and backup tools, and expand the different data center environments where servers could be provisioned.

5.3.3 Cost Analysis

The reference architecture outlined in Chapter 4 provides a framework for understanding the technologies / products, and therefore the costs, required for an organization to enable SDI. The greater the degree of support for relevant APIs across the deployed infrastructure base, in other words, the degree to which the organization's products can be connected into something resembling Figure 4.24 – the lower the initial investment will be. Any key components that do not support automation represent products that will need to be upgraded or replaced to fully enable a workflow. In particular, three subsystems are critical in order to provide any centralized SDI: the monitoring system, the ticketing system, and the execution environment itself. All other subsystems could be manually worked around until they can be upgraded or replaced, but with a reduction in the potential benefits of automation.

Initial Investment

In the case study organization, no new hardware or software was required, which reduced initial costs. All of the needed subsystems were in use prior to the effort, simply used in a manual fashion. 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 GitLab 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 that an annual cost to maintain the automated provisioning code is approximately 40 hours a year for a medium-level full-time DevOps engineer, after transitioning from the third-party DevOps engineer. The organization estimated an annual fully charged salary based on a market assessment and also priced an ongoing service for the maintenance of the code.

5.3.4 Benefits Analysis

One of the primary benefits from the creation of the hybrid simulation model outlined in Chapter 3 is the ability to use it to estimate the impact of automation and, from this, derive benefits that have a strong basis in theory. For any given use case, IT leaders can estimate the impact automation would have in terms of two easily measurable benefits (either tangible or intangible) most commonly associated with the DES simulation:

- *Reduction in the arrival rate of work*. Will this automation reduce the number of tickets or tasks and, if so, by how much? This could be a reduction or complete elimination of a percentage of tasks or tickets to complete that use case.
- *Reduction in the required service time*. Will this automation reduce the amount of time a staff member needs to spend on a particular work item, and if so, by how much? For example,

automation that retains a “human in the loop” could significantly reduce the time required but would not eliminate it if a staff engineer must review and complete any follow-up work.

- Note that even automation of admin tasks can have a positive impact on relevant metrics for operational ticks and project tasks by reducing contention for resource time.

In addition, there are several areas that automation can target associated with the SD simulation. These can be much harder to estimate and are more associated with intangible benefits.

- *Reduction in the time to respond.* Will this automation reduce the time required to respond to a request or event, and if so, by how much? In the case of system downtime where the cost of failure is well understood, this could result in real loss avoidance. In the case of user requests, this may be directly tied to employee productivity (the cost of which can be estimated) or could be associated with customer satisfaction.
- *Reduction in the error rate.* Will this automation reduce the amount and / or severity of errors, and if so by how much? For example, high-quality automation could reduce the potential for incidents as a result of change, which as modeled, reduces the amount of “new” incidents.

Tangible Benefits

In general, automation of server provisioning was estimated to “save” the equivalent of two equivalent full-time engineers’ time; however, these were not directly realizable as a financial saving (or cost avoidance) as those hours were spread across many individuals on six different teams. Instead, these represent hours available to apply to other tasks within those teams. The simulation model in its current version could not be used for a single prediction in this case, since it is limited to a single team; a more complex multi-team model would be required for that purpose. However, the model could be used separately for each team involved if reconfigured with the appropriate number of servers and work volumes. Server provisioning times were reduced to less than two 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. Finally, the quality of the provisioning process improved, eliminating instances of non-standard server naming, missing required software installations (to enable patching or security, for example), and servers not configured for monitoring or backups. They also expected to reinvest the engineering time saved in other deferred work areas, both operational and project. Note that some organizations (including the case study organization) will discount intangible benefits and perhaps even exclude them from the bottom-line analysis.

ROI and Financial Metrics

As the 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, and the use of financial tools such as Net Present Value (NPV) was unnecessary to justify the effort. However, where this is not the case, and procurement is required, the costs of any new or upgraded infrastructure or tools to support automation should be defined over a suitable time horizon (commonly five years), including both initial investments and annual maintenance or subscription costs. The accounting rules for amortization and depreciation can vary by organization, and IT leaders are highly encouraged to consult with their financial teams on how to represent them. These costs and benefits should then be compared with the "null hypothesis" of business as usual (without automation).

5.3.5 Assumptions and Constraints

In the case study organization, the provisioning of servers to remote data centers (in hospitals) was excluded from the scope, due to the lack of underlying infrastructure with exposed APIs, in turn, a result of insufficient historical capital investment. Furthermore, the high degree of variability between hospital data centers (in terms of infrastructure and applications) reduced the potential for repeatability, which is crucial to the viability of automation. The goal is to consider this in the future, when routine refresh, application consolidation / standardization, and other projects may have improved the potential for automation.

The following additional assumptions were made:

- The affected teams would be able to provide adequate training to ensure adoption by IT personnel.
- There would be no major changes to tools or infrastructure during implementation, with stable or predictable infrastructure demands during or immediately after automation rollout.
- Management support and resources (budget, personnel) for automation would remain consistent.
- The effort would heavily depend on the availability of the vendor or consulting for initial implementation in order to meet project timelines.

In addition, the following constraints were identified:

- Organizational policies (e.g., security, data privacy) limited initial tool choices and lengthened the time to select any new tools; therefore, a tool already purchased through a product bundle was selected as the foundation for automation.
- Budget and resource limits constrained the extent of automation.
- Legacy systems did not fully support automation, requiring additional effort to find workarounds.
- Vendors were not engaged to extend automation to the application level due to cost and availability of resources.

5.3.6 Risks and Mitigation

Technical risks related to the provisioning process were identified, focusing on the potential for the automation code to incorrectly assign network names or addresses that duplicated (and therefore conflicted with) existing production systems. The potential to over-provision resources could also cause a performance impact on production systems. These were mitigated by a cross-functional analysis of potential errors and the incorporation of adequate error-checking logic and processes such as peer review.

More generally, the potential exposure of administrative credentials through the automation code was a key initial concern, particularly with the selection of a cloud-based code repository for the project. Incorporation of the organization's existing commercial Privileged Access Management (PAM) system was a key mitigation step.

The dependence on automation tools for the provisioning process introduced a new potential single point of failure. This was mitigated through two mechanisms: first, through clear documentation of the process, available in a shared location so that it could be completed manually in the event the automation system failed; second, by ensuring the resiliency and recoverability of the automation system itself in secondary data centers.

Throughout the project, risks were reduced by performing limited initial proofs of concept, progressing from single-server to multi-server provisioning using staged rollouts, and slowly expanding the functionality provided by limiting provisioning to a single central data center initially and expanding to others as confidence in the automation grew. Furthermore, the initial utilization of the automated provisioning process was completed under human supervision and with notification to stakeholder teams through the operational change management process.

From an organizational point of view, there was an initial concern of resistance to change, where staff would be reluctant to change to manual methods. Although this did not turn out to be the case, the potential for a negative impact on staff security should be considered in any automation process. Additionally, at the start of the effort, there was a key lack of in-house expertise with new automation tools, which required initial and ongoing training to overcome, as well as an increase in cooperation with vendor partners.

Financial risks should always be considered, especially with regard to the potential for budget overruns due to unexpected costs for licenses, professional services, or additional infrastructure. However, in this case, these were minimal as the scope was selected in a data center where the base technical investments had already been made. In that context, organizational leaders considered the possibility of a "benefit shortfall" where the improved responsiveness and productivity gains were less than projected, but the risk was deemed low as the required financial investments were negligible.

5.4 Roadmap for SDI Implementation

- Identify specific use cases and quantify the anticipated benefit before incurring costs. Note that these benefits may include hard savings (elimination or avoidance of cost) or quality attributes such as improved quality or timeliness, as outlined in 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. Accept that the necessary data may not be available initially and consider focusing first on changes to the process or tools to provide that data.
- Identify the underlying infrastructure and management tool subsystems needed to automate these use cases and determine their starting level of support for automation. Not every organization has implemented solutions with appropriate APIs. Inventory any areas requiring an upgrade or a refresh to enable automation.
  - 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 as preconditions for the 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. In particular, focus on tools that may already exist within the environment, perhaps purchased through a bundle with

other products (for example, the case study organization adopted the VMware product that they already owned). Although the focus of this research has been on the use of automation on-premises, consideration of an environment-agnostic tool capable of functioning both on-premises and in the cloud should be made. In the case study organization, recoverability of the SDI has had to be completely rethought as the organization shifted its disaster recovery environment to a public cloud provider. 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. Product development processes and tools are not trivial for an organization without this competency to adopt. 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). Leaders must recognize that having skilled individuals is not the same as having a process-driven organizational competency.

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

5.5 Findings

5.5.1 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. This work is the first to address this question for on-premises data centers in a vendor-agnostic manner.

"Productized" examples of the core 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, used by the case study organization). 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 (especially external systems as their vendors evolve them over time). 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.

5.5.2 Research Question 2: Under what conditions should healthcare providers implement on-premises SDI?

This question is partially explored in Chapter 3 and is supported by existing research described in Chapter 2. The qualified answer to the specific question "should healthcare providers implement on-premises SDI *at all?*" is "yes", insofar as it is related to identifying clear potential benefits. However, the question of whether the general investments required to *build* SDI itself are worth those benefits given the constraints of healthcare provider IT is subject to other factors and the target use cases for automation. 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 it needs to perform that function. This work is the first to provide a guide for IT leaders in identifying and prioritizing automation opportunities, as well as a mechanism for predicting the results of those efforts in the form of the hybrid simulation model.

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

The synthesis of the completed work in Chapters 3 (addressing priorities) and 4 (that addresses components and prerequisites) enabled the creation of the business case and the roadmap proposed earlier in this chapter for implementation of on-premises SDI by healthcare providers. This work is the first to provide a guide to IT leaders in justifying and implementing automation of their work within their on-premises SDI.

5.6 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 of whether it is worth the time and financial investment of healthcare provider organizations to build it in the first place remains open. Of course, this question can only be answered in the context of each individual organization, including their starting technical and process state, as well as financial goals and constraints.

It appears clear that a full-scale shift to the “New School” technologies and processes outlined in Section 1.1 is unrealistic for an IT organization of healthcare providers that does not already have an internal development capability. Instead, IT leaders should focus on an incremental transition being justified on a use case by use case basis, with gains “standardized” before tackling the next use case in accordance with Deming’s “Wheel of Continuous Improvement” (or "PDCA cycle") [125]. This implies a gradual shift from a near complete reliance on manual administration to an increasing leverage of automation. Given a high degree of infrastructure and application standardization and continued expansion of automation, an organization may eventually reach an inflection point where these efforts make sense to accelerate and where the benefits are perceived by the organization as worth the investment, but this is not the starting posture for many organizations. Getting to that point takes persistence, and in the interim, most IT leaders will need to maintain a hybrid approach to work management that combines manual and automated completion to varying degrees.

Chapter 6

Summary, Implications, and Future Work

6.1 Summary

The motivation for this research was described in Chapter 1 and narrowed to a focus on non-profit healthcare provider IT organizations characterized by a high degree of dependence on COTS applications deployed in on-premises infrastructures and with a high reliance on manual administration of technologies and work activities on them. This situation is contrasted with that of development-oriented IT organizations that heavily leverage public cloud environments, where heavy use of automation is made to complete work. The question is posed of whether automation makes sense for the target IT organizations and, if so, how those leaders should consider making that transition.

The relevant literature is explored in Chapter 2, including operational and project-based work management processes and their elaboration through both queuing and systems dynamic simulation models for both descriptive and predictive purposes. These references also justify the coupling of the models into a hybrid system with better explanatory and predictive power. Following these topics, the literature discussing consideration of infrastructure in general (and IT infrastructures) as complex SoS is explored. The literature associated with various discrete forms of infrastructure automation (such as SDN) are discussed as components of an overall architecture. The utility of systems engineering tools in the design and documentation of infrastructure, including SysML and MBSE, is justified, along with their use in creating a reference architecture for automation.

Chapter 3 is focused on the development of two simulation models, with the goal of better understanding *what* IT functions should be automated. The SD model is first elaborated using Vensim, explaining the base structure of the different types of work through diagrams, and expanded on with the dynamic influencing factors (such as “management pressure”). Next, the DES model is elaborated with work “generators”, queues, and “servers” (individual engineers) receiving and completing work in accordance with the mathematics of queuing theory. The models are then tied together in a cycle, where the output of the DES model is fed back into the SD model, and the results of that simulation are fed back into the first. This can be automated and iterated with configurable simulation durations and numbers of iterations. A more detailed discussion of the model inputs and uncertainties, and attempts to validate the models are presented in the following sections. A brief discussion follows regarding how a multi-team simulation could provide more realism to the simulation output. Finally, the chapter concludes with an outline of the target areas for automation indicated through the models and their outputs.

In Chapter 4, the discussion is changed to focus on the technical dependencies required to support the automation on-premises of common IT processes through the development of a reference architecture for SDI. The tools of systems engineering (SysML and MBSE) are leveraged for the purpose, following the MBSAP methodology to develop the architecture of Logical Viewpoint, allowing it to remain product- and tool-agnostic. Additionally, the use of Cameo Systems Modeler to realize the MBSE model allows the creation of a "digital thread" linking the system requirements through all of the artifacts at every level of abstraction. At the Logical Viewpoint level, the architecture is narrowed to focus on the requirements of the server provisioning process for two reasons: first, because the architecture would grow extremely large, and second, to illustrate that the successive expansion of automation will correspond to the affected infrastructure in lockstep with the use case(s) being automated. Even with this focus on a single use case, the artifacts discussed for each perspective are illustrative and can be greatly expanded. The artifacts provided align with the solution as implemented by the case study organization.

Chapter 5 begins by addressing how IT leaders should approach developing the business case for automation and the plan to implement it. The validity of the simulation models is briefly discussed. The chapter then revisits the three research questions and answers them in the context of the preceding chapters. The overarching conclusion of the research is that there is a place for on-premises SDI, but it should be approached from the point of view of individual use cases and prioritized according to the anticipated costs and perceived benefits to enable each. Finally, a list of peer-reviewed papers is provided on the basis of this research.

6.2 Research Contributions

The hybrid DES and SD models built here address two shortcomings in the literature related to the management of work in IT. At least some teams have responsibility for both operational tickets and project tasks, and they are subject to the short-term effects of queuing theory and long-term dynamic effects. For these organizations, the hybrid model presented in this research can enable IT leaders to predict the impact on their outcomes based on changes from other effects than automation, including the hiring of staff, the outsourcing of work, or the effect of a merger or acquisition. Essentially, any event that changes the volume of work, the composition of that work, the duration of work, or the availability of staff can be predicted with relevant modifications to the models.

This research provides a model-based approach to guide the prioritization and justification of automation in use cases generally, with an eye toward their implementation in on-premise environments. Any organization with a reasonably good record of past work completed (tasks and tickets) can mine that information to identify activities that frequently happen as a first cut of priorities to consider for automation. Although this has been focused on IT and a particular type of IT team, the framework here can be extended to other functional teams elsewhere in an organization to guide the selection of opportunities for broader "digital transformation". While modification to the processes modeled may be required, the division of work between requests, incidents, project tasks, routine maintenance, and administrative work is ubiquitous; changes to the relative volume of each time and the service times associated with them may be all that is needed to apply to an entirely different business area. In combination, the above methods contribute to systems engineering research by providing a more realistic method for analyzing the workflows of teams driven by both operational and project work and, by extension, enabling better designs for systems intended for these environments.

Previous information has been limited to addressing how certain technologies or products of a particular vendor can be leveraged in a "software-defined" manner to enable automation. This research provides a reference architecture for on-premises software-defined infrastructure that addresses the broad set of services commonly provided by cloud service providers. In addition, this research provides a roadmap and business case framework to support the build-out of an on-premises software-defined infrastructure. The case study organization used these tools to successfully complete their SDI and DevOps journey. In addition, the use of MBSE and SysML for the design and documentation of the RA demonstrates the applicability of the tools and methods of Systems Engineering methods to IT, and particularly IT infrastructures, where the introduction of hardware systems and logistics often confounds methods commonly used in software system design [126].

Cloud-based, development-focused organizations have adopted DevOps methods and tools in part because the costs to implement in terms of investments required to expose the necessary APIs as well as those to build the necessary skills and processes are relatively low. The perceived value in that scenario does not need to be particularly high and may not require any formal justification. This is not true for IT organizations focused on on-premises systems. In combination, this research can potentially contribute to an acceleration of the adoption of DevOps methods and tools to a broader set of IT organizations, specifically those highly dependent on on-premises infrastructure.

6.3 Implications

As mentioned throughout this work, the models developed here can be broadly applicable beyond healthcare providers, wherever there are similar industry characteristics that prevent an organization from fully migrating to the public cloud, including a high reliance on COTS applications, preference for capital funding, and a heavy regulatory environment (especially with regard to data sensitivity). A strong orientation towards ITIL-based organizational management and waterfall project management, and skill-based organizational structures are also common characteristics. This can include large-scale financial services, defense, and manufacturing enterprises.

Changes in the vendor marketplace will continue to impact the importance and adoption of SDI, whether on-premises or in the public cloud. As an example, the continued shift (at the time of this writing) of pricing models to subscriptions, even for on-premises hardware and software, cuts against the traditional preference of for-profit healthcare providers for capitalizable purchases and will tend to make the consumption of cloud services more competitive. Any increased shift to the public cloud by IT organizations in the industry will help overcome residual technical limitations to the adoption of SDI, and potentially serve to increase the importance of adopting DevOps practices in the industry. The models and practices described in Chapters 3 and 5 remain useful for organizations making this transition.

Just because an organization is heavily invested in the public cloud does not mean it will stay there. In a recently acknowledged trend, many organizations that were "born in the cloud" or moved to a "cloud-first" architecture have learned that certain workloads – particularly AI workloads – can be prohibitively expensive when billed on a consumption basis. As a result, some organizations are beginning to move toward the "repatriation" of workloads from the public cloud back to on-premise data centers. These organizations, many of which have mature DevOps practices, will have to determine how to enable these practices on premises. The reference architecture in Chapter 4 and the tools in Chapter 5 can be helpful for organizations that are making the transition back to the data center.

6.4 Limitations

Models are, of course, approximations, but (we hope) useful ones. The following are several areas where the proposed models have limited applicability.

Hybrid models can allow for a closer study of the interactions between operational tickets and project tasks within the same team. For example, operational tickets are completed from the time they are identified (request submitted or incident alerted) *forward*, while project tasks are ideally completed sometime before (or *backward* from when) they are scheduled to be completed (although they can extend beyond that). It is likely that the dynamics of escalation are different for the two - the priority of a ticket increases over time until it reaches a maximum (unless, as a critical ticket, it starts that way), while the priority of a task increases the closer it gets to being on the critical path as well as to its due date.

Data limitations are particularly acute within the simulation models and the case study organization more generally. With regard to the DES model, the values for service times of tasks and tickets, the interarrival times of project / maintenance and administrative tasks, error rates, and actual team and individual utilization rates are all based on reasonable estimates by stakeholders of the case study team. The adoption of best practices and tools for project management, as well as the adoption of a time accounting solution, can greatly improve the accuracy of the simulation by replacing some of these estimates with actual values. In the SD model, all dynamic equations are based on estimates from stakeholders as these are much more difficult to validate; there is no system (as yet) to determine the values of "managerial pressure" or its impact on, for example, service quality. Additionally, the main dynamic factors in the SD model have been designed in a way to prevent runaway causal loops over multiple iterations – useful for model stability but not necessarily true to life.

The simulations currently represent work management within a single team, and as a result, the prediction of any benefits from automation using the model is limited to its impact on that team alone. From the standpoint of the percentage of work that any team is tackling, this is often sufficient. However, it does not begin to address the issues associated with the relatively small but significant percentage of work items that require interaction and dependency between multiple teams. The issues associated with a work item being constantly re-queued (if processed between teams in a serial rather than parallel manner), as well as the potential for any given work item to have a different relative priority in different teams, can significantly extend the overall time to complete the work. In addition, the introduction of multiple teams likely introduces new dynamics around coordination and escalation that are not seen when work is contained within a single team. The ability to predict the impact of automation that crosses teams, such as the provisioning example used throughout this study, on key metrics such as overall completion (sojourn) time depends on the expansion of the model to accommodate these new dynamics.

The coupled simulations represent the work of a team responsible for scheduled and unscheduled work. This is true for some teams in some organizations, but not universally. In some cases (even within the case study organization), there are teams organized to separate responsibility for project engagement from operational support. While the model – and in particular the DES model – can accommodate this through manipulation of the various interarrival times into the work generators (e.g., by preventing the creation of project or maintenance tasks, or conversely, the creation of requests and incidents) – there may be residual dynamics in the models that limits their utility in these teams.

The time frame of each simulation in the hybrid model (10 days) is an attempt to balance between allowing the DES model time to initialize and the SD time to show some impacts and feed those back into the queueing system. This is an artifact of using two different modeling tools with different time scales, as discussed in Chapter 3. It is possible that the trade-offs required in the integration obscure the true dynamics of the system, and one of the alternative methods is necessary to accurately reflect them.

The built simulation models treat the required service time of all work items as having the same average duration and fitting to the same negative exponential distribution. This assumption may not be valid; in fact, the potential for project tasks, in particular, to be of longer duration (due to higher complexity or novelty) is indicated in the literature and may have a non-negligible impact on simulation outcomes. More generally, models are built in such a way that any server can be assigned a particular work item, regardless of its inherent characteristics (such as priority and complexity). However, it is common for more senior staff to be assigned more difficult work. This simplifying design decision is valid for the case study organization as the team's engineers were of similar skill, but this may not hold in other teams with wider variability in skill levels – the availability of the senior-most engineer could become the rate limiter of the system.

Many organizations have a long history of in-house software development and have migrated substantial portions of their workloads to the public cloud (or both). As a result, they may have developed strong capabilities around DevOps and automation. In these cases, the parts of the roadmap and business case related to employee skill development and process adoption may not be relevant. These organizations may already have a strong sense of the value of automating certain use cases and, in fact, may have a library of previously automated use cases. In these cases, Chapter 4 may be the only section worth reviewing. Finally, the unique challenges of repatriating certain workloads (such as Kubernetes containers or AI/ML) from the public cloud to on-premises data centers are not addressed, although the base architecture of a SDI should remain valid.

6.5 Recommendations for 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.

The evolution of the simulation model 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 and non-linear dynamics of work management are expected to increase significantly as the number of teams increases, and new dynamics related to the coordination of work across the teams are introduced. This would likely also introduce the need to consider the impact of additional theoretical frameworks, such as network theory and queuing network analysis. In addition, the DES model, in particular, should be further enhanced to allow changes to the number of servers in the model through parameterization instead of manual configuration. Finally, some input parameters (such as the Required Service Time) should be modified to allow the ability to support multiple values or distributions, which would improve the flexibility of the model to represent the effect of different automation use cases.

This research focuses on a specific 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 improvements in the other five categories, 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:

- Modeling the impact of the separation of 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, although this is dependent on the expansion of the simulations to address the interactions of two or more teams on a single ticket or task.

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. However, the ability to quantify the relative impact of these interventions prior to implementation and then compare the observed results would change the basis of the business case from more subjective to objective.

In addition, the influence of "managerial pressure" (and related upstream and downstream factors) on software-defined 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 SD models.

The modeling of different work types can be refined in several areas in future work. For example, the models in this work make the simplifying assumption that both the interarrival times and the required service times for project tasks and operational tickets are the same. As mentioned in Chapter 3.3.3, there are anecdotal claims in the literature that these arrival rates are different from those of operational tickets, but there is little quantitative justification for these claims. The models are built to enable testing these claims and comparing them with observational data. Another simplifying assumption is that project tasks and operational tickets can be considered similar in size and complexity, allowing the DES model to use a single exponential distribution to define the "required service time." The model can be easily modified to enable different distributions. There is no hard justification for doing so at this point; future research to quantify the difference (if any) between the two types of work would be fruitful.

The utility of simulation in the area of mergers, divestitures, and acquisitions is a potentially fruitful area of research. In particular, with some modifications to allow a simple modification of the team size, the hybrid models explored in this work could be used to predict the impact of integrating a new set of systems, including the volume of work and available resources, into the parent organization.

With regard to the on-premises SDI architecture, the research can take several directions. Clearly, 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 make 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. However, 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 required to make use of them – serve to limit both the speed of adoption and the 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.

6.6 Final Reflections

As with many long journeys, this one started with what I assumed to be a simple question: "What activities should my team automate, and how do I go about doing that?" I didn't understand at the time why it was so difficult for my team and our vendor partners to answer. Now I do. What it will take to enable ("how you do it") depends on what you need (in terms of technical components) to support the automation, which in turn depends on what function or use case you intend to automate. Whether the effort is worth it depends on the value derived from automating that particular function or use case. Automation of some use cases is more valuable than others, and whether you *can* automate something is often very different than whether you *should*. When I asked the question, there was no comprehensive and coherent source, academic or otherwise, to help IT leaders determine these answers. Now there is. I hope it helps my peers find the motivation to begin the process much more quickly.

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] J. Webster and R. T. Watson, “Analyzing the Past to Prepare for the Future: Writing a Literature Review,” *MIS Quarterly*, vol. 26, no. 2, pp. xiii–xxiii, 2002, publisher: Management Information Systems Research Center, University of Minnesota. url: https://www.jstor.org/stable/4132319.

[3] Y. Levy and T. J. Ellis, “A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research,” *Informing Science: The International Journal of an Emerging Transdiscipline*, vol. 9, pp. 181–212, 2006, doi: 10.28945/479.

[4] 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.

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

[6] 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/.

[7] HIMSS, *The Cphims Review Guide*. Taylor & Francis Group, Dec. 2021.

[8] 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.

[9] 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.

[10] 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.

[11] F. S. Glaiel, A. Moulton, and S. E. Madnick, “Agile Project Dynamics: A System Dynamics Investigation of Agile Software Development Methods,” Massachusetts Institute of Technology. Engineering Systems Division, Working Paper, Oct. 2014, url: https://dspace.mit.edu/handle/1721.1/103024.

[12] “Rational Unified Process,” Mar. 2022, url: https://en.wikipedia.org/w/index.php?title=Rational_Unified_Process&oldid=1078006369.

[13] 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/doi/abs/10.1145/1877725.1877730.

[14] 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.

[15] 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.

[16] 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.

[17] *ITIL 4: High-velocity IT*. TSO, The Stationary Office, Feb. 2020.

[18] J. R. San Cristóbal, L. Carral, E. Diaz, J. A. Fraguela, and G. Iglesias, “Complexity and Project Management: A General Overview,” *Complexity*, vol. 2018, Oct. 2018, doi: 10.1155/2018/4891286.

[19] 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.

[20] 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, doi: 10.1002/sdr.213.

[21] 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, doi: 10.1057/palgrave.ejis.3000256.

[22] 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.

[23] 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.

[24] 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, doi: 10.1002/sdr.377.

[25] 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.

[26] 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, doi: 10.1109/32.21738.

[27] D. N. Ford and J. D. Sterman, "Dynamic modeling of product development processes," *System Dynamics Review*, vol. 14, no. 1, pp. 31–68, 1998, doi: 10.1002/(SICI)1099-1727(199821)14:1<31::AID-SDR141>3.0.CO;2-5.

[28] 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, doi: 10.1177/8756972819832203.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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, doi: 10.1109/TSE.2004.1265735.

[34] 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.

[35] 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, doi: 10.1109/17.846792.

[36] 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.

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

[38] 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, doi: 10.1109/TNSM.2010.1009.I9P0321.

[39] 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, doi: 10.1109/ICSM.2001.972764.

[40] *ITIL Foundation, ITIL.* TSO, The Stationary Office, 2019.

[41] 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.

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

[43] 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, doi: 10.1109/INM.2015.7140506.

[44] 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.

[45] 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.

[46] 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.

[47] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A Survey of DevOps Concepts and Challenges," *ACM Computing Surveys*, vol. 52, no. 6, pp. 1–35, Nov. 2020, doi: 10.1145/3359981.

[48] M. L. Pedra, M. F. da Silva, and L. G. Azevedo, "DevOps Adoption: Eight Emergent Perspectives," Sep. 2021, url: http://arxiv.org/abs/2109.09601.

[49] R. E. i. C. Cloutier, "The Guide to the Systems Engineering Body of Knowledge (SEBoK)," Hoboken, NJ, US, 2023, url: www.sebokwiki.org.

[50] 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/stable/2640328?sid=primo.

[51] J. W. Forrester, *Principles of Systems*. Productivity Press, 1990.

[52] 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, doi: 10.1109/ICSMC.2001.971932.

[53] 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.

[54] T. K. Abdel-Hamid, "The dynamics of software development project management : an integrative system dynamics perspective," Thesis, Massachusetts Institute of Technology, 1984, url: https://dspace.mit.edu/handle/1721.1/38235.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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, doi: 10.1108/17410391311289541.

[60] 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. Springer International Publishing, 2016, pp. 69–81, doi: 10.1007/978-3-319-31295-8_5.

[61] A. Greasley, *Simulating Business Processes for Descriptive, Predictive, and Prescriptive Analytics*. Walter de Gruyter GmbH & Co KG, Oct. 2019.

[62] 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.

[63] 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 Conference (WSC)*, Dec. 2011, pp. 2710–2722, doi: 10.1109/WSC.2011.6147977.

[64] A. Borshchev, *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*. AnyLogic North America, 2013.

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

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

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] D. Ribes and T. A. Finholt, "The Long Now of Infrastructure: Articulating Tensions in Development," *Journal of the Association for Information Systems*, vol. 10, no. 5, May 2009, url: https://repository.library.georgetown.edu/handle/10822/557392.

[72] M. Morgan, T. Holzer, and T. Eveleigh, "Synergizing model-based systems engineering, modularity, and software container concepts to manage obsolescence," *Systems Engineering*, vol. 24, no. 5, pp. 369–380, 2021, doi: 10.1002/sys.21591.

[73] 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, doi: 10.1007/s10270-014-0407-3.

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

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

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

[77] 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.

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

[79] "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.

[80] 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.

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

[82] 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.

[83] M. G. Avram, “Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective,” *Procedia Technology*, vol. 12, pp. 529–534, Jan. 2014, doi: 10.1016/j.protcy.2013.12.525.

[84] K. Cresswell, A. D. Hernández, R. Williams, and A. Sheikh, “Key Challenges and Opportunities for Cloud Technology in Health Care: Semistructured Interview Study,” *JMIR Human Factors*, vol. 9, no. 1, Jan. 2022, doi: 10.2196/31246.

[85] R. Kent, B. C. Mohan, and K. Ryszard, “Enterprise adoption of cloud computing with application portfolio profiling and application portfolio assessment,” *Journal of Cloud Computing*, vol. 10, no. 1, Jan. 2021, doi: 10.1186/s13677-020-00210-w.

[86] D. Ray, “Cloud Adoption Decisions: Benefitting from an Integrated Perspective,” *Electronic Journal of Information Systems Evaluation*, vol. 19, no. 1, pp. 3–22, Mar. 2016, url: https://www.proquest.com/docview/1792068014/abstract/30D7AB0031F44BD8PQ/1.

[87] B. Astuto, M. ca, X. Nguyen, K. Obraczka, and T. Turletti, “A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, Jan. 2014, doi: 10.1109/SURV.2014.012214.00180.

[88] D. Kreutz, F. Ramos, P. Veríssimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, “Software-Defined Networking: A Comprehensive Survey,” *ArXiv e-prints*, vol. 103, Jun. 2014, doi: 10.1109/JPROC.2014.2371999.

[89] J. Sandobalin, E. Insfran, and S. Abrahao, “End-to-End Automation in Cloud Infrastructure Provisioning,” *International Conference on Information Systems Development (ISD)*, Sep. 2017, url: https://aisel.aisnet.org/isd2014/proceedings2017/ISDMethodologies/5.

[90] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, “A systematic mapping study of infrastructure as code research,” *Information and Software Technology*, vol. 108, pp. 65–77, Apr. 2019, doi: 10.1016/j.infsof.2018.12.004.

[91] J. Sandobalin, E. Insfran, and S. Abrahao, “On the effectiveness of tools to support infrastructure as code: Model-driven versus code-centric,” *IEEE Access*, vol. 8, pp. 17 734–17 761, 2020, doi: 10.1109/ACCESS.2020.2966597.

[92] “(3) Cattle vs. Pets in Infrastructure: Embracing Immutable Infrastructure for Speed and Scalability | LinkedIn,” url: https://www.linkedin.com/pulse/cattle-vs-pets-infrastructure-embracing-immutable-speed-jackson-6axkf/.

[93] “IEEE/ISO/IEC International Standard for software, systems and enterprise–Architecture description,” Nov. 2022, conference Name: ISO/IEC/IEEE 42010:2022(E). doi: 10.1109/IEEESTD.2022.9938446.

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

[95] 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, doi: 10.48550/arXiv.2209.14714.

[96] 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, doi: 10.1109/EDOC.2013.33.

[97] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, and D. Leaf, “NIST Cloud Computing Reference Architecture,” Sep. 2011.

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

[99] D. Bayyou and M. Dumlao, "Cloud Computing Reference Architecture from Different Vendor's Perspective," *International Journal of Emerging Technology and Advanced Engineering*, vol. 3, no. 11, Nov. 2013, url: www.ijetae.com.

[100] 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.

[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] C. Porch, G. Timbrell, and M. Rosemann, “Platforms: A Systematic Review Of The Literature Using Algorithmic Historiography,” *ECIS 2015 Completed Research Papers*, May 2015, doi: 10.18151/7217443.

[106] R. Sun, S. Gregor, and B. Keating, “Information Technology Platforms: Conceptualisation and a Review of Emerging Research in IS Research,” *Australasian Conference on Information Systems (ACIS) 2015 Proceedings*, pp. 1–17, 2015, url: https://eprints.qut.edu.au/199495/.

[107] M. A. Bach Tobji, R. Jallouli, H. Sadok, K. Lajfari, D. Mafamane, and H. Mahboub, Eds., *Digital Economy. Emerging Technologies and Business Innovation: 9th International Conference on Digital Economy, ICDEc 2024, Rabat, Morocco, May 9–11, 2024, Proceedings, Part I*, ser. Lecture Notes in Business Information Processing. Cham: Springer Nature Switzerland, 2025, vol. 530, doi: 10.1007/978-3-031-76365-6.

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

[109] 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.

[110] 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.

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

[112] 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, doi: 10.1109/RCIS.2015.7128911.

[113] 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.

[114] 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.

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

[116] 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.

[117] 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/.

[118] 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.

[119] 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, doi: 10.1002/sys.21644.

[120] 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.

[121] "Palm–Khintchine theorem," Sep. 2022, page Version ID: 1109210741. url: https://en.wikipedia.org/w/index.php?title=Palm%E2%80%93Khintchine_theorem&oldid=1109210741#cite_note-heymansobel-1.

[122] E. S. Enos and D. R. Herber, "Using Hybrid System Dynamics and Discrete Event Simulations to Identify High Leverage Targets for Process Improvement in a Skill-based Organizational Structure," in *2024 IEEE International Systems Conference (SysCon)*. Montreal, QC, Canada: IEEE, Apr. 2024, pp. 1–8, doi: 10.1109/SysCon61195.2024.10553565.

[123] R. R. Rhinehart, *Engineering Optimization: Applications, Methods and Analysis*. John Wiley & Sons, Mar. 2018.

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

[125] "PDCA," Feb. 2025, url: https://en.wikipedia.org/w/index.php?title=PDCA&oldid=1276283077.

[126] E. S. Enos and D. R. Herber, "Should IT Systems Engineers Adopt Systems Engineering Tools and Methods?" *IT Professional*, 2025, doi: 10.1109/MITP.2025.3548932.

Appendix A

Simulation Scripts

Matlab Simulink Script

The structure of the code is modular, with a main script (DESAutomation_main.m) calling multiple subordinate scripts sequentially as follows:

1. initializeSimulation.m sets base parameters for the simulation.
2. configureSimEvents.m sets up the SimEvents DES model.
3. runSimEvents.m calls the DES model and passes through the configuration data.
4. processSimOutput.m pulls the results of the DES model and begins the analysis.
5. analyzeUtilization.m determines the utilization of the individual engineers and the team as a whole.
6. analyzeTimeData.m determines the duration of the different work times - incidents, requests, etc.
7. analyzeQueueData.m determines the queue depth of the individual engineers and the team as a whole.
8. excelOperations.m writes the data from the DES results to an Excel file (sddatain.xlsx), for loading into the SD model.
9. triggerVensimSimulation.m calls a command script (sdcommand.cmd), which in turn initializes the SD model. This model automatically loads the data from sddatain.xlsx, and saves results to a second Excel file (sddataout.xlsx).
10. updateIterationParameters.m extracts the SD results from sddataout.xlsx and prepares them for the next iteration.

DESAutomation_main.m

```
%% DESAutomation_main.m
clear; clc;
```

```
% 1. Initialization: get base parameters and DES input history.
[baseParams, desHistory] = initializeSimulation();

% Define maximum iterations.
MaxIterations = 1;

for Iteration = 1:MaxIterations
    fprintf('Iteration %d\n', Iteration);

    % --- Pre-Iteration Cleanup: ensure any previous instance is unloaded ---
    if bdIsLoaded('iterateddes')
        close_system('iterateddes', 0);
    end
    pause(1);  % allow time for cleanup

    % 2. Configure simulation parameters for SimEvents (loads model fresh)
    simIn = configureSimEvents(baseParams);

    % 3. Run the SimEvents simulation
    simOut = runSimEvents(simIn);

    % 4. Process simulation output.
    simMetrics = processSimOutput(simOut, baseParams);

    % 5. Analyze additional signals.
    util_pred = analyzeUtilization(simOut);
    [IncTimePDFLambdaSim, ReqTimePDFLambdaSim] = analyzeTimeData(simOut);
    queue_pred = analyzeQueueData(simOut);

    % 6. Build the SDParameters vector using fields from simMetrics.
    TicketsPickedUpPerDay = simMetrics.TicketsPickedUpPerDay;
    TasksPickedUpPerDay   = simMetrics.TasksPickedUpPerDay;
    TicketsStoppedPerDay  = simMetrics.TicketsStoppedPerDay;
    TasksStoppedPerDay    = simMetrics.TasksStoppedPerDay;
    TicketsCompletedPerDay = simMetrics.TicketsCompletedPerDay;
    TasksCompletedPerDay  = simMetrics.TasksCompletedPerDay;
    BaseReworkRate        = simMetrics.BaseReworkRate;
    BaseIncFromChange     = simMetrics.BaseIncFromChange;
    BaseMgmtPress         = simMetrics.BaseMgmtPress;
    BaseFatigue           = simMetrics.BaseFatigue;
    BaseQueuedTasks       = simMetrics.BaseQueuedTasks;
    BaseQueuedTickets     = simMetrics.BaseQueuedTickets;
    BaseMgmtPreempt       = simMetrics.BaseMgmtPreempt;
    BaseCompleteTasks     = simMetrics.BaseCompleteTasks;
    BaseCompleteTickets   = simMetrics.BaseCompleteTickets;
    % --- New items: include the current error rate and service time ---
    ErrorRateVal          = baseParams.ErrorRate;
    ServiceTimeActVal     = baseParams.ServiceTimeAct;
    IterationVal          = simMetrics.Iteration;

    % Now build an 18-element vector.
    SDParameters = [TicketsPickedUpPerDay;
                    TasksPickedUpPerDay;
                    TicketsStoppedPerDay;
                    TasksStoppedPerDay;
```

```
                    TicketsCompletedPerDay;
                    TasksCompletedPerDay;
                    BaseReworkRate;
                    BaseIncFromChange;
                    BaseMgmtPress;
                    BaseFatigue;
                    BaseQueuedTasks;
                    BaseQueuedTickets;
                    BaseMgmtPreempt;
                    BaseCompleteTasks;
                    BaseCompleteTickets;
                    ErrorRateVal;
                    ServiceTimeActVal;
                    IterationVal];

    % 7. Write the SDParameters vector to Excel.
    excelOperations('write', SDParameters);

    % 8. Trigger external Vensim simulation and wait for completion.
    triggerVensimSimulation();

    % 9. Update iteration parameters and compute residuals.
    [baseParams, desHistory, residuals] = updateIterationParameters(baseParams, desHistory, []);

    % --- Post-Iteration Cleanup: unload the model completely ---
    if bdIsLoaded('iterateddes')
        close_system('iterateddes', 0);
    end
    pause(1);
end
```

initializeSimulation.m

```
function [baseParams, desHistory] = initializeSimulation()
    % initializeSimulation.m
    % This function initializes the simulation parameters and the DES input history.

    % Base simulation parameters and independent variables
    baseParams.BaseMgmtPreempt = 1;
    baseParams.BaseMgmtPress   = 1;
    baseParams.BaseFatigue     = 1;
    baseParams.BaseQueuedTasks = 0;
    baseParams.BaseQueuedTickets = 0;
    baseParams.BaseCompleteTickets = 0;
    baseParams.BaseCompleteTasks = 0;
    baseParams.ErrorRate = 0.005;
    baseParams.Iteration = 1;

    % Arrival rates and other parameters
    baseParams.IncTaskArrivalRate = 0.2024906;
    baseParams.ReqTaskArrivalRate = 0.0930726;
    baseParams.util_obs = 0.95;
    baseParams.queue_obs = 4.0;
```

```
    baseParams.ReqTimePDFLambdaObs = 0.1961;
    baseParams.IncTimePDFLambdaObs = 0.1469;

    % Initial guess for independent variables
    baseParams.ServiceTimeAct = 0.3;
    baseParams.ReqServiceTime = 0.15;
    baseParams.ProjTaskArrivalRate = 0.2;
    baseParams.MaintTaskArrivalRate = 0.125;
    baseParams.AdminTaskArrivalRate = 2.0;

    % Regression-determined variables
%    baseParams.ServiceTimeAct = 0.3424;
%    baseParams.ReqServiceTime = 0.5994;
%    baseParams.ProjTaskArrivalRate = 0.2411;
%    baseParams.MaintTaskArrivalRate = 0.1543;
%    baseParams.AdminTaskArrivalRate = 0.1713;

    % Initialize DES input history as a numeric matrix (each column is an iteration):
    % [ErrorRate; ServiceTimeAct; Iteration]
    desHistory = [baseParams.ErrorRate; baseParams.ServiceTimeAct; baseParams.Iteration];
end
```

configureSimEvents.m

```
function simIn = configureSimEvents(baseParams)
% configureSimEvents Configures the SimEvents simulation input.
%   simIn = configureSimEvents(baseParams)
%
% Attempts to load the SimEvents model "iterateddes.slx" from the parent folder.
% If not found, it then checks the current folder. Before loading, any previously
% loaded instance of the model is closed to ensure a fresh start. The SimulationInput
% is then configured with 'LoadInitialState' set to 'off', forcing a fresh simulation
% state.
%
% This version also sets each generator block's AttributeInitialValue so
% that the modular approach fully matches the monolithic DESAutomation.m.

    mdlName = 'iterateddes';
    mdlFile = [mdlName, '.slx'];

    % Try to locate the model in the parent folder
    mdlPathParent = fullfile('..', mdlFile);
    if exist(mdlPathParent, 'file')
        mdlPath = mdlPathParent;
    else
        % If not found, try the current folder
        mdlPathCurrent = fullfile(pwd, mdlFile);
        if exist(mdlPathCurrent, 'file')
            mdlPath = mdlPathCurrent;
        else
            error('Model file %s not found in either the parent folder or the current folder.', mdlFile);
        end
    end
```

```

% Pre-load cleanup: if the model is already loaded, close it.
if bdIsLoaded(mdlName)
    close_system(mdlName, 0);
end
pause(1);  % Allow time for cleanup

% Load the system freshly
load_system(mdlPath);

% Create a SimulationInput object
simIn = Simulink.SimulationInput(mdlName);

% Ensure that a saved final state is not used for the next run.
simIn = setModelParameter(simIn, 'LoadInitialState', 'off');

% ---------------------------------------------------------------------
% (1) Set the intergeneration time actions based on arrival rates.
% ---------------------------------------------------------------------
simIn = setBlockParameter(simIn, [mdlName, '/IncGen'], ...
    'IntergenerationTimeAction', ...
    "dt = -" + baseParams.IncTaskArrivalRate + "*log(1-rand());");

simIn = setBlockParameter(simIn, [mdlName, '/ReqGen'], ...
    'IntergenerationTimeAction', ...
    "dt = -" + baseParams.ReqTaskArrivalRate + "*log(1-rand());");

simIn = setBlockParameter(simIn, [mdlName, '/ProjTaskGen'], ...
    'IntergenerationTimeAction', ...
    "dt = -" + baseParams.ProjTaskArrivalRate + "*log(1-rand());");

simIn = setBlockParameter(simIn, [mdlName, '/MaintTaskGen'], ...
    'IntergenerationTimeAction', ...
    "dt = -" + baseParams.MaintTaskArrivalRate + "*log(1-rand());");

simIn = setBlockParameter(simIn, [mdlName, '/AdminTaskGen'], ...
    'IntergenerationTimeAction', ...
    "dt = -" + baseParams.AdminTaskArrivalRate + "*log(1-rand());");

% ---------------------------------------------------------------------
% (2) Match the monolithic AttributeInitialValue settings for ALL blocks.
% ---------------------------------------------------------------------
% Format in DESAutomation.m:
% "0|0|0|ReqServiceTime|0|0|0|0|0|ErrorRate|(WorkType)|1|0|1|ServiceTimeAct|(finalFlag)"
%
% WorkType:  1=Inc, 2=Req, 3=Proj, 4=Maint, 5=Admin
% finalFlag: 2 for Inc/Req/Proj/Maint, 1 for Admin
%
% (a) IncGen
simIn = setBlockParameter(simIn, [mdlName, '/IncGen'], ...
    'AttributeInitialValue', ...
    "0|0|0|" + baseParams.ReqServiceTime + "|0|0|0|0|0|" + baseParams.ErrorRate ...
    + "|1|1|0|1|" + baseParams.ServiceTimeAct + "|2");

% (b) IncidentsfromReworkGen - same settings as IncGen, still WorkType=1
simIn = setBlockParameter(simIn, [mdlName, '/IncidentsfromReworkGen'], ...
```

```
        'AttributeInitialValue', ...
        "0|0|0|" + baseParams.ReqServiceTime + "|0|0|0|0|0|" + baseParams.ErrorRate ...
        + "|1|1|0|1|" + baseParams.ServiceTimeAct + "|2");

    % (c) ReqGen - WorkType=2
    simIn = setBlockParameter(simIn, [mdlName, '/ReqGen'], ...
        'AttributeInitialValue', ...
        "0|0|0|" + baseParams.ReqServiceTime + "|0|0|0|0|0|" + baseParams.ErrorRate ...
        + "|2|1|0|1|" + baseParams.ServiceTimeAct + "|2");

    % (d) ProjTaskGen - WorkType=3
    simIn = setBlockParameter(simIn, [mdlName, '/ProjTaskGen'], ...
        'AttributeInitialValue', ...
        "0|0|0|" + baseParams.ReqServiceTime + "|0|0|0|0|0|" + baseParams.ErrorRate ...
        + "|3|1|0|1|" + baseParams.ServiceTimeAct + "|2");

    % (e) MaintTaskGen - WorkType=4
    simIn = setBlockParameter(simIn, [mdlName, '/MaintTaskGen'], ...
        'AttributeInitialValue', ...
        "0|0|0|" + baseParams.ReqServiceTime + "|0|0|0|0|0|" + baseParams.ErrorRate ...
        + "|4|1|0|1|" + baseParams.ServiceTimeAct + "|2");

    % (f) AdminTaskGen - WorkType=5, finalFlag=1
    simIn = setBlockParameter(simIn, [mdlName, '/AdminTaskGen'], ...
        'AttributeInitialValue', ...
        "0|0|0|" + baseParams.ReqServiceTime + "|0|0|0|0|0|" + baseParams.ErrorRate ...
        + "|5|1|0|1|" + baseParams.ServiceTimeAct + "|1");

end
```

runSimEvents.m

```
function simOut = runSimEvents(simIn)
% runSimEvents Runs the SimEvents simulation with the given input configuration.
%   simOut = runSimEvents(simIn)
%
% Input:
%   simIn  - A Simulink.SimulationInput object configured for the simulation.
%
% Output:
%   simOut - The simulation output structure.

    simOut = sim(simIn);
end
```

processSimOutput.m

```
function simMetrics = processSimOutput(simOut, baseParams)
% processSimOutput Processes simulation output from SimEvents and extracts metrics.
%
% This version ensures that each engineer (E1, SE1, SE2, SE3) is handled
```

```
% separately for "picked up" work, matching the monolithic DESAutomation.m script.

    %% 1. Basic info
    IterationLength = simOut.SimulationMetadata.ModelInfo.StopTime;
    simMetrics.IterationLength = IterationLength;

    %% 2. Totals for Stopped Work
    % -- E1
    TS_E1 = get(simOut.logsout, "CompSwitchEng1Stop").Values.WorkType;
    [StoppedIncidentsE1, StoppedRequestsE1, StoppedProjTasksE1, StoppedMaintTasksE1, StoppedAdminTasksE1] ...
        = countStoppedByWorkType(TS_E1);
    AllTicketsStoppedE1    = StoppedIncidentsE1 + StoppedRequestsE1;
    AllWorkTasksStoppedE1  = StoppedProjTasksE1 + StoppedMaintTasksE1;

    % -- SE1
    TS_SE1 = get(simOut.logsout, "CompSwitchSrEng1Stop").Values.WorkType;
    [StoppedIncidentsSE1, StoppedRequestsSE1, StoppedProjTasksSE1, StoppedMaintTasksSE1, StoppedAdminTasksSE1] ...
        = countStoppedByWorkType(TS_SE1);

    % -- SE2
    TS_SE2 = get(simOut.logsout, "CompSwitchSrEng2Stop").Values.WorkType;
    [StoppedIncidentsSE2, StoppedRequestsSE2, StoppedProjTasksSE2, StoppedMaintTasksSE2, StoppedAdminTasksSE2] ...
        = countStoppedByWorkType(TS_SE2);

    % -- SE3
    TS_SE3 = get(simOut.logsout, "CompSwitchSrEng3Stop").Values.WorkType;
    [StoppedIncidentsSE3, StoppedRequestsSE3, StoppedProjTasksSE3, StoppedMaintTasksSE3, StoppedAdminTasksSE3] ...
        = countStoppedByWorkType(TS_SE3);

    AllTicketsStopped = (StoppedIncidentsE1 + StoppedRequestsE1) ...
                      + (StoppedIncidentsSE2 + StoppedRequestsSE2) ...
                      + (StoppedIncidentsSE3 + StoppedRequestsSE3) ...
                      + (StoppedIncidentsSE1 + StoppedRequestsSE1);

    AllWorkTasksStopped = (StoppedProjTasksE1 + StoppedMaintTasksE1) ...
                        + (StoppedProjTasksSE1 + StoppedMaintTasksSE1) ...
                        + (StoppedProjTasksSE2 + StoppedMaintTasksSE2) ...
                        + (StoppedProjTasksSE3 + StoppedMaintTasksSE3);

    %% 3. Totals for Picked-Up Work
    % E1
    [PickedUpIncidentsE1, PickedUpRequestsE1, PickedUpProjTasksE1, PickedUpMaintTasksE1] = ...
        countPickedUpNonAdmin(simOut, "IndivQueueE1Out");

    % SE1
    [PickedUpIncidentsSE1, PickedUpRequestsSE1, PickedUpProjTasksSE1, PickedUpMaintTasksSE1] = ...
        countPickedUpNonAdmin(simOut, "IndivQueueSE1Out");

    % SE2
    [PickedUpIncidentsSE2, PickedUpRequestsSE2, PickedUpProjTasksSE2, PickedUpMaintTasksSE2] = ...
        countPickedUpNonAdmin(simOut, "IndivQueueSE2Out");

    % SE3
    [PickedUpIncidentsSE3, PickedUpRequestsSE3, PickedUpProjTasksSE3, PickedUpMaintTasksSE3] = ...
        countPickedUpNonAdmin(simOut, "IndivQueueSE3Out");

```

```
AllTicketsPickedUp = (PickedUpIncidentsE1 + PickedUpRequestsE1) ...
                   + (PickedUpIncidentsSE1 + PickedUpRequestsSE1) ...
                   + (PickedUpIncidentsSE2 + PickedUpRequestsSE2) ...
                   + (PickedUpIncidentsSE3 + PickedUpRequestsSE3);

AllTasksPickedUp = (PickedUpProjTasksE1 + PickedUpMaintTasksE1) ...
                 + (PickedUpProjTasksSE1 + PickedUpMaintTasksSE1) ...
                 + (PickedUpProjTasksSE2 + PickedUpMaintTasksSE2) ...
                 + (PickedUpProjTasksSE3 + PickedUpMaintTasksSE3);

AllWorkedPickedUp = AllTicketsPickedUp + AllTasksPickedUp;

%% 4. Generated Work (same as monolithic)
IncidentsGenerated         = countGenerated(simOut, "IncGen");
IncFromChgGenerated        = countGenerated(simOut, "IncidentsfromReworkGen");
RequestsGenerated          = countGenerated(simOut, "ReqGen");
ProjectTasksGenerated      = countGenerated(simOut, "ProjTaskGen");
MaintenanceTasksGenerated  = countGenerated(simOut, "MaintTaskGen");
AdminTasksGenerated        = countGenerated(simOut, "AdminTaskGen");

AllTicketsGenerated        = IncidentsGenerated + RequestsGenerated;
AllWorkTasksGenerated      = ProjectTasksGenerated + IncFromChgGenerated + MaintenanceTasksGenerated;
AllWorkGenerated           = AllTicketsGenerated + AllWorkTasksGenerated;

if AllWorkGenerated > 0
    PercentTasks   = AllWorkTasksGenerated / AllWorkGenerated;
    PercentTickets = AllTicketsGenerated   / AllWorkGenerated;
else
    PercentTasks   = 0;
    PercentTickets = 0;
end

%% 5. Completed Work (same as monolithic)
IncidentsCompleted          = countCompleted(simOut, "IncComp");
RequestsCompleted           = countCompleted(simOut, "ReqComp");
ProjectTasksCompleted       = countCompleted(simOut, "ProjTaskComp");
MaintenanceTasksCompleted   = countCompleted(simOut, "MaintTaskComp");
AdminTasksCompleted         = countCompleted(simOut, "AdminWorkComp");

AllTicketsCompleted         = IncidentsCompleted + RequestsCompleted;
AllWorkTasksCompleted       = ProjectTasksCompleted + MaintenanceTasksCompleted;
AllWorkCompleted            = AllTicketsCompleted + AllWorkTasksCompleted;

%% 6. Compute Rates
TicketsPickedUpPerDay   = AllTicketsPickedUp     / IterationLength;
TasksPickedUpPerDay     = AllTasksPickedUp       / IterationLength;
TicketsStoppedPerDay    = AllTicketsStopped      / IterationLength;
TasksStoppedPerDay      = AllWorkTasksStopped    / IterationLength;
TicketsCompletedPerDay  = AllTicketsCompleted    / IterationLength;
TasksCompletedPerDay    = AllWorkTasksCompleted  / IterationLength;

errRate = baseParams.ErrorRate;  % Or from simOut if appropriate
ErrorsPerDay       = AllWorkCompleted * errRate;
BaseReworkRate     = ErrorsPerDay * PercentTasks;
BaseIncFromChange  = ErrorsPerDay * PercentTickets;
```

```
    %% 7. Build simMetrics
    simMetrics.TicketsPickedUpPerDay    = TicketsPickedUpPerDay;
    simMetrics.TasksPickedUpPerDay      = TasksPickedUpPerDay;
    simMetrics.TicketsStoppedPerDay     = TicketsStoppedPerDay;
    simMetrics.TasksStoppedPerDay       = TasksStoppedPerDay;
    simMetrics.TicketsCompletedPerDay   = TicketsCompletedPerDay;
    simMetrics.TasksCompletedPerDay     = TasksCompletedPerDay;
    simMetrics.BaseReworkRate           = BaseReworkRate;
    simMetrics.BaseIncFromChange        = BaseIncFromChange;
    simMetrics.BaseMgmtPress            = baseParams.BaseMgmtPress;
    simMetrics.BaseFatigue              = baseParams.BaseFatigue;
    simMetrics.BaseQueuedTasks          = baseParams.BaseQueuedTasks;
    simMetrics.BaseQueuedTickets        = baseParams.BaseQueuedTickets;
    simMetrics.BaseMgmtPreempt          = baseParams.BaseMgmtPreempt;
    simMetrics.BaseCompleteTasks        = baseParams.BaseCompleteTasks;
    simMetrics.BaseCompleteTickets      = baseParams.BaseCompleteTickets;
    simMetrics.Iteration                = baseParams.Iteration;

    % Also store any raw totals if desired.
    simMetrics.AllTicketsPickedUp       = AllTicketsPickedUp;
    simMetrics.AllWorkTasksPickedUp     = AllTasksPickedUp;
    simMetrics.AllTicketsStopped        = AllTicketsStopped;
    simMetrics.AllWorkTasksStopped      = AllWorkTasksStopped;
    simMetrics.AllTicketsCompleted      = AllTicketsCompleted;
    simMetrics.AllWorkTasksCompleted    = AllWorkTasksCompleted;
end

%% Helper Functions
function [nInc, nReq, nProj, nMaint, nAdmin] = countStoppedByWorkType(tsVals)
    if isempty(tsVals.Time)
        nInc=0; nReq=0; nProj=0; nMaint=0; nAdmin=0;
    else
        T      = timetable2table(timeseries2timetable(tsVals));
        Summ   = groupsummary(T, "WorkType");
        nInc   = getCount(Summ,1);
        nReq   = getCount(Summ,2);
        nProj  = getCount(Summ,3);
        nMaint = getCount(Summ,4);
        nAdmin = getCount(Summ,5);
    end
end

function [inc, req, proj, maint] = countPickedUpNonAdmin(simOut, signalName)
    TS_admin = get(simOut.logsout, signalName).Values.IsAdmin;
    if isempty(TS_admin.Time)
        inc=0; req=0; proj=0; maint=0;
        return
    end
    T_admin = timetable2table(timeseries2timetable(TS_admin));
    T_type  = timetable2table(timeseries2timetable(get(simOut.logsout, signalName).Values.WorkType));
    % Remove the time column from T_type
    T_type(:,1) = [];
    Combined = [T_admin, table(T_type.WorkType, 'VariableNames', {'WorkType'})];
    NonAdmin = Combined(Combined.IsAdmin ~= 1, :);
    Summ = groupsummary(NonAdmin,"WorkType");
    inc   = getCount(Summ,1);
```

```
    req   = getCount(Summ,2);
    proj  = getCount(Summ,3);
    maint = getCount(Summ,4);
end

function nGenerated = countGenerated(simOut, blockName)
    valObj = get(simOut.logsout, blockName).Values.IsAdmin;
    if isempty(valObj.Data)
        nGenerated = 0;
    else
        nGenerated = height(timetable2table(timeseries2timetable(valObj)));
    end
end

function nCompleted = countCompleted(simOut, blockName)
    tsObj = get(simOut.logsout, blockName).Values.IsAdmin;
    if isempty(tsObj.Time)
        tsObj = timeseries(0,0);
    end
    nCompleted = height(timetable2table(timeseries2timetable(tsObj)));
end

function c = getCount(tbl, wtype)
    idx = find(tbl.WorkType == wtype, 1);
    if isempty(idx)
        c = 0;
    else
        c = tbl.GroupCount(idx);
    end
end
```

analyzeUtilization.m

```
function util_pred = analyzeUtilization(simOut)
% analyzeUtilization Analyzes utilization data.
%   util_pred = analyzeUtilization(simOut)
%
% Retrieves utilization time series for each engineer, converts them to timetables,
% synchronizes the timetables, computes the average utilization over time, and returns
% the final average value.

    E1UtilTS = simOut.E1Utilization;
    if isempty(E1UtilTS.Time)
        E1UtilTS = timeseries(0,0);
    end
    E1UtilTT = timeseries2timetable(E1UtilTS);

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

```

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

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

    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);
end
```

analyzeTimeData.m

```
function [IncTimePDFLambdaSim, ReqTimePDFLambdaSim, AllTimePDFlambdaSim] = analyzeTimeData(simOut)
% analyzeTimeData Analyzes total time data for incidents, requests, and optionally for
% project, maintenance, and admin tasks to replicate the monolithic script fully.
%
%   [IncTimePDFLambdaSim, ReqTimePDFLambdaSim, AllTimePDFlambdaSim] = analyzeTimeData(simOut)
%
% IncTimePDFLambdaSim:   1 / mean(IncTime), or 0 if no incident times
% ReqTimePDFLambdaSim:   1 / mean(ReqTime),  or 0 if no request times
% AllTimePDFlambdaSim:   1 / mean of all times combined (Inc, Req, Proj, Maint, Admin),
%                        or 0 if no data

    % --- Incidents ---
    IncTimeTS = get(simOut.logsout, "TotalTimeInc").Values;
    if isempty(IncTimeTS.Time)
        IncTimeTS = timeseries(0,0);
    end
    IncTimeTT = timeseries2timetable(IncTimeTS);
    if isempty(IncTimeTT.Time)
        IncTimePDFLambdaSim = 0;
        IncTimeTTData = table();
    else
        incVarName  = IncTimeTT.Properties.VariableNames{1};
        IncTimeMean = mean(IncTimeTT.(incVarName));
        IncTimePDFLambdaSim = 1 / IncTimeMean;
        % rename the data column so we can combine easily
        IncTimeTTData = IncTimeTT;
        IncTimeTTData.Properties.VariableNames{incVarName} = 'Data';
    end

    % --- Requests ---
    ReqTimeTS = get(simOut.logsout, "TotalTimeReq").Values;
    if isempty(ReqTimeTS.Time)
        ReqTimeTS = timeseries(0,0);
    end
```

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

% --- Projects ---
ProjTimeTS = get(simOut.logsout, "TotalTimeProj").Values;
if isempty(ProjTimeTS.Time)
    ProjTimeTS = timeseries(0,0);
end
ProjTimeTT = timeseries2timetable(ProjTimeTS);
if isempty(ProjTimeTT.Time)
    ProjTimeTTData = table();
else
    projVarName = ProjTimeTT.Properties.VariableNames{1};
    ProjTimeTTData = ProjTimeTT;
    ProjTimeTTData.Properties.VariableNames{projVarName} = 'Data';
end

% --- Maintenance ---
MaintTimeTS = get(simOut.logsout, "TotalTimeMaint").Values;
if isempty(MaintTimeTS.Time)
    MaintTimeTS = timeseries(0,0);
end
MaintTimeTT = timeseries2timetable(MaintTimeTS);
if isempty(MaintTimeTT.Time)
    MaintTimeTTData = table();
else
    maintVarName = MaintTimeTT.Properties.VariableNames{1};
    MaintTimeTTData = MaintTimeTT;
    MaintTimeTTData.Properties.VariableNames{maintVarName} = 'Data';
end

% --- Admin ---
AdminTimeTS = get(simOut.logsout, "TotalTimeAdmin").Values;
if isempty(AdminTimeTS.Time)
    AdminTimeTS = timeseries(0,0);
end
AdminTimeTT = timeseries2timetable(AdminTimeTS);
if isempty(AdminTimeTT.Time)
    AdminTimeTTData = table();
else
    adminVarName = AdminTimeTT.Properties.VariableNames{1};
    AdminTimeTTData = AdminTimeTT;
    AdminTimeTTData.Properties.VariableNames{adminVarName} = 'Data';
end

% --- Combine all data for a single "AllTime" distribution ---
AllTimeTT = [IncTimeTTData; ReqTimeTTData; ProjTimeTTData; MaintTimeTTData; AdminTimeTTData];
```

```
    if isempty(AllTimeTT)
        AllTimePDFlambdaSim = 0;
    else
        % Fit an exponential distribution across *all* data, just like the monolithic script:
        AllTimePDF = fitdist(AllTimeTT.Data, 'Exponential');
        AllTimePDFlambdaSim = 1 / AllTimePDF.mu;
    end
end
```

analyzeQueueData.m

```
function queue_pred = analyzeQueueData(simOut)
% analyzeQueueData Analyzes queue length data.
%   queue_pred = analyzeQueueData(simOut)
%
% Retrieves the queue length time series for each engineer's queue, converts them
% to timetables, synchronizes the timetables, computes the average queue length,
% and returns the final average value.

    E1QueueTS = simOut.E1QueueLength;
    if isempty(E1QueueTS.Time)
        E1QueueTS = timeseries(0,0);
    end
    E1QueueTT = timeseries2timetable(E1QueueTS);

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

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

    SE3QueueTS = simOut.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
```

excelOperations.m

```
function excelOperations(operation, SDParameters)
```

```
% excelOperations Performs Excel file operations such as closing or writing data.
%   excelOperations('close')
%   excelOperations('write', SDParameters)
%
% For the 'write' operation, SDParameters should be a numeric array containing
% the following values (in order) in column A:
% TicketsPickedUpPerDay,
% TasksPickedUpPerDay,
% TicketsStoppedPerDay,
% TasksStoppedPerDay,
% TicketsCompletedPerDay,
% TasksCompletedPerDay,
% BaseReworkRate,
% BaseIncFromChange,
% BaseMgmtPress,
% BaseFatigue,
% BaseQueuedTasks,
% BaseQueuedTickets,
% BaseMgmtPreempt,
% BaseCompleteTasks,
% BaseCompleteTickets,
% Iteration
%
% The data is written to the Excel file "sddatain.xlsx" in the specified folder,
% with no column headings. Before writing, the function attempts to close any
% open instance of the workbook.

    targetWorkbookName = 'sddatain.xlsx';
    filePath = fullfile('C:\Users\enos9\OneDrive - Colostate\combined\', targetWorkbookName);

    switch lower(operation)
        case 'close'
            try
                excelApp = actxGetRunningServer('Excel.Application');
            catch
                % If no Excel instance is running, nothing to close.
                disp('No running Excel instance found.');
                return;
            end

            % Close any open workbook matching the target name.
            for i = excelApp.Workbooks.Count:-1:1
                if strcmpi(excelApp.Workbooks.Item(i).Name, targetWorkbookName)
                    excelApp.Workbooks.Item(i).Close(false);
                end
            end
            release(excelApp);
            disp('All matching workbooks closed.');

        case 'write'
            if nargin < 2
                error('For the write operation, you must provide the SDParameters numeric array.');
            end

            % Attempt to close the workbook if it is open.
            try
```

```
                excelApp = actxGetRunningServer('Excel.Application');
                for i = excelApp.Workbooks.Count:-1:1
                    wb = excelApp.Workbooks.Item(i);
                    if strcmpi(wb.Name, targetWorkbookName)
                        wb.Close(false);
                    end
                end
                release(excelApp);
            catch
                % If Excel is not running, we simply continue.
            end

            % Ensure SDParameters is a column vector.
            if size(SDParameters, 2) > 1
                SDParameters = SDParameters(:);
            end

            % Write the matrix to Excel without any column headings.
            writematrix(SDParameters, filePath, 'Sheet', 1);
            disp(['Data written to ', filePath]);

        otherwise
            error('Unknown operation specified for excelOperations.');
    end
end
```

triggerVensimSimulation.m

```
function triggerVensimSimulation()
% triggerVensimSimulation Launches the external Vensim simulation and monitors its execution.
%
% This function triggers Vensim using an external command (via system calls) and
% monitors the process. If the simulation does not complete within the timeout,
% it terminates the process.

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

    % Launch the command asynchronously.
    system(['start "" ' externalCommand]);

    % Define timeout duration (in seconds).
    timeoutDuration = 90;

    disp('Monitoring the external Vensim simulation...');
    startTime = tic;

    while true
        [~, result] = system('tasklist');
        % Check if Vensim is still running (process name vendss64.exe).
        if ~contains(result, 'vendss64.exe')
            disp('External Vensim simulation completed successfully.');
```

```
            break;
        end

        if toc(startTime) > timeoutDuration
            disp('Timeout exceeded. Terminating external Vensim simulation...');
            system('taskkill /IM vendss64.exe /F /T');
            break;
        end

        pause(1);
    end
end
```

Vensim Command Script

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

updateIterationParameters.m

```
function [baseParams, desHistory, residuals] = updateIterationParameters(baseParams, desHistory, ~)
% updateIterationParameters Updates carry-over parameters and computes residuals.
%   [baseParams, desHistory, residuals] = updateIterationParameters(baseParams, desHistory, ~)
%
% This function reads the Vensim output file "sddataout.xlsx" and computes the
% mean for each row (ignoring labels and the time row). The resulting vector is
% assumed to have at least 17 rows. In particular:
%
%   Row 13: BaseMgmtPreempt
%   Row 9:  BaseMgmtPress
%   Row 10: BaseFatigue (which is then multiplied by 0.75)
%   Row 16: BaseQueuedTasks
%   Row 17: BaseQueuedTickets
%
% The function updates baseParams accordingly, updates ServiceTimeAct (with a
% lower bound of 1/48 to ensure at least 10 minutes per task), increments the iteration
% number, and updates the DES input history.
%
% (Any extra rows not needed are ignored.)

    % Read Vensim output from Excel.
```

```
outputFile = "C:\Users\enos9\OneDrive - Colostate\combined\sddataout.xlsx";
SDDataOut = readmatrix(outputFile, 'Sheet', 1);

% Remove the first column (labels) and the first row (time).
SDDataOut(:,1) = [];
SDDataOut(1,:) = [];

% Compute the mean of each row across the simulation days.
SDDataOutMean = mean(SDDataOut, 2);

% We now expect at least 17 rows (the 17th row gives BaseQueuedTickets).
expectedRows = 17;
if length(SDDataOutMean) < expectedRows
    warning('SDDataOutMean has fewer rows than expected. Missing values will be set to 0.');
    SDDataOutMean(end+1:expectedRows) = 0;
end

% Extract the values needed.
newMgmtPreempt   = SDDataOutMean(13);
newMgmtPress     = SDDataOutMean(9);
newFatigue       = SDDataOutMean(10) * 0.75; % apply fatigue adjustment
newQueuedTasks   = SDDataOutMean(16);
newQueuedTickets = SDDataOutMean(17);

% For complete work numbers, use the existing baseParams values.
newCompleteTasks  = baseParams.BaseCompleteTasks;
newCompleteTickets = baseParams.BaseCompleteTickets;

% Save the old BaseMgmtPreempt for computing the change.
oldMgmtPreempt = baseParams.BaseMgmtPreempt;

% Update the base parameters with the new values from Vensim output.
baseParams.BaseMgmtPreempt   = newMgmtPreempt;
baseParams.BaseMgmtPress     = newMgmtPress;
baseParams.BaseFatigue       = newFatigue;
baseParams.BaseQueuedTasks   = newQueuedTasks;
baseParams.BaseQueuedTickets = newQueuedTickets;
baseParams.BaseCompleteTasks = newCompleteTasks;
baseParams.BaseCompleteTickets = newCompleteTickets;

% Update ServiceTimeAct based on the change in management preemption.
SDMgmtChange = newMgmtPreempt - oldMgmtPreempt;
SDSvcTimeChange = (1 + SDMgmtChange);
baseParams.ServiceTimeAct = max(baseParams.ServiceTimeAct * SDSvcTimeChange, 1/48);

% Increment the iteration number.
baseParams.Iteration = baseParams.Iteration + 1;

% Update the DES input history.
newHistory = [baseParams.ErrorRate; baseParams.ServiceTimeAct; baseParams.Iteration];
if isstruct(desHistory)
    if isfield(desHistory, 'history')
        desHistory.history = [desHistory.history, newHistory];
    else
        desHistory.history = newHistory;
    end
```

```
    else
        desHistory = [desHistory, newHistory];
    end

    % (Residuals can be computed here if needed; placeholders below.)
    residuals = zeros(4,1);
end
```

Appendix B

Regression Scripts

Regression Script

```
% Establish parameters - initial 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)
```

```matlab
    % 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
```