# A Model-Based System for On-Premises Software Defined Infrastructure

Eric S. Enos

PhD Preliminary Exam

18 December 2023

WALTER SCOTT, JR. COLLEGE OF ENGINEERING
COLORADO STATE UNIVERSITY

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY

## Introduction

- United States Military Academy, BS, Russian and German, 1990
- University of Colorado, Colorado Springs, MBA, Finance and Project Management, 2004
- Chief Technology Officer, Lifepoint Health 2016-Present
- Leading healthcare provider IT organizations of various sizes since 1999

## Motivation for this Research

- My current team consists of roughly 150 people responsible for the enterprise technology stack supporting ~70 community hospitals, 40 rehabilitation hospitals, 20 behavioral health hospitals and dozens of ambulatory clinics and physician practices across the US.
- Like most other organizations, we are constantly challenged to “do more with less (or at least not more)” in support of our patients and clinicians.
- The DevOps model is *the* trend for infrastructure management – if you’re heavily deployed in the public cloud and write your own applications. We (and most healthcare providers) are not – in fact, many key clinical systems, including core Electronic Medical Records (EMRs), are unsupported in the cloud.
- The key question: can an IT organization like Lifepoint’s leverage DevOps processes, techniques and technologies to meet our mandate?

## On-Premises COTS Solutions in Healthcare

- Generally speaking, healthcare providers are heavily dependent on purchased applications for key functions including:
  - Enterprise Resource Planning (ERPs, such as SAP or Oracle) that provide finance, HR, supply chain, and other functions.
  - Electronic Health Records (EHRs such as Epic or Cerner) which provide core clinical functions.
  - Ancillary systems such as laboratory and radiology systems.
  - Monitoring systems such as telemetry and fetal monitoring.
- The historical deployment models for these applications all rely on on-premises infrastructure, and many do not support cloud-based deployment models (or if they do only in recent versions).
- Some applications are tightly coupled to on-premises systems such as biomedical monitors, lab equipment, and large radiology systems such as MRI and CT scanners – and many are life critical and should not be deployed remotely.
- The EBITDA-focused incentive structure for providers actively discourages cloud consumption models, which are primarily accounted for as OPEX expenses.

- Health Information Management Systems Society (HIMSS), The Cphims Review Guide. Taylor & Francis Group, Dec. 2021, google-Books-ID: IHJatwEACAAJ.

# The Healthcare Provider IT Organization

- Following from the nature of the systems involved, provider IT organizations are built to evaluate, purchase, deploy, integrate, support and upgrade / replace systems (primarily on premises) – not to build them (either on-premises or in the cloud).
- Hospitals and health systems provide services from extensive facilities that require IT services throughout, with network and computing equipment deployed throughout.
- Teams are organized around departmental silos: technologies (such as networks or endpoint computing), applications (especially certain types of COTS applications), and functions (e.g., project management, integration, or data & analytics).
- Cost pressures IT keep teams relatively small and outsourcing is common, so teams are responsible for both project and operational tasks within their departmental silos.
- Activities that require multiple skills must be coordinated across departmental silos.

- 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/.

# Cloud Deployment Models

- NIST characteristics of cloud services:
  1. *On-demand, self-service*
  2. Broad network access
  3. Resource pooling
  4. *Rapid elasticity or expansion*
  5. Measured service

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.

- **Private Cloud**: Only one organization can use the cloud service and the underlying resources… may be on premises or off premises and provides much greater control over data, underlying systems, and applications
- **Public Cloud**: Unrelated organizations use the shared cloud service and the underlying resources
- **Hybrid Cloud**: At least two or more distinct cloud infrastructures are connected together to facilitate hosted data and application portability

- 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

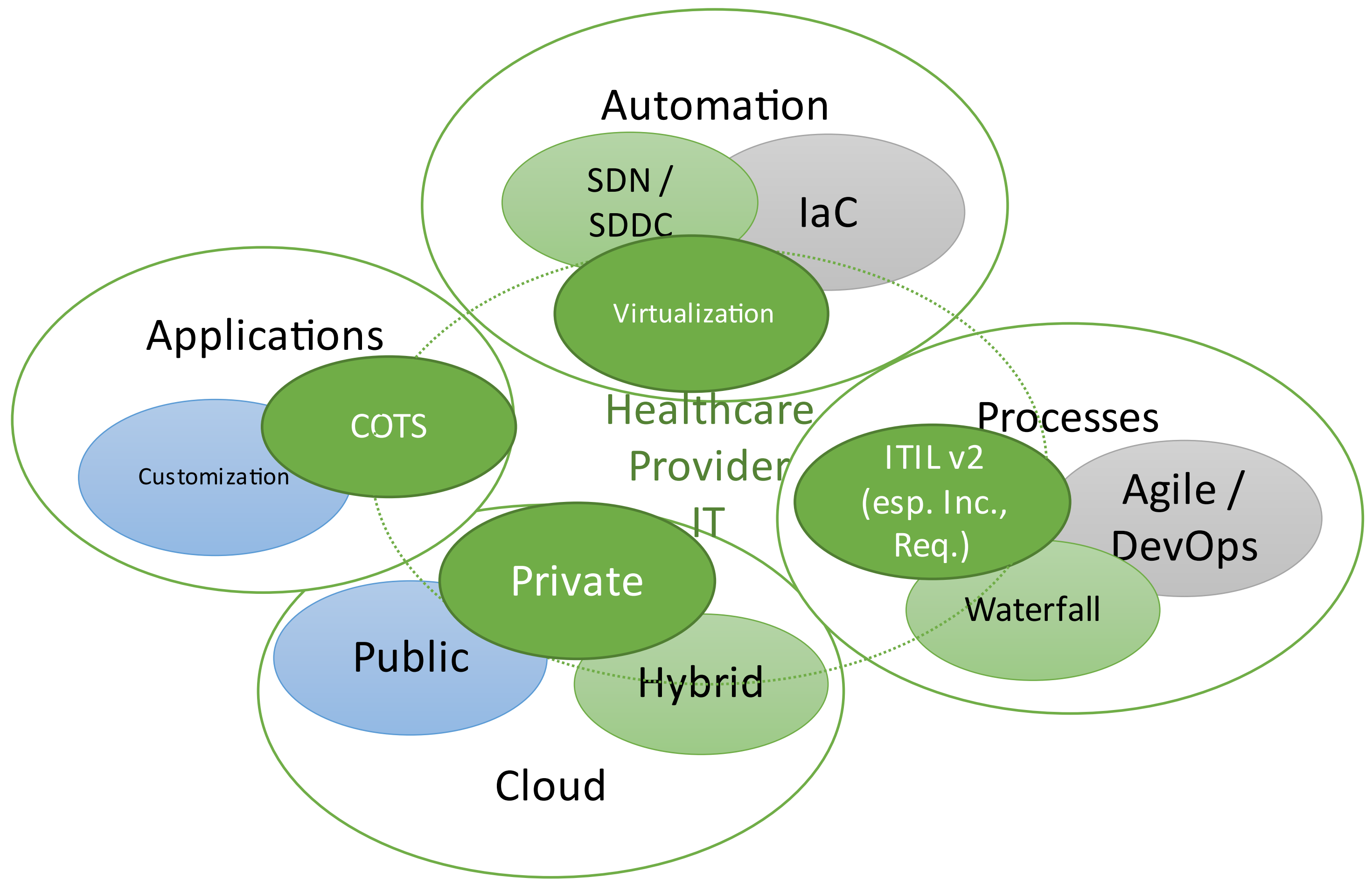
## Software Defined Infrastructure – Underpinning Private Cloud

- NIST refers to the underpinning infrastructure as “Cloud Infrastructure”:
  - The collection of hardware and software that enables the five essential characteristics of cloud computing
  - Managed by the cloud provider and not the cloud consumer in the context of public cloud services
  - Enables IaaS (Infrastructure as a Service), PaaS (Platform as a Service) and SaaS (Software as a Service)
- In the context of the on-premises private cloud environments that are the focus of this research, I will use the term *Software Defined Infrastructure* (SDI) to refer to the following combination of systems:
  - Networking (e.g., SDN)
  - Servers & Storage (e.g., SDDC)
  - Enabling software
    - Virtualization (the ability to run multiple instances of logical infrastructure on single devices)
    - Toolsets (Domain Naming Service, IP Address Management, identity directories, ticketing systems, monitors, etc.)
    - Automation / orchestration engines (the “glue”)
    - Middleware, databases, etc.

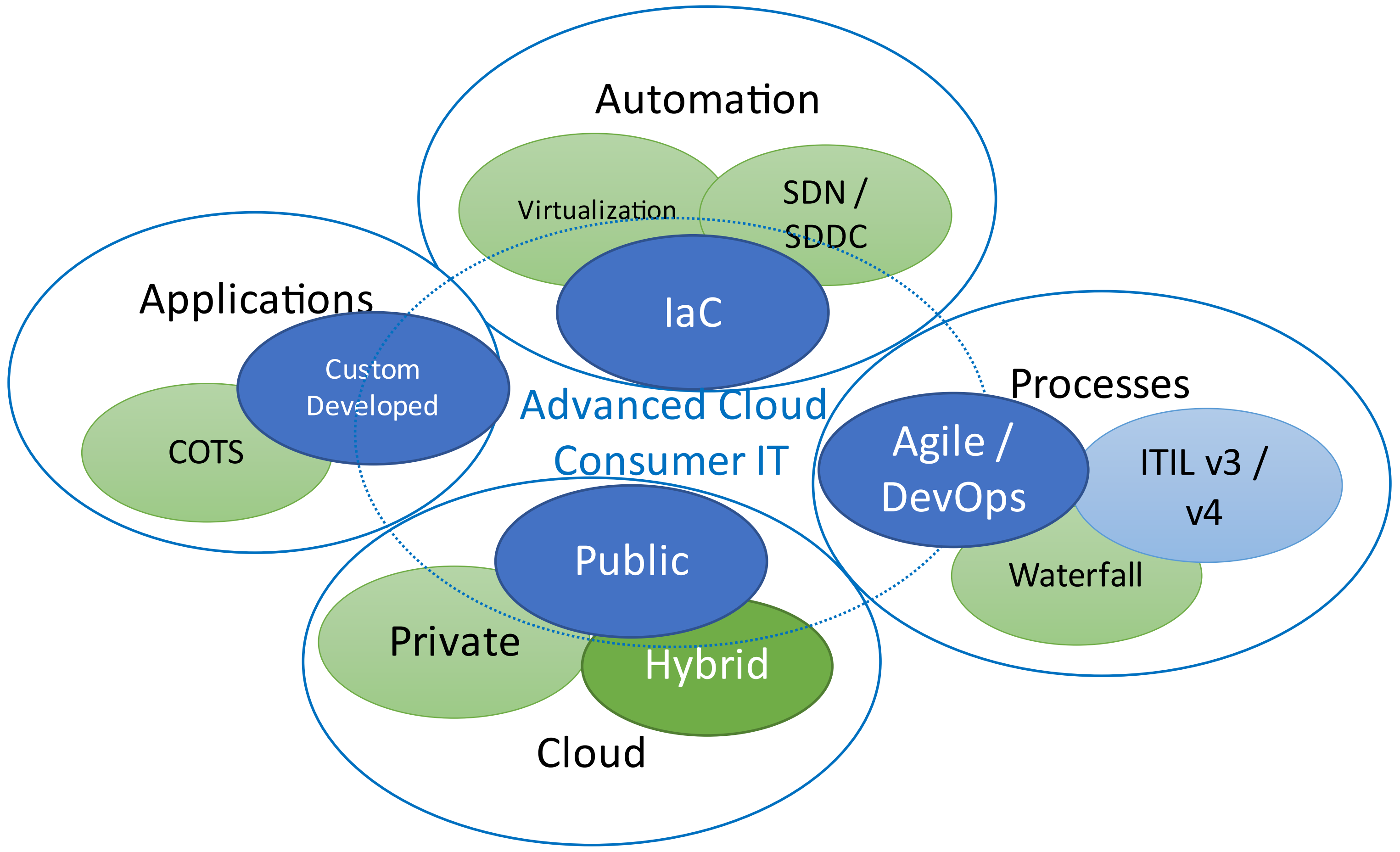
# Key Processes and Capabilities

- Work types:
  - Unscheduled work: Incident and Request tickets - IT Information Library (ITIL).
  - Scheduled work: Project and Maintenance tasks.
  - Teams are responsible for both and manage them through queues.
- Other relevant concepts:
  - DevOps: DevOps is a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software versions, while guaranteeing their correctness and reliability.
  - Infrastructure as Code (IaC): Infrastructure as code (IaC) is the practice to automatically configure system dependencies and to provision local and remote instances.

- ITIL Foundation, ITIL. TSO, The Stationary Office, 2019.
- L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, “A Survey of DevOps Concepts and Challenges,” ACM Comput. Surv., vol. 52, no. 6, pp. 1–35, Nov. 2020, doi: 10.1145/3359981.
- 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.

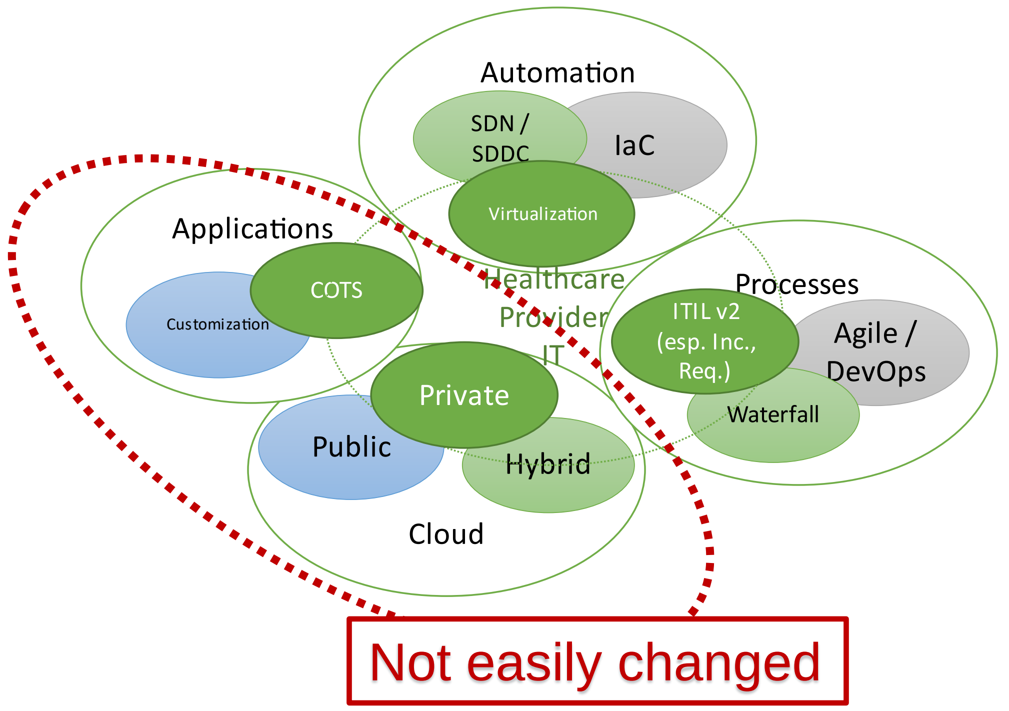


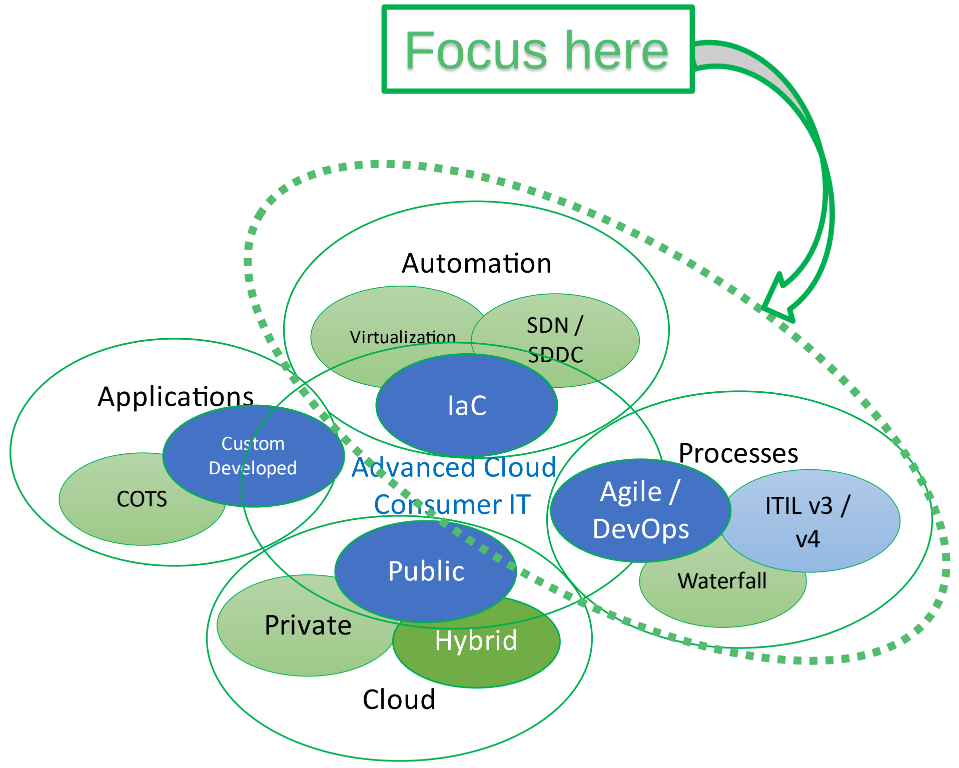
## The Healthcare Provider IT Environment



## Modern IT Environments

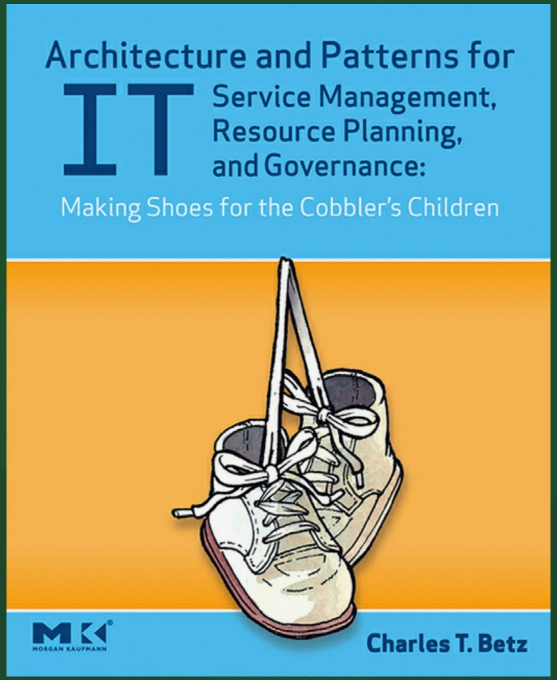
# Can we make this…





## …look more like this?

## “The cobbler’s children have no shoes...”


Architecture and Patterns for IT Service Management, Resource Planning, and Governance: Making Shoes for the Cobbler’s Children

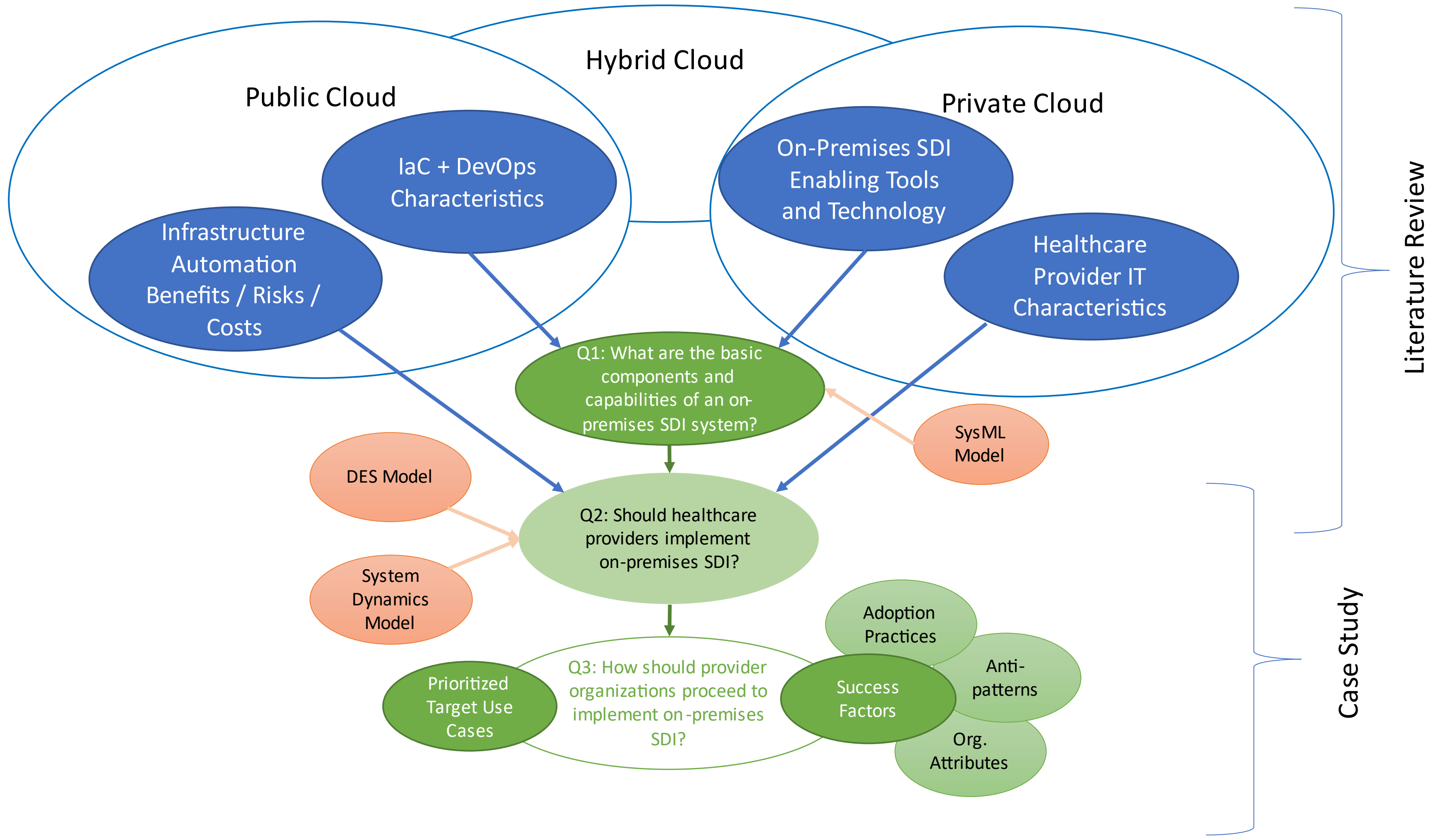
MK MORGAN KAUFMANN

Charles T. Betz

## Research Questions

- RQ1: What are the basic components and capabilities of an on-premises SDI system?
- RQ2: Should healthcare providers implement on-premises SDI?
- RQ3: How should provider organizations proceed to implement on-premises SDI, if at all?

# Flow of Research


Hybrid Cloud
Public Cloud
Private Cloud
IaC + DevOps Characteristics
Infrastructure Automation Benefits / Risks / Costs
On-Premises SDI Enabling Tools and Technology
Healthcare Provider IT Characteristics
Q1: What are the basic components and capabilities of an on-premises SDI system?
SysML Model
DES Model
Q2: Should healthcare providers implement on-premises SDI?
System Dynamics Model
Adoption Practices
Anti-patterns
Success Factors
Org. Attributes
Q3: How should provider organizations proceed to implement on-premises SDI?
Prioritized Target Use Cases
Literature Review
Case Study

## Leveraging Simulation to Determine Potential Benefits

Coupling Discrete Event Simulation and System Dynamics

## Start with *Whether* (or *How* Doesn’t Much Matter)

- Understand where opportunities exist and quantify the potential benefit:
  - Discrete Event Simulation – predict the performance of queuing in terms of total time, with prioritization by ticket or task.
  - System Dynamics Simulation – account for influences that total time, especially the impact of managerial pressure on rework and work stoppages driven by re-prioritization.
- Couple models into a hybrid simulation to obtain benefits of both modeling methods through iteration through each model individually.

- 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
- B. K. Choi, D. Kang, and B. K. Choi, Modeling and Simulation of Discrete Event Systems. Somerset, US: John Wiley & Sons, Incorporated, 2013, url: http://ebookcentral.proquest.com/lib/csu/detail.action?docID=1402564
- K. Chahal, T. Eldabi, and T. Young, “A conceptual framework for hybrid system dynamics and discrete event simulation for healthcare,” Journal of Enterprise Information Management, vol. 26, no. 1/2, pp. 50–74, Jan. 2013, publisher: Emerald Group Publishing Limited. doi: 10.1108/17410391311289541

## Basic Organizational Structure

- Teams are built around specific technologies: network engineering, email, virtualization, server administration, etc.
- Teams support both operational tickets and project tasks within their specialty.
  - Managing multiple projects within the same team is recognized as much more complex than single projects.
  - Managing multiple operational and project efforts – of varying priorities - within a single time is even more complex.
  - Work of any type (tickets or tasks) are effectively managed as queued, by priority.
- Teams have engineers of varying levels of skill.
  - There are generally fewer resources with high skill levels (“senior engineers”).
  - There are usually more with medium and lower levels of skill (“engineers” and “associate engineers”).
  - This varies by team – actual team structures are used in the models.
- 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).

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

## Basic Work Characteristics

- Work items (Tickets and tasks) can require a resource with a certain level of technical skill.
- Work items generally represents <5 hours of work.
  - Staff may not have sufficient service time available to complete a ticket or task, resulting in re-queuing and switching costs.
  - Available service time can change based on re-prioritization (escalating / expediting) of other work.
- Work items have a priority (low, medium or high) which can change over time.
  - Project tasks become more time-sensitive as they get close to – or pass – their due date, especially for those on the critical path (“timeliness”).
  - Operational tickets are more likely to be escalated the longer they’re open (“responsiveness”).
- Quality – or the lack of it - is reflected through the creation of rework (in the case of tasks) and incidents (in the case of tickets).
  - A base probability for rework / incidents is based on existing data, but can change due to dynamic influences.
  - Quality is influenced through a difference between the required skill level and the ability of the assigned resource.

- J. M. Lyneis and D. N. Ford, “System dynamics applied to project management: a survey, assessment, and directions for future research,” System Dynamics Review, vol. 23, no. 2-3, pp. 157–189, 2007, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sdr.377. doi: 10.1002/sdr.377.
- J. Wiik and K.-P. Kossakowski, “Dynamics of Incident Response,” Apr. 2017, url: https://www.first.org/resources/papers/2005.

# Single Team vs Multiple Teams

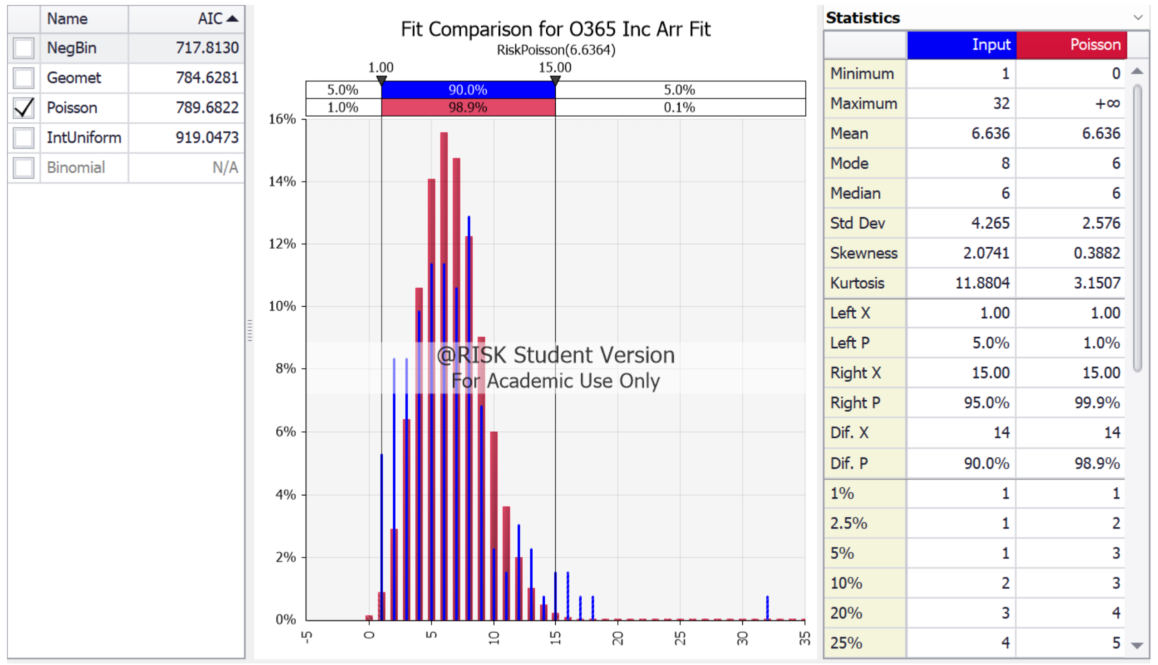
- Work items can be completed within a team or span multiple teams which results in one or more transfers.
- Work items can also be assigned incorrectly, resulting in a transfer.
- Transferred work items sometimes require execution of steps in a particular order by the different teams.
- Transferred work results in another random queuing time, based on its priority at the time in comparison to other work in the new team's queue.
- The modeling of interactions between two teams remains in process.
  - This will be limited to two teams as the complexity of the interactions increases as a "scale-free" network.

## Available Work Performance Data

- Operational requests and incidents are managed in ServiceNow
  - Data gathered by team, ticket type and priority for model for six months, from 1/1/23 to 6/30/23 including both arrival and closure dates.
  - Arrival data fit as unique Poisson distributions by team using Palisade @ Risk for use in the DES models as inter-arrival data.
  - Closure data will be used in verification and validation of the DES results.
  - Priority breakdowns are based on actual data by team.
    - In the case of the team analyzed: High 0.5%, Medium 40%, Low 59.5%.
- Other data elements determined through interview by team managers.
  - Validation and verification of parameters is underway.

# Fitting Existing Operational Data

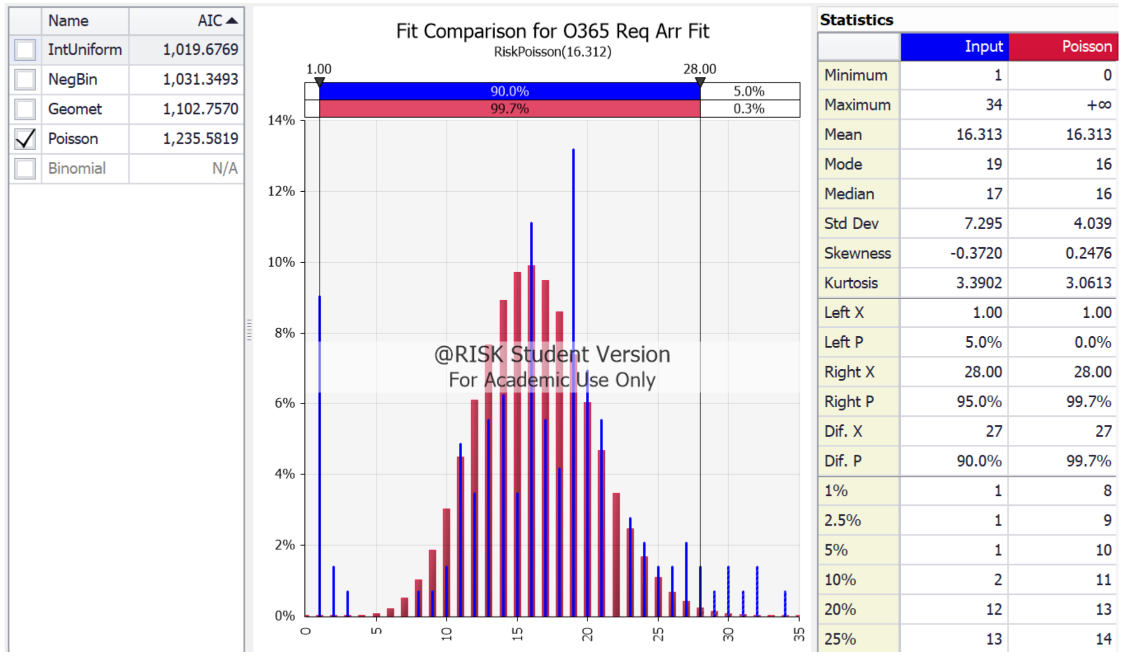
- Poisson distribution selected for fit of arrival times as this has direct translation into DES inter-arrival times based on exponential random arrival times.
- Example: Incident arrival fitted Poisson distribution mean of 6.6364 yields lambda of 0.150685 -> SimEvents generation equation of dt = -0.150685*log(1-rand()).



| | Name | AIC |
|---|---|---|
| ☐ | NegBin | 717.8130 |
| ☐ | Geomet | 784.6281 |
| ✓ | Poisson | 789.6822 |
| ☐ | IntUniform | 919.0473 |
| ☐ | Binomial | N/A |

Statistics

| | Input | Poisson |
|---|---|---|
| Minimum | 1 | 0 |
| Maximum | 32 | +∞ |
| Mean | 6.636 | 6.636 |
| Mode | 8 | 6 |
| Median | 6 | 6 |
| Std Dev | 4.265 | 2.576 |
| Skewness | 2.0741 | 0.3882 |
| Kurtosis | 11.8804 | 3.1507 |
| Left X | 1.00 | 1.00 |
| Left P | 5.0% | 1.0% |
| Right X | 15.00 | 15.00 |
| Right P | 95.0% | 99.9% |
| Dif. X | 14 | 14 |
| Dif. P | 90.0% | 98.9% |
| 1% | 1 | 1 |
| 2.5% | 1 | 2 |
| 5% | 1 | 3 |
| 10% | 2 | 3 |
| 20% | 3 | 4 |
| 25% | 4 | 5 |



| | Name | AIC |
|---|---|---|
| ☐ | IntUniform | 1,019.6769 |
| ☐ | NegBin | 1,031.3493 |
| ☐ | Geomet | 1,102.7570 |
| ✓ | Poisson | 1,235.5819 |
| ☐ | Binomial | N/A |

Statistics

| | Input | Poisson |
|---|---|---|
| Minimum | 1 | 0 |
| Maximum | 34 | +∞ |
| Mean | 16.313 | 16.313 |
| Mode | 19 | 16 |
| Median | 17 | 16 |
| Std Dev | 7.295 | 4.039 |
| Skewness | -0.3720 | 0.2476 |
| Kurtosis | 3.3902 | 3.0613 |
| Left X | 1.00 | 1.00 |
| Left P | 5.0% | 0.0% |
| Right X | 28.00 | 28.00 |
| Right P | 95.0% | 99.7% |
| Dif. X | 27 | 27 |
| Dif. P | 90.0% | 99.7% |
| 1% | 1 | 8 |
| 2.5% | 1 | 9 |
| 5% | 1 | 10 |
| 10% | 2 | 11 |
| 20% | 12 | 13 |
| 25% | 13 | 14 |

WALTER SCOTT, JR.
COLLEGE OF ENGINEERING
COLORADO STATE UNIVERSITY

# Estimated Work Performance Data

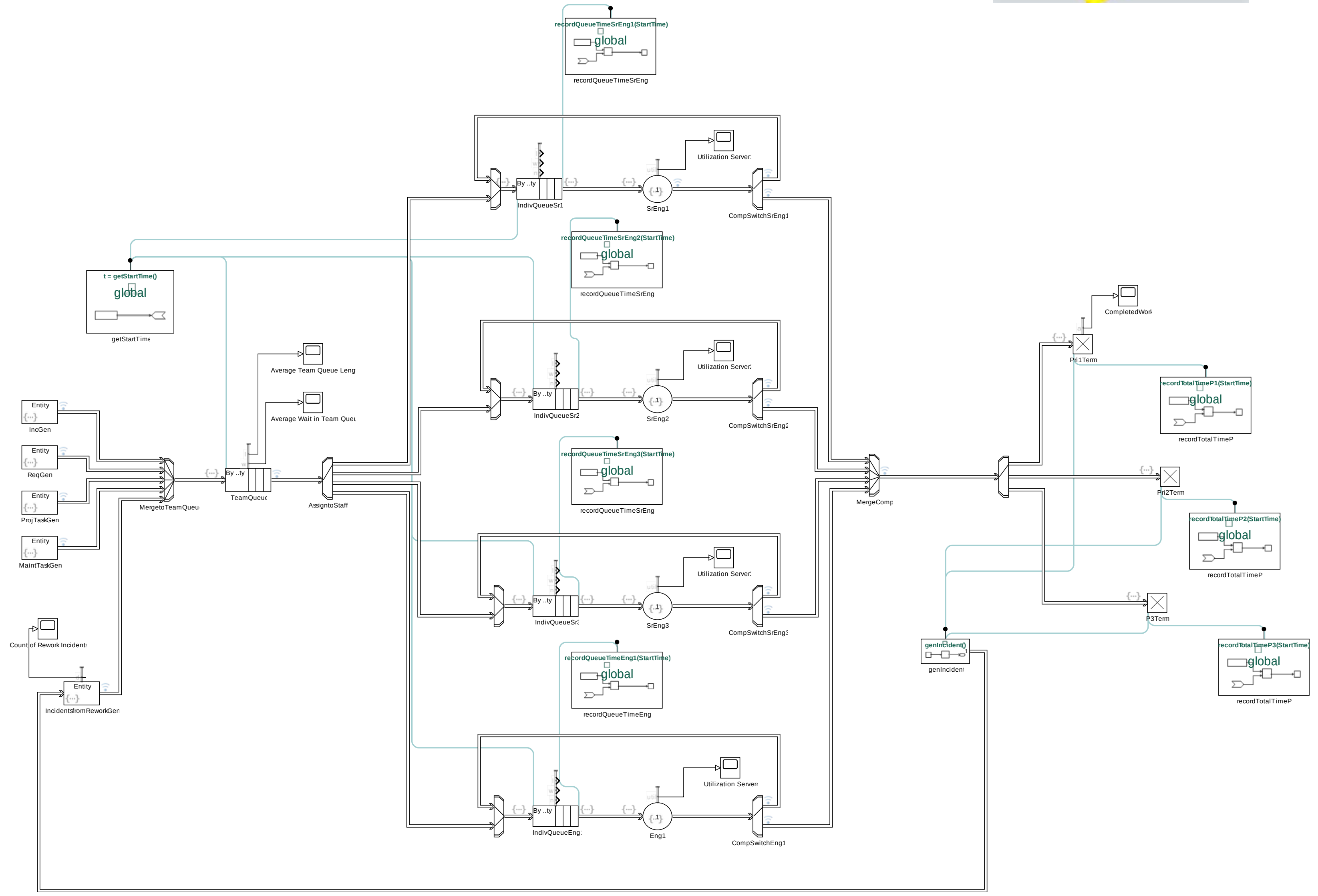
- Data determined through interviews with department managers:
  - Scheduled task arrival and closure data.
    - Est. as exponential (lambda of .5) for project tasks and mean of 1 for maintenance tasks.
  - Required skill level, set on entity generation: High 5%, Medium 25%, Low 75%.
    - Actual skill level is set at each resource (server).
  - Rework data, set on entity generation with base probability of .005.
  - Modification of rework probability and required service times, based on skill level data.
  - Service Times:
    - Required Service Time, est. as exponential (lambda of .15), calculated on entity generation.
    - Service Time, est. as exponential (lambda of .3), calculated as entity is “picked up” for work.
- Parameters requiring further justification:
  - Effects of managerial pressure and fatigue.
  - Initial error and restart rates.

# Dynamic Parameters

- Work stoppage rates calculated based on:
  - Required Service Time / Service Time.
  - An entity with remaining “Required Service Time” is returned to the queue.
- Rework generation rates:
  - Base .5% chance of generating an error (either rework or new incident), set on entity generation.
  - Modified by difference between required skill level and assigned skill level by rule. Examples:
    - If the “skill difference” is 2 (a high skill level resource working on a low skill task) the rework probability drops to .3%.
    - With a “skill difference” of -2 (a low skill resource working a high skill task) the rework probability increases to 1%.
- Required service time also increases by rule based on skill level differences. Examples:
  - A high skill resource working on a low skill task reduces the required service time (set on entity creation) by 20%.
  - A low skill resource working a high skill task increase the required service time by 20%.

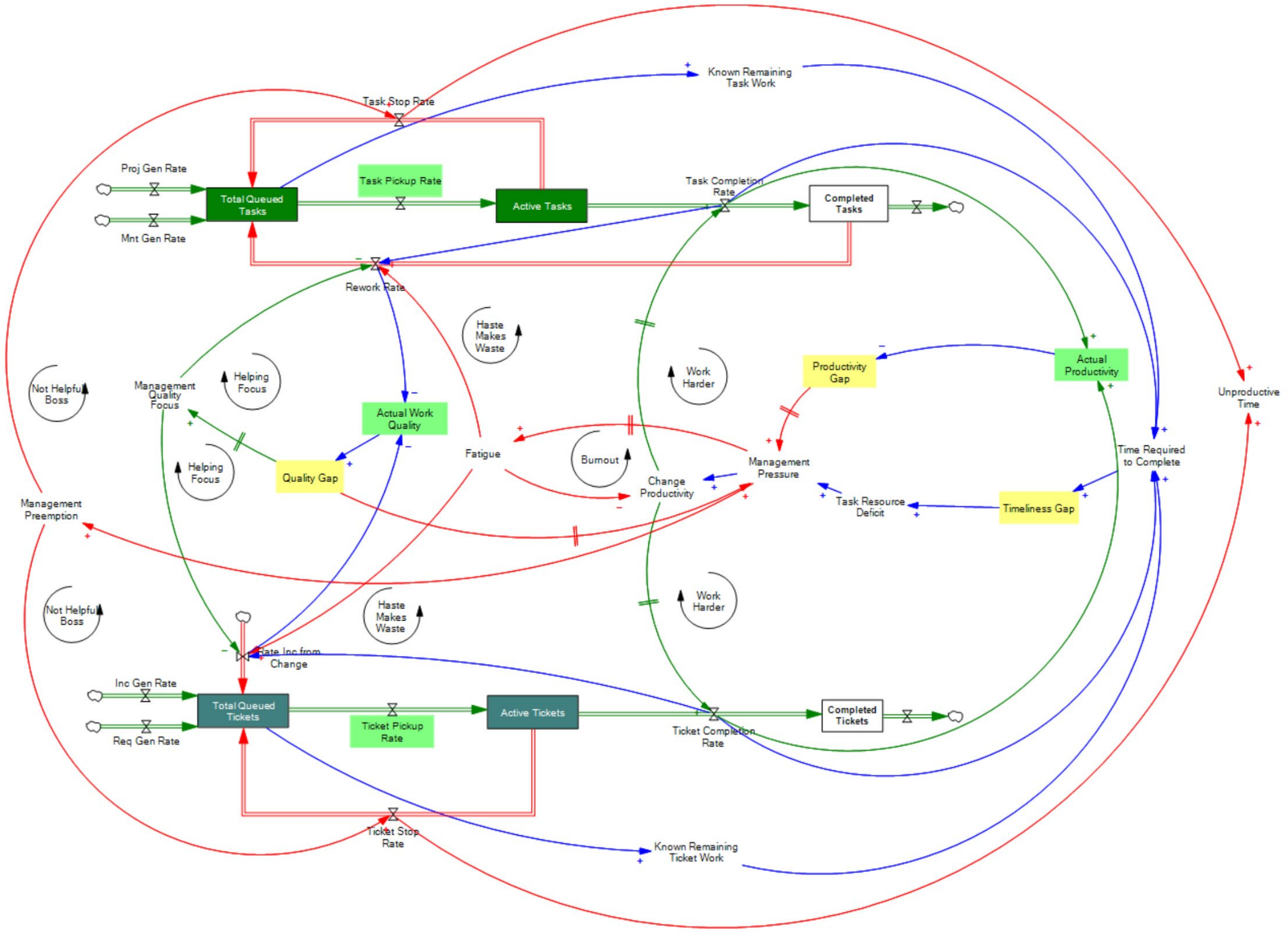
# Single-Team DES Model

- Five work generators with separate inter-arrival equations:
  - Incidents
  - Requests
  - Project tasks
  - Maintenance tasks
  - Incidents from work / rework
- A single team queue with individual queues per resource (“server”).
- Restart and error generation functions.
- *NOTE: model does not yet enable managerial escalation of priority while an entity is queued.*



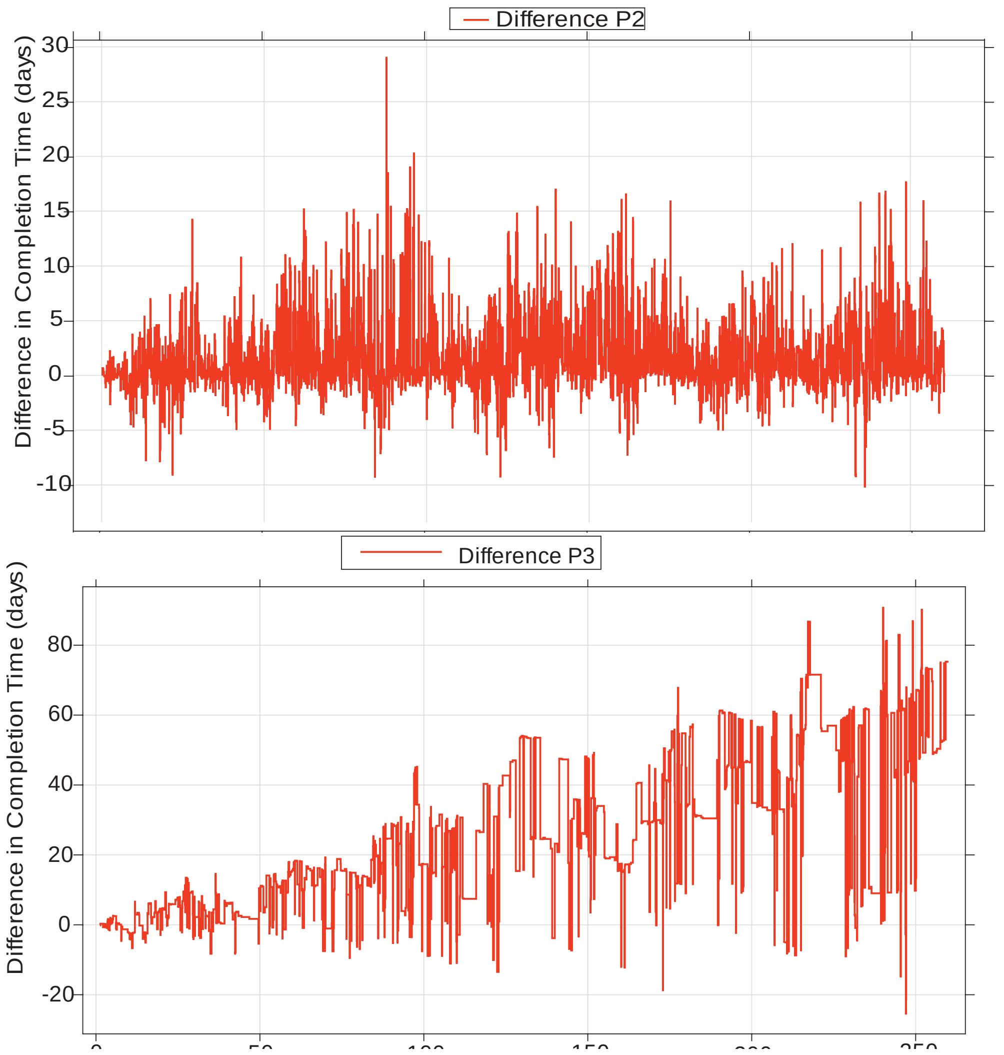
# Single-Team SD Model

- Modeled as two parallel flows of work:
  - Enable future differences in modeling of timeliness vs. responsiveness.
  - Enable differences in handling of rework vs. incidents resulting from operational changes.
- Influencing factors affecting both work flows shown in the center of the model:
  - Gaps in quality, productivity and timeliness.
  - Managerial pressure and quality focus as well as fatigue.



# Results after two iterations

- Key parameters shared between models:
  - Inter-arrival rates.
  - Work pickup and stoppage rates.
  - Error generation rates.
- After a full iteration of the DES and SD model, injecting SD results back into DES results in:
  - Significant increase in incidents from errors / rework.
  - Minimal to no impact to completion times in high priority work.
  - Significant difference in completion times of medium and high priority work.



# Observations from Simulation

- The team as modeled is not in control in either iteration:
  - Medium and high priority work items are handled reasonably quickly initially, but low priority items experience significant and growing delays.
  - With the impact of managerial pressure and fatigue added to a second iteration, performance on medium priority work degrades and low priority work delays grow increasingly long – reaching between 140-220 days by the end of the simulation run.
- 1/3 more work items were stopped and re-queued due to management pressure, and there was a very large increase in Incidents from Rework.
- This resulted in a 25% increase in work completed in response to incidents (because there were so many more of them) as well as an increase in all completion times by 18% for P1s and a 125% increase in P2s.
- For all intents and purposes, many P3s simply remained queued with 75% less completed during the simulation.

## Potential Automation Targets

- Reduce overall completion times through significant reduction or elimination of queuing times.
- Improve responsiveness for predictable work items — especially after hours where skilled staffing is often reduced or primarily on-call.
- Improve actual work quality due to increased accuracy, completeness, consistency, etc. of the work completed, with an associated reduction in the rate of rework generation (under the assumption of high-quality / well-tested automation).
- Reduce switching costs and cross-team coordination from either partial or full automation of complex, cross-team processes (such as server provisioning).
- Significantly increase various pickup rates for automated activities, which in turn drive improvements in the scale of work that can be completed in any given time period.
- Work that requires the sequential use of several tools / applications by the assigned resource to complete repetitive tasks, which could be automated together in sequence to improve individual productivity.

## Architecting a Software Defined Infrastructure Using MBSE

Coupling Discrete Event Simulation and System Dynamics

## Building on What’s Already There

- Technology and vendor support for on-premises automation is increasingly ubiquitous.
- In many cases, the capabilities exist in deployed infrastructure and tools even if they’re not leveraged, although they may require licensing.

cisco DEVNET

SOLARWINDS

servicenow. Solutions Products

Automation Engine

Getlabs

Microsoft® System Center Orchestrator

Delinea
Defining the boundaries of access

## What’s Generally Missing? The Orchestration

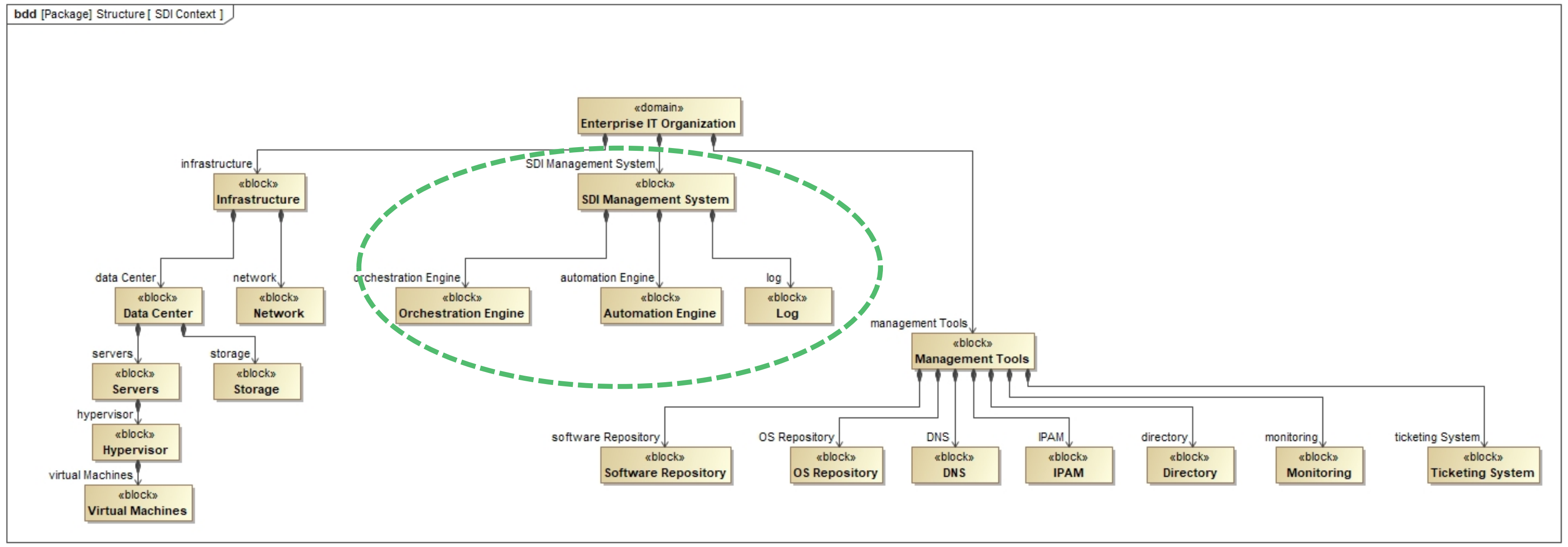
- The key component to SDI remains what will be referred to as the “integration engine”:
  - This is where the code runs that ties together the infrastructure and the tools to complete tasks.
  - These are often packaged as separately purchased tools, although can in some cases be bundled into existing tools.
  - These can include plug-ins to integrate with various infrastructure and tool products, or can rely on APIs published by those products.

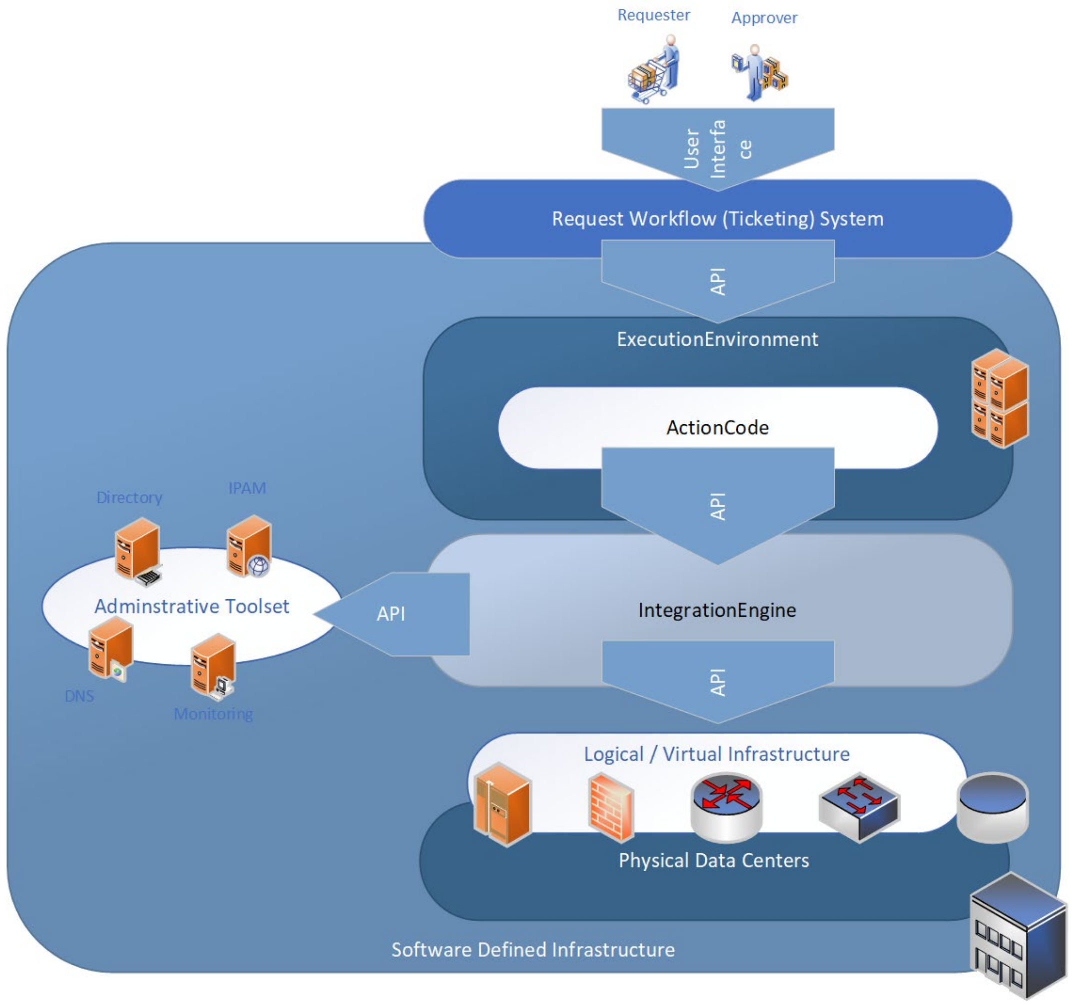
Terraform

VMware Aria Automation™

# Contextual Perspective

- Ultimately the SDI Management system provides a platform that enables integration of the infrastructure and tools.
- This integration requires custom code developed by the organization – some assembly is required.



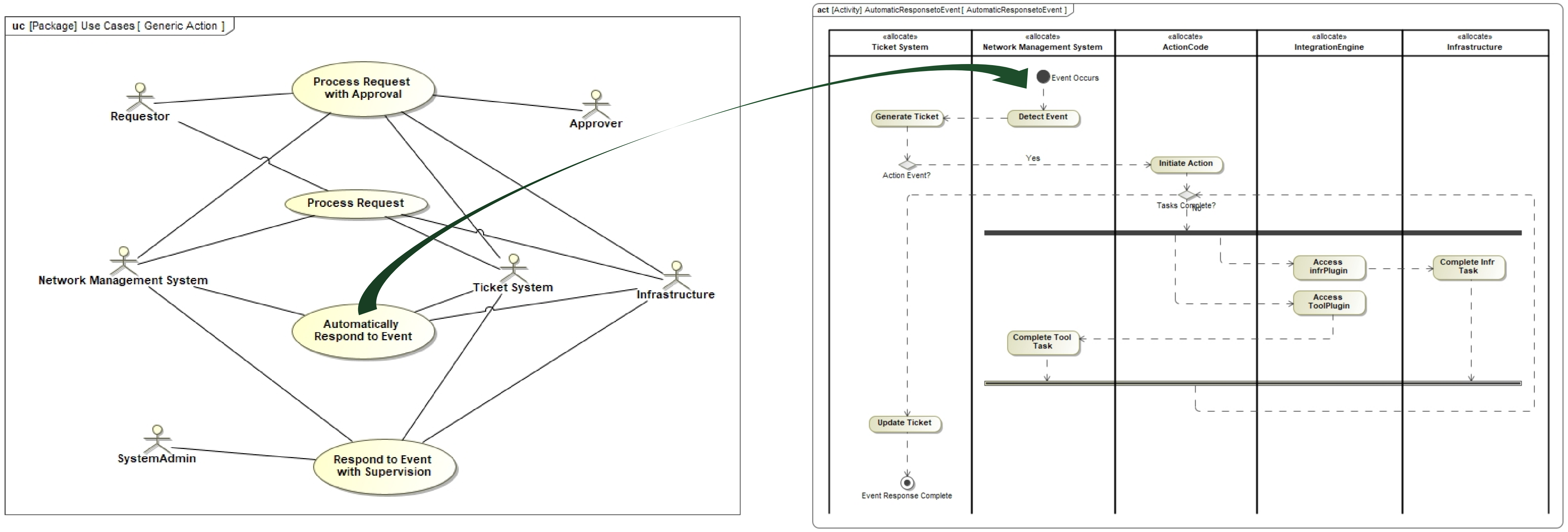


## Canonical Example: Server Provisioning

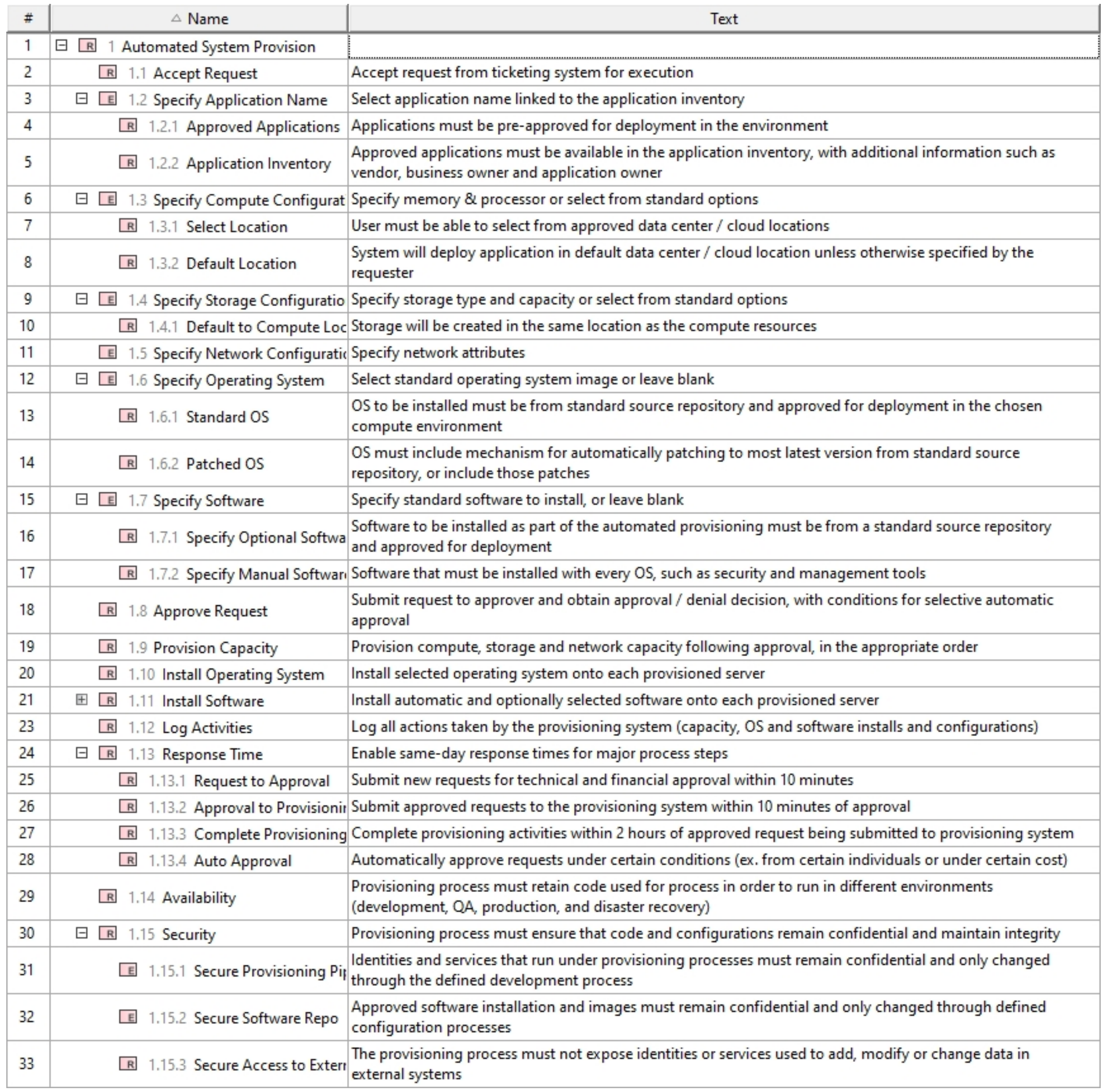
- This is the primary example used in the popular book on DevOps, *the Phoenix Project.*
- Server provisioning is the process of assembling the network, server, and storage capacity in a data center environment on which to deploy a server operating system and other software (e.g., middleware, DBMS and application software).
- Characteristics that make this a great example:
  - Cross departmental – multiple skill-based teams participate.
  - Leverages multiple technologies and tools.
  - Often required to provision multiple servers in one batch request.
  - A core service offering in the public cloud as IaaS.
  - Heavily studied as part of the DevOps practice of Continuous Integration / Continuous Delivery (CI/CD).
- At the start of the research, the case study organization spent on average 6 weeks to provision a set of servers for a new application.
  - Following automation of the process, this elapsed time was reduced to 2 hours from the approval of the request.

# Behavioral Perspective

- At the highest level, the SDI Management System must respond to a request or an event (incident or alarm).
- This can be completely automated or include a “human in the loop”.

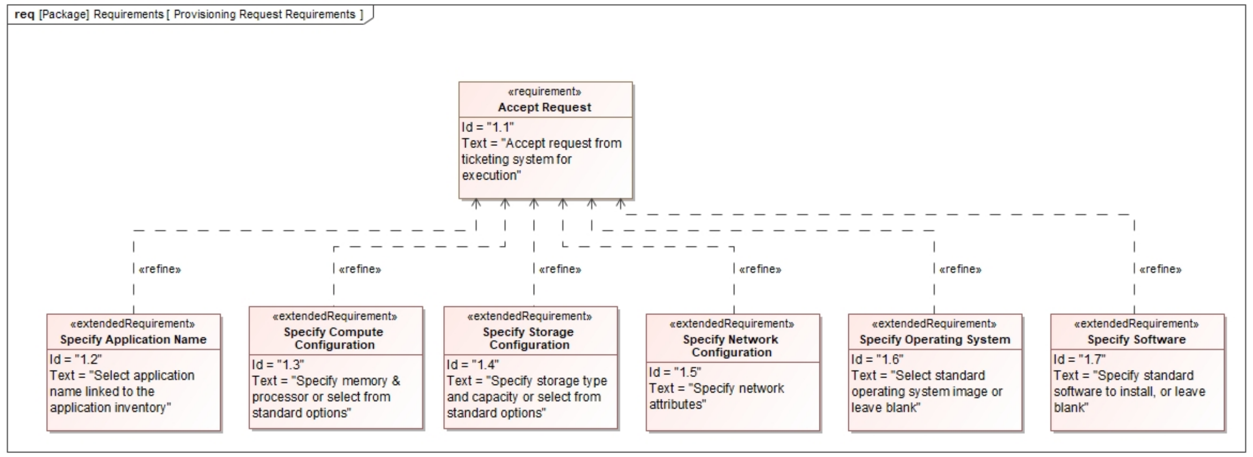

uc [Package] Use Cases [ Generic Action ]
Requestor
Process Request with Approval
Approver
Process Request
Network Management System
Ticket System
Infrastructure
Automatically Respond to Event
SystemAdmin
Respond to Event with Supervision
act [Activity] AutomaticResponsetoEvent [ AutomaticResponsetoEvent ]
«allocate» Ticket System
«allocate» Network Management System
«allocate» ActionCode
«allocate» IntegrationEngine
«allocate» Infrastructure
Event Occurs
Generate Ticket
Detect Event
Yes
Action Event?
Initiate Action
Tasks Complete?
No
Access infrPlugin
Complete Infr Task
Access ToolPlugin
Complete Tool Task
Update Ticket
Event Response Complete

# Provisioning Requirements



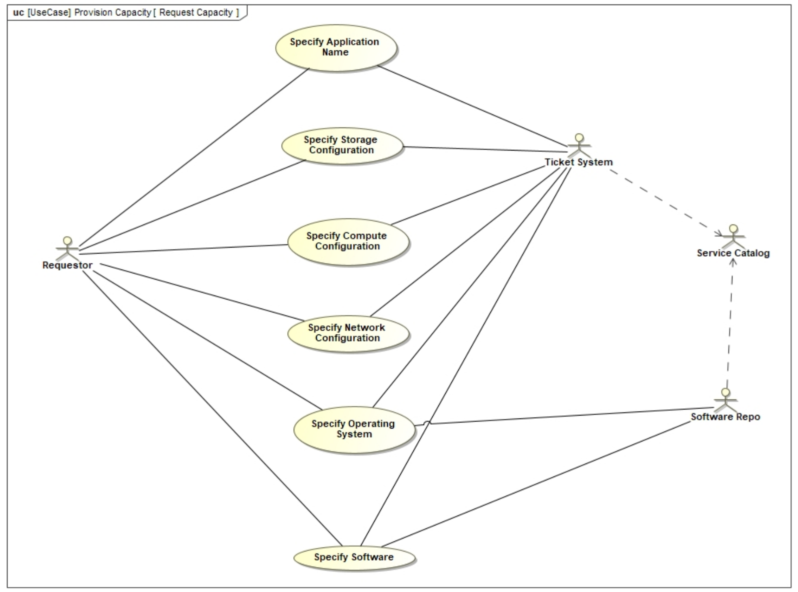
| # | Name | Text |
|---|---|---|
| 1 | 1 Automated System Provision | |
| 2 | 1.1 Accept Request | Accept request from ticketing system for execution |
| 3 | 1.2 Specify Application Name | Select application name linked to the application inventory |
| 4 | 1.2.1 Approved Applications | Applications must be pre-approved for deployment in the environment |
| 5 | 1.2.2 Application Inventory | Approved applications must be available in the application inventory, with additional information such as vendor, business owner and application owner |
| 6 | 1.3 Specify Compute Configurat | Specify memory & processor or select from standard options |
| 7 | 1.3.1 Select Location | User must be able to select from approved data center / cloud locations |
| 8 | 1.3.2 Default Location | System will deploy application in default data center / cloud location unless otherwise specified by the requester |
| 9 | 1.4 Specify Storage Configuratio | Specify storage type and capacity or select from standard options |
| 10 | 1.4.1 Default to Compute Loc | Storage will be created in the same location as the compute resources |
| 11 | 1.5 Specify Network Configurati | Specify network attributes |
| 12 | 1.6 Specify Operating System | Select standard operating system image or leave blank |
| 13 | 1.6.1 Standard OS | OS to be installed must be from standard source repository and approved for deployment in the chosen compute environment |
| 14 | 1.6.2 Patched OS | OS must include mechanism for automatically patching to most latest version from standard source repository, or include those patches |
| 15 | 1.7 Specify Software | Specify standard software to install, or leave blank |
| 16 | 1.7.1 Specify Optional Softwa | Software to be installed as part of the automated provisioning must be from a standard source repository and approved for deployment |
| 17 | 1.7.2 Specify Manual Softwar | Software that must be installed with every OS, such as security and management tools |
| 18 | 1.8 Approve Request | Submit request to approver and obtain approval / denial decision, with conditions for selective automatic approval |
| 19 | 1.9 Provision Capacity | Provision compute, storage and network capacity following approval, in the appropriate order |
| 20 | 1.10 Install Operating System | Install selected operating system onto each provisioned server |
| 21 | 1.11 Install Software | Install automatic and optionally selected software onto each provisioned server |
| 23 | 1.12 Log Activities | Log all actions taken by the provisioning system (capacity, OS and software installs and configurations) |
| 24 | 1.13 Response Time | Enable same-day response times for major process steps |
| 25 | 1.13.1 Request to Approval | Submit new requests for technical and financial approval within 10 minutes |
| 26 | 1.13.2 Approval to Provisionii | Submit approved requests to the provisioning system within 10 minutes of approval |
| 27 | 1.13.3 Complete Provisioning | Complete provisioning activities within 2 hours of approved request being submitted to provisioning system |
| 28 | 1.13.4 Auto Approval | Automatically approve requests under certain conditions (ex. from certain individuals or under certain cost) |
| 29 | 1.14 Availability | Provisioning process must retain code used for process in order to run in different environments (development, QA, production, and disaster recovery) |
| 30 | 1.15 Security | Provisioning process must ensure that code and configurations remain confidential and maintain integrity |
| 31 | 1.15.1 Secure Provisioning Pi | Identities and services that run under provisioning processes must remain confidential and only changed through the defined development process |
| 32 | 1.15.2 Secure Software Repo | Approved software installation and images must remain confidential and only changed through defined configuration processes |
| 33 | 1.15.3 Secure Access to Exterr | The provisioning process must not expose identities or services used to add, modify or change data in external systems |

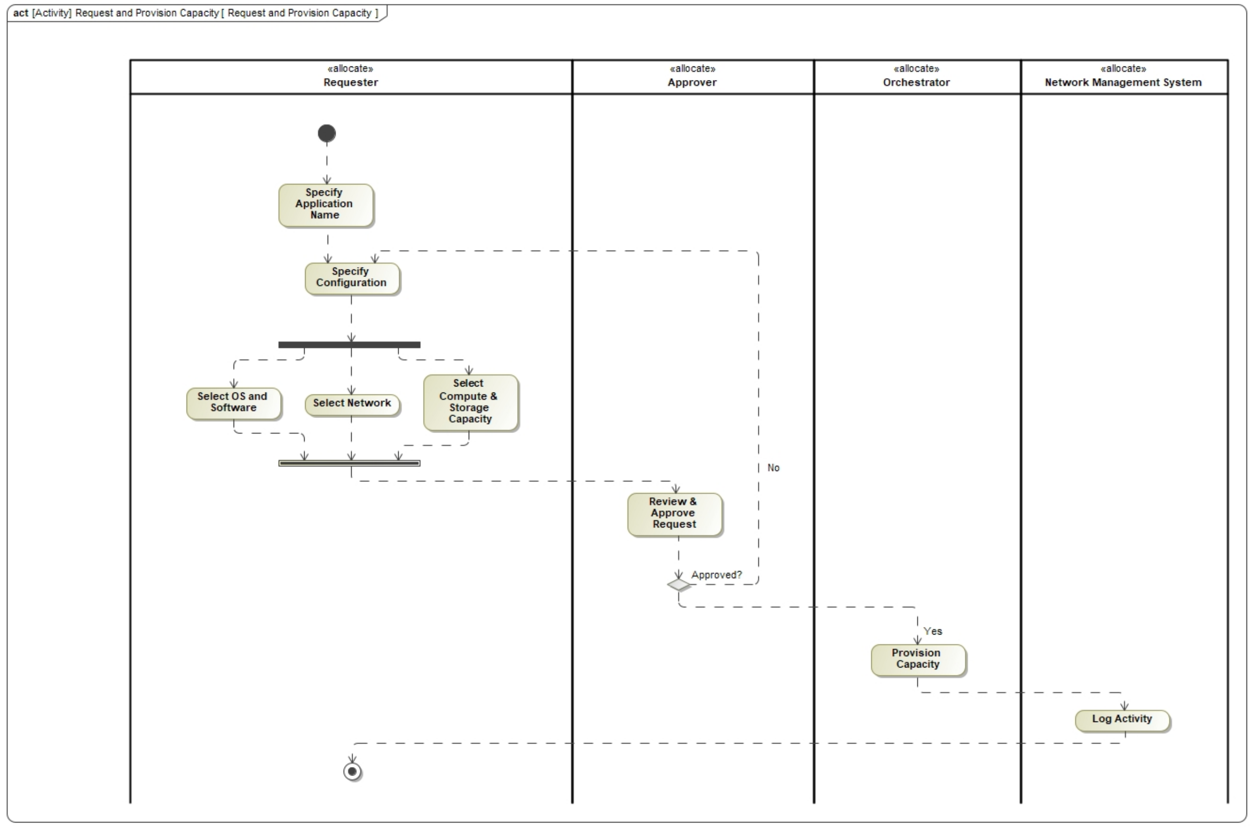
- Focus is on the tasks being automated by administrators to meet the business needs, not on the functional requirements of the administrators themselves.


req [Package] Requirements [ Provisioning Request Requirements ]
«requirement»
Accept Request
Id = "1.1"
Text = "Accept request from ticketing system for execution"
«refine»
«refine»
«refine»
«refine»
«refine»
«refine»
«extendedRequirement»
Specify Application Name
Id = "1.2"
Text = "Select application name linked to the application inventory"
«extendedRequirement»
Specify Compute Configuration
Id = "1.3"
Text = "Specify memory & processor or select from standard options"
«extendedRequirement»
Specify Storage Configuration
Id = "1.4"
Text = "Specify storage type and capacity or select from standard options"
«extendedRequirement»
Specify Network Configuration
Id = "1.5"
Text = "Specify network attributes"
«extendedRequirement»
Specify Operating System
Id = "1.6"
Text = "Select standard operating system image or leave blank"
«extendedRequirement»
Specify Software
Id = "1.7"
Text = "Specify standard software to install, or leave blank"

# Requesting Capacity

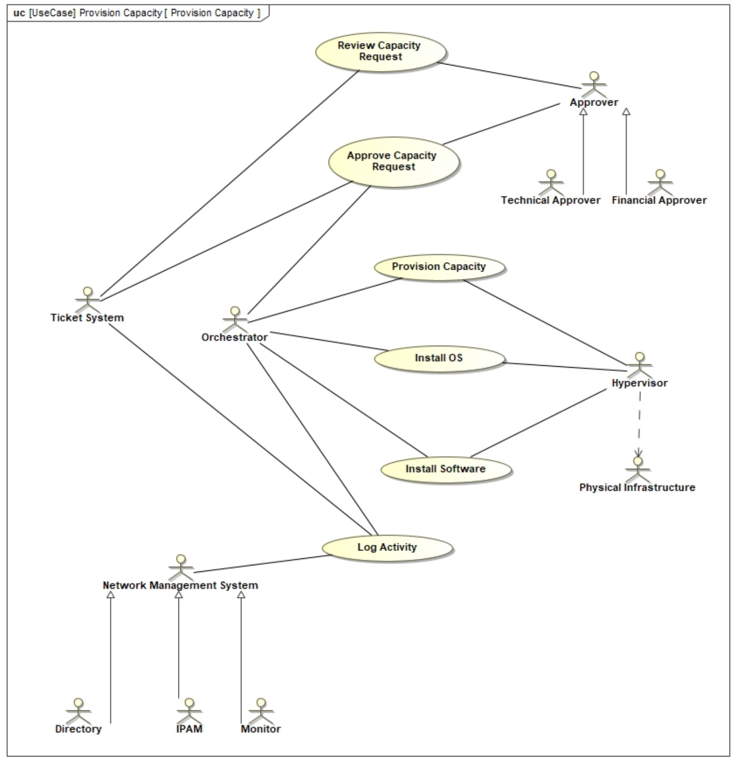
- Requesting capacity is often implemented as an activity requiring approval for two reasons:
  - It requires review by staff with the appropriate technical skills.
  - It must be deployed in the right place.
  - It must be logged appropriately in management tools
  - It may need to be iterated until complete and accurate.

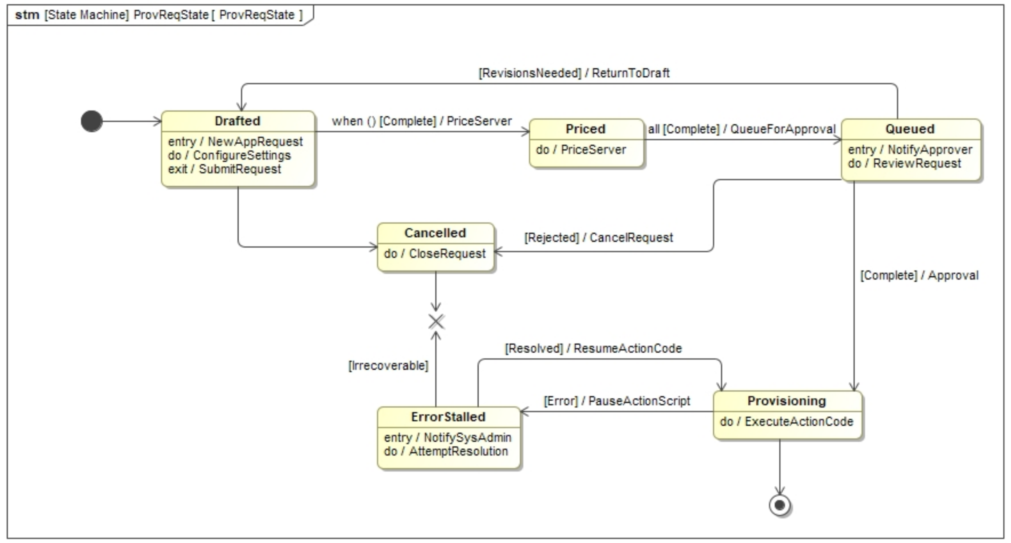

uc [UseCase] Provision Capacity [ Request Capacity ]
Specify Application Name
Specify Storage Configuration
Specify Compute Configuration
Specify Network Configuration
Specify Operating System
Specify Software
Requestor
Ticket System
Service Catalog
Software Repo



# Provisioning Capacity

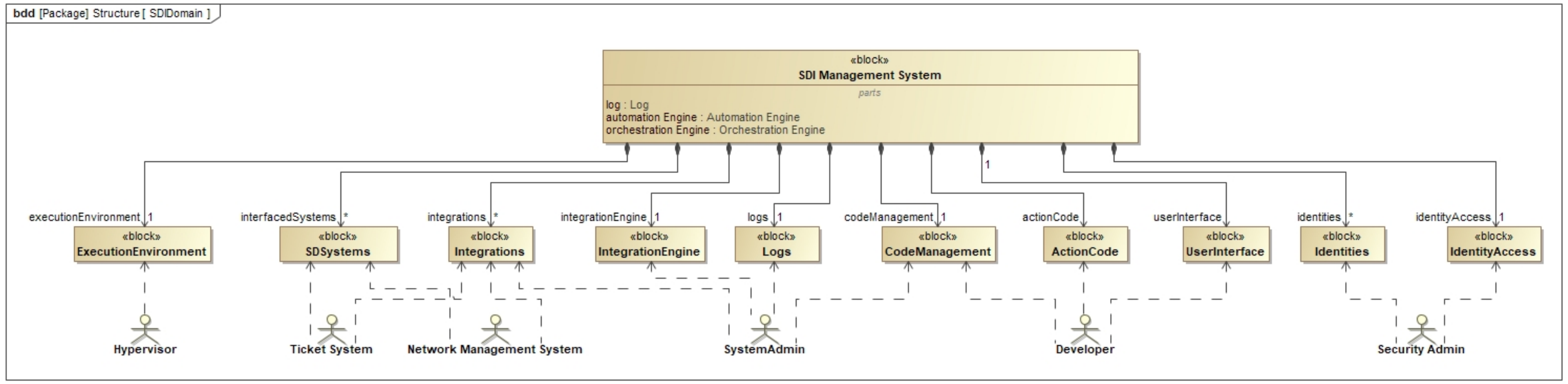
- Provisioning capacity is also implemented as an approval process, as it often entails incremental cost.
- The process of provisioning may be iterated for multiple servers as required for the intended purpose, so the application software may be deployed for use.

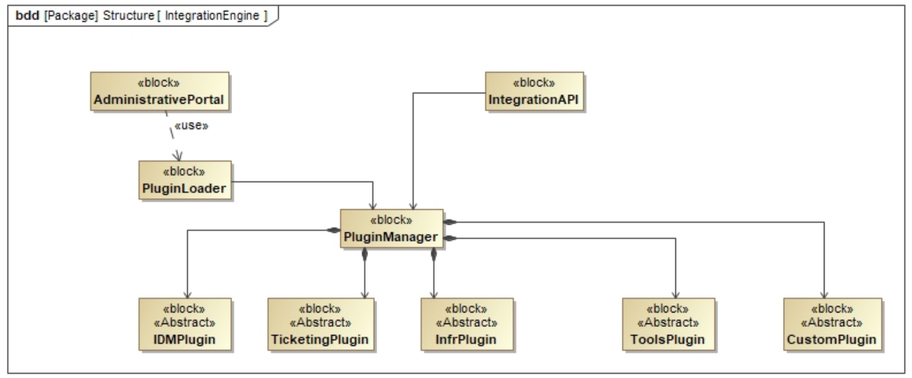




## Structural Perspective

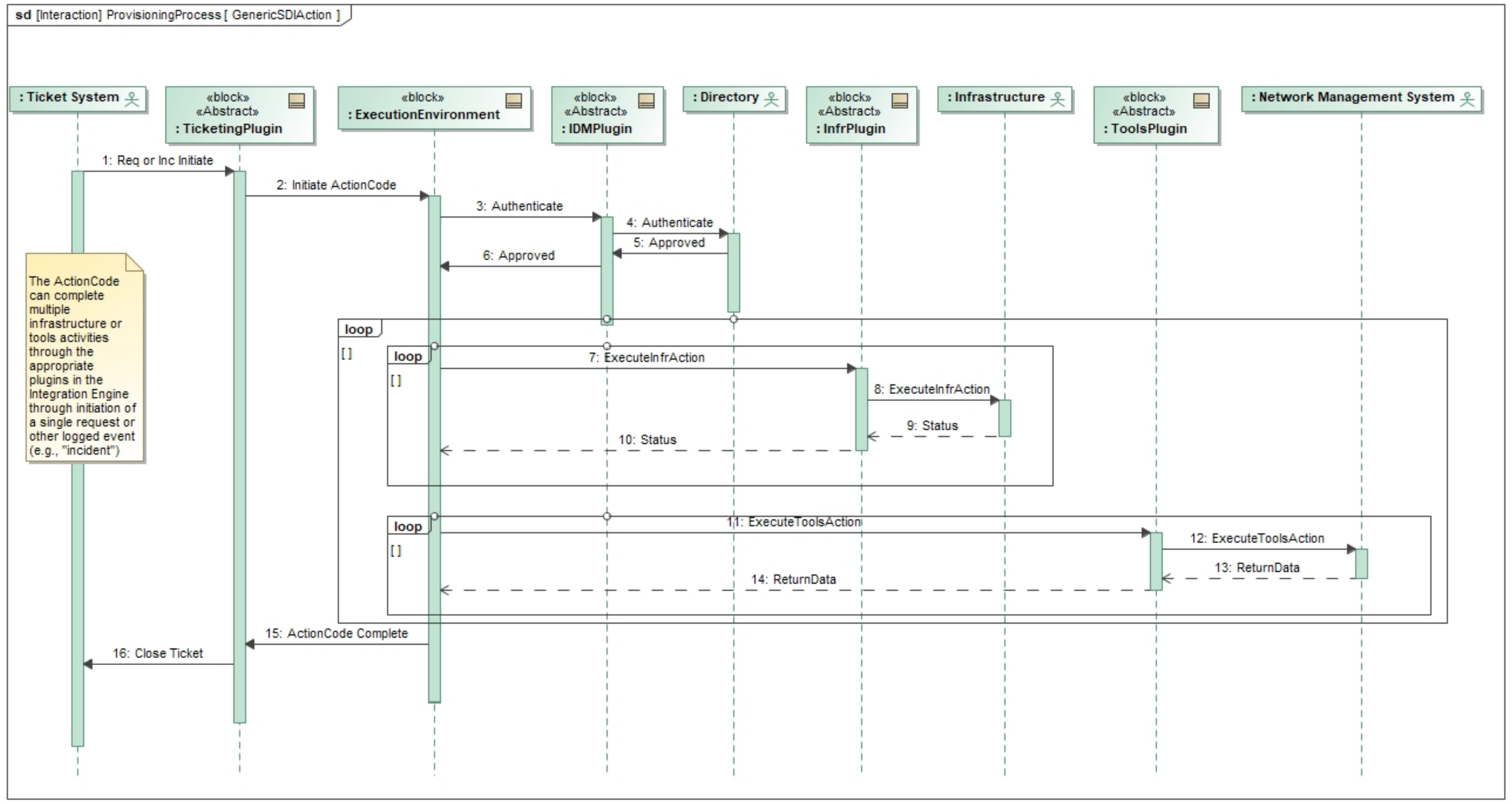
- The SDI Manager may be a coherent packaged system itself, or assembled from multiple packages, in particular:
  - The SD Systems in the domain diagram below are external – the underlying infrastructure and tools.
  - The Identity systems are often externalized, purpose-built systems due to their high security requirements.
  - Code management is often provided by external services, such as GetLabs.
- The integrations can be provided by the SDI Manager in a plugin model (as shown) or leverage the APIs built into the external infrastructure and tools.




bdd [Package] Structure [ IntegrationEngine ]
«block» AdministrativePortal
«use»
«block» IntegrationAPI
«block» PluginLoader
«block» PluginManager
«block» «Abstract» IDMPlugin
«block» «Abstract» TicketingPlugin
«block» «Abstract» InfrPlugin
«block» «Abstract» ToolsPlugin
«block» «Abstract» CustomPlugin

# Provisioning Sequence Diagram

- The code obtains the necessary credentials to complete the provisioning tasks from a directory.
- It then iterates through the provisioning of separate infrastructure components in order.
- It also iterates through logging its actions in management tools such as DNS, IP Address Management, etc.
- This continues until all components are deployed and configured.



sd [Interaction] ProvisioningProcess [ GenericSDIAction ]
: Ticket System
«block» «Abstract» : TicketingPlugin
«block» : ExecutionEnvironment
«block» «Abstract» : IDMPlugin
: Directory
«block» «Abstract» : InfrPlugin
: Infrastructure
«block» «Abstract» : ToolsPlugin
: Network Management System
1: Req or Inc Initiate
2: Initiate ActionCode
3: Authenticate
4: Authenticate
5: Approved
6: Approved
The ActionCode can complete multiple infrastructure or tools activities through the appropriate plugins in the Integration Engine through initiation of a single request or other logged event (e.g., "incident")
loop
[]
loop
[]
7: ExecuteInfrAction
8: ExecuteInfrAction
9: Status
10: Status
loop
[]
11: ExecuteToolsAction
12: ExecuteToolsAction
13: ReturnData
14: ReturnData
15: ActionCode Complete
16: Close Ticket

# What’s next?

- Elaboration of the architecture will continue to refine the next level of detail outlining:
  - The broad types of infrastructure needed to complete the tasks.
  - The various tool capabilities required to enable full configuration and availability of the provisioned capacity & software to the requester.
- The intent is to provide sufficient detail for IT leaders to compare their currently deployed capabilities to those required, in order to identify gaps to be closed and – ultimately – costs.

## Preliminary Conclusions and Future Research

Expanding simulation and architecture

## Preliminary Conclusions

- Potential improvement areas that can be influenced by automation:
  - Reducing completion times by reducing queue times, either from stop and restart or from transfers.
  - Improving responsiveness through triggering automation on certain events (e.g., incident or request arrival).
  - Improving actual work quality / reducing the rate of rework and the generation of new incidents by reducing or eliminating human error and variability.
  - Increasing pickup and completion rates for high-volume / repeatable work items.
  - NOTE: automation best addresses *high-volume* and *repeatable* work.
- Additional options for improvement are suggested, but not the focus of this research:
  - Ensure close coordination between project and functional leaders on work management and priorities.
  - Limit the number of projects in process through a governance and prioritization function.
  - Separate operational and project responsibilities between different staff within each department.
  - Hire and (and train) multi-skilled resources.
  - Create cross functional teams to prevent transfers of work between department queues.

## Revisiting Research Questions

- RQ1: What are the basic components and capabilities of an on-premises SDI system?
  - This question is partially addressed in Chapter 4 on the Architectural Framework.
  - Further effort is needed to expand and refine a product-agnostic architectural framework and connect it with extant research.
  - A particular focus area will be in identifying any research available supporting the underlying technical architectures of the cloud providers themselves, rather than in how customers can or should leverage those services.
- RQ2: Should healthcare providers implement on-premises SDI?
  - This question is addressed within the context of a single team, and research is largely complete and documented in the Literature Review section.
  - The preliminary answer to the question is a qualified "yes," insofar as it relates to identifying clear potential benefits.
  - The question as to whether the investments required to build the SDI itself are worth those benefits, given the constraints of healthcare provider IT, remains a topic for ongoing research.
- RQ3: How should provider organizations proceed to implement on-premises SDI, if at all?
  - This question is partially addressed with the decision framework for IT leaders.
  - The synthesis of completed work in Chapters 3 (that attempts to identify and prioritize targets) and 4 (that addresses components and prerequisites) will enable the creation of a proposed road map for implementation of on-premises SDI.

# Publications

- *Using Hybrid System Dynamics and Discrete Event Simulations to Identify High Leverage Targets for Process Improvement in a Skill-based Organizational Structure.*
  - Submitted to IEEE's 18th Annual International Systems Conference (SysCon) scheduled in April 2024, focused on the single-team simulation models built to partially answer Research Question 2.
- Additional papers planned>
  - A paper outlining the utility of MBSE in the creation of IT system architectures, utilizing the SDI architectural framework as an example.
    - While UML is commonly used in the development of software, the use of SysML represents an opportunity for use in IT infrastructure, where physical components and logistics come into play.
    - This paper is targeted for submission in mid-2024 in a systems engineering focused journal (INCOSE's *Systems Engineering*) or IT-focused journal (IEEE's *IT Professional*).
  - A paper proposing the SDI architecture and its applicability to enabling DevOps-style benefits within on-premises IT environments.
    - The intent is to make the results of this entire body of research available to IT leaders, providing the rigorous recommendations on how to proceed that are currently unavailable.
    - This paper is targeted for submission in late-2024 in an IT-focused journal (*Journal of Information Technology*).

## Expanding Simulation to Two Teams

- Adding a second team to the SD and DES models introduces additional rules for managing work passing between them:
  - Each team supports mixed work types – e.g., both unscheduled and scheduled – that arrive at differing rates.
  - Each team supports multiple concurrent projects / products at different stages of planning and execution, as well as multiple concurrent incidents and requests.
  - Active work in progress can be returned to the queue for a new reason: 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 work item.
  - Reassignments between teams require coordination to ensure efficient completion and is an indicator of a higher level of overall complexity.
  - The models only depicts interactions between two skill-based teams – interactions become much more complex as additional teams become part of the work process.
- New dynamics are expected that could introduce new criteria for identification and prioritization of automation targets.

# Ongoing Data Analysis

- The comparison of baseline simulation results to actual team performance data to validate remains in process.
- The refinement of the SD models in particular to better justify the internal parameters regarding the effects of management pressure and fatigue is envisioned.
- Sensitivity analysis of overall completion times to restart and error rates, and upstream influences in the SD model, to assist with the prioritization of automation targets.

## Future Work Areas

- Determine the feasibility of predicting the scale of benefits from automation.
  - For example: decreasing the queue time in this activity by x% will result in reduction of completion time in that activity by y%.
- Expanding literature review to determine availability of research on cloud infrastructure.
  - To date, most literature seems focused on how to *use* it, not how to *build* it; this seems to be left to the various vendor documentation.
- Expanding literature review to bolster research underpinning automation as a process improvement tool.
- Develop the implementation framework:
  - Propose a suggested roadmap for the implementation of SDI, including technical and process considerations.
  - Propose a decision model for use case identification and prioritization, taking into account both potential benefits as well as implementation cost and complexity.
  - Tie together both the benefits of automation with the anticipated costs (based on the architecture) into an overall decision model for IT leaders.

## Questions?

# Thank you

WALTER SCOTT, JR.
COLLEGE OF ENGINEERING
COLORADO STATE UNIVERSITY

SYSTEMS ENGINEERING
COLORADO STATE UNIVERSITY