Modeling and Simulation of an IT Operations Command Center

Sanjit Mehta
Tata consultancy services
54-B, Hadapsar Industrial Estate,
Pune, MH, 411013, India.
Sanjit.mehta@tcs.com

Veerendra K Rai
Tata consultancy services
54-B, Hadapsar Industrial Estate,
Pune, MH, 411013, India.
veerendrak.rai@tcs.com

Rahul Roy
Department of Management Information Systems, Indian Institute of Management Calcutta, DH Road, Kolkata – 700104, India.
rahul@iimcal.ac.in

Abstract

This paper models and simulates IT operations command center. It outlines the activities performed by the command center and describes the model and modeling approach to simulate these activities. The paper appreciates the discrete and continuous aspects of modeling the operations of the command center. The purpose of this model is to predict the behavior of a command center engagement over time. The model is used to test and understand the effect of scenarios on the engagement. This provides a good governance framework for the engagement as governance is key to success and failure of a service engagement. Policies are defined and analyzed to mitigate impact of scenarios and/or achieve objectives thus implementing a feedback loop monitoring system for command center operations management.

Keywords: IT operations command center, governance, service management, service engagement.

1. INTRODUCTION

ITIL (2013) defines services as a means of delivering values to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks. Services are enabled through configuration items. A configuration item could be any component required to deliver IT services. Examples of configuration items are – hardware, software and process documentation. The goal for IT services is to fulfill business objectives, add value and contribute to business. The hierarchy from IT infrastructure to business objectives is described briefly as follows.

- IT Infrastructure layer: contains IT infrastructure consisting of mainframes, servers, storage and networks etc.
- Configuration layer: this layer enables services and consists of a set of applications and corresponding documentation.
- Services layer: contains the set of services, which when executed fulfill the business objectives they have been designed for.
- Business objective layer: contains set of business objectives defined by the top management of the organization and, which is subject to change with change in the business environment.

Any disruption in services is unwarranted and may have varying levels of negative business impact. To ensure provision of services to best meet business objectives, formal controls termed as 'Service level agreements' (SLA) are defined and assessed on a regular basis Goo *et al* (2009). Penalty is levied on deviation of services from agreed levels. Continuous and consistent operation of configuration items is mandatory to provision of services within defined service level agreements.

ITIL (2013) terms these activities as IT operations and control, which are aimed at ensuring that the technology required to deliver and support services is operating effectively and efficiently.

The monitoring and control loop is at the center of IT operations and control (ITIL, 2013). Services, activities and configuration items may be monitored proactively or reactively. Any deviations from defined norms observed in service operations may be resolved by the team or forwarded to responsible teams. Corrective actions (referred to as control) ensure continuous operations of configuration items. A delay in implementing corrective actions may lead to disruption in services and consequent business impact.

In this paper, we model an IT operations and control engagement, hereafter termed as a command centre. Command centre provides a centralized hub for monitoring and control of configuration items which enable provision of services to meet business objectives. Configuration items may be monitored and controlled remotely or non-remotely depending on context. For example, certain tasks such as hardware upgrade, disk change would require physical access to configuration items whereas applying software patches, running batch jobs may be executed remotely. Command centre teams refer to an extensive repository of 'Standard Operating Procedures' (SOP), process documents, checklists and knowledge articles to perform activities pertaining to monitoring and control. Events, incidents or corrective actions beyond the scope of a command centre are forwarded to incident management, problem management, change management or other service delivery processes.

For the model described in this paper, we have identified a list of tasks based on the activities performed within a command centre engagement within our organization (Table 1). We describe the model and modeling approach to simulate these activities. The purpose of this model is to predict behavior of a command center engagement over time. The model is used to test and understand the effect of scenarios on the engagement. Policies are defined and analyzed using the model before implementation to mitigate impact of scenarios and/or achieve objectives. System dynamics approach facilitates holistic assessment of scenario impact and/or policy decisions. The model therefore provides a governance tool to execute a command centre engagement in steady state, wherein it consistently meets desired SLA and continually improve its performance to achieve business objectives.

Table 1.Command centre activities

Category	Task	Description
Alert Driven	Application Alert Resolution	Resolution of alerts generated by configuration items.
	Batch Alert Resolution	Resolution of failures, delays and dependencies in batch processes.
	Infrastructure Alert Resolution	Resolution of events generated by infrastructure elements.
Proactive & Hygiene	Ready For Business Checks	Proactive health check of configuration items before start of business.
	Database Maintenance	Planning, analysis and scheduling maintenance activities of databases.
	Connectivity Checks	Checks on application connection with systems and external entities.
Planned Event Management	Release deployment	Deployment of releases and checks to be carried out after deployments based on run books.
	Planned Event Coordination	Coordination of planned events and checks carried on configuration items.
	Outage Management	Initiation, coordination, and documentation for post mortem of major incidents due to system failure.
Service Delivery Management	SLA and Metrics based Governance	KPI and SLA definition to monitor and improve services.

Section 2 outlines a meta-model for command centre operations which provides a logical abstraction of the model. Section 3 describes the modeling technique, modules and elements of the model. Section 4 shows results obtained from simulation of the model for a particular scenario and policy configuration. Section 5 concludes the paper discusses further work towards exploiting the model for business benefits.

2. Activities and Processes

2.1 The activities

Table 1 lists the activities, which are performed within a command centre engagement in our organization. Based on our analysis of this engagement, practitioner's experiences and domain knowledge we abstracted the activities into a singular concept – 'Ticket'. A ticket may be defined as a unit of work, which may be performed by one or more resources in accordance with the processes defined for addressing a given type of ticket. We identified and modeled 5 types of tickets (Refer – Table 1).

a. **Alerts** – These pertain to alert-driven tasks, i.e. resolution of alerts generated by configuration items.
b. **Batch incidents** – These include resolution of failures, delays and dependencies in batch jobs.
c. **Scheduled tasks** – These span all tasks which are scheduled and/or repetitive in nature. Example – planned checks, maintenance activities, storage and backup.
d. **Delegated tasks** – These pertain to tasks which may be delegated to the command centre from other units (business or IT). This ticket type also includes certain user incidents, service requests and access requests which do not require extensive solutions and can be resolved using SOP and knowledge articles.
e. **Release** – This ticket type includes all release and deployment tasks and post-deployment checks.

It is to be noted that these ticket types may be modified based on context. Classification of ticket data along these types (Ticket type distribution) for initialization of the model is based on definitions provided for each ticket type. In addition to ticket type distribution, following data points pertaining to tickets are required for model initialization and simulation.

a. **Ticket volumes** – Volume distribution (Rate of arrival of tickets) for each type.

b. **Ticket priority distribution** – Priority distribution (Immediate, High, Medium, Low) for each ticket type. Ticket priority is the product of its urgency and level of impact as given in Table 2.

c. **Ticket effort distribution** – Effort in person-hours required to address a particular ticket type.

d. **Service level agreements** – Defined levels of service for metrics such as resolution time, response time, and release deployment time, etc. for each ticket type and priority combination. For example – time taken to resolve an immediate priority alert must not exceed 10 minutes.

Table 2.Ticket prioritization

		Impact		
		High	Medium	Low
Urgency	High	1 (Immediate)	2 (High)	3 (Medium)
	Medium	2 (High)	3 (Medium)	4 (Low)
	Low	3 (Medium)	4 (Low)	4 (Low)

2.2 The process

The command center model consists of three core processes. Monitoring is a continuous process wherein the team is required to monitor and detect any deviation from normal, referred to as 'Event' during operations of configuration items. If discrepancies are detected, a ticket is raised and corrective actions implemented. Service desk responds to tickets delegated by other units as well as other types of tickets. Response implies acknowledgement, data validation and routing the ticket to responsible personnel. Resolution team implements corrective actions and/or executes required tasks based on available SOP and knowledge articles for a given type of ticket.

The model also consists of a set of supporting processes. Knowledge management is a set of activities to create, document, validate and add knowledge articles to repository. Knowledge database is repository of documentation such as SOP, process documents, checklists and knowledge articles referred by the resolution team. The repository constantly evolves over time as more knowledge is garnered with respect to configuration items, tickets, ticket types and their resolution. Processes are improved through enhancements, task automation and virtualization. Process improvement reduces ticket volume, effort required to respond to and resolve a ticket and saves costs in long-term. In this paper, we model reduced ticket volume as a result of process improvements. Resource management models resource availability and utilization for processes described above. Costs associated with processes are modeled through a set of cost parameters.

Tickets may be scheduled, raised or detected through monitoring. Apart from configuration items, batch jobs are monitored, which result in batch incidents due to failures or delays. An event, detected through monitoring may be an alert or batch incident. An event requires action within a certain timespan. If the event is not detected due to lack of resources or attention, then it may (or may not, depending on the type of event) cause an incident post a certain period of time. An incident is a disruption in service and it has far-reaching effects in terms of costs to restore the system. Apart from events (alerts and batch incidents), other ticket types such as scheduled, delegated tasks and service requests are assigned to the team based on a schedule, from other teams, or from end users.

Tickets are acknowledged by the service desk and sent to teams for resolution. Tickets are resolved with assistance from knowledge base. A knowledge base or known error database (KEDB) is a repository of knowledge articles which consist of instructions on steps to resolve a particular type of ticket. If an article is not present, the responsible team member may either solve the ticket, if competent or escalate it to a higher level. After resolution of the ticket a knowledge article pertaining to steps taken to resolve the ticket is created, validated and added to the repository. This constitutes a negative

feedback loop, shown in figure 1, which increases the number of articles in knowledge repository over time. The rate of increase in knowledge articles decrease over time. Process metrics are computed to capture process behavior. Metrics include – Ticket response time, Ticket resolution time, Knowledge base effectiveness, Events detected and Events which may cause potential incidents. These metrics are part of the model and are detailed in subsequent sections.

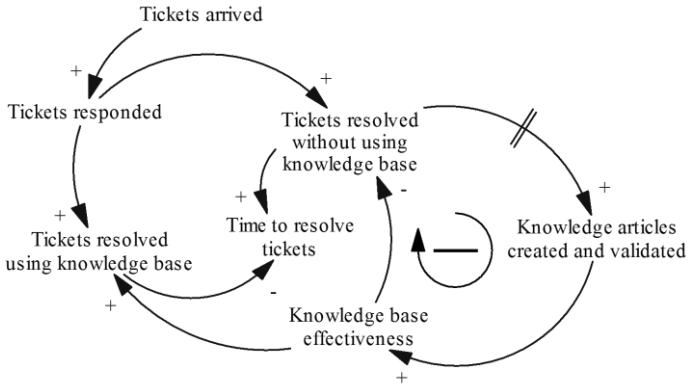

Figure 1. Knowledge base effectiveness loop

3. SYSTEM DYNAMICS MODELING

In addition to understanding system dynamics modelling and simulation approach to appreciate the behavior of complex systems over time given in Forrester (1969) and Sterman (2001) we also need to apprise ourselves with the latest state-of-art tools (Vensim, 2013 and Stella, 2013). A system dynamics model could be logically divided into sectors, wherein each sector represents a part of the model. A concept central to our model is 'Arrays'. Arrays in system dynamics allow a particular variable to represent multiple concepts. In the context of our model, 'Ticket type' is an array. It represents – 'Alerts', 'Batch incidents', 'Scheduled tasks' and 'Delegated tasks'. Arrays allow a user to disaggregate a particular variable in to less abstract concepts.

3.1 Purpose and scope of system dynamics modelling

IT service operations and control has been traditionally modelled as discrete-event based simulation. SYMIAN is a discrete-event simulation tool for incident management Bartolini *et al* (2008). It uses a queuing system to model incidents. Time to resolve an incident is computed based on time spent per incident as well as waiting time per ticket based on employee's availability. Assuncao *et al* (2012) use simulation to evaluate the impact of incident dispatching policies for incident management. As described in section 2, incidents, as tickets, are classified along type and priority. Policies are defined to select and resolve incidents from the queue. An example policy would be to solve incidents of immediate priority as and when they arrive, while de-prioritizing the incident-at-hand if it is of lesser priority. Since SLA for immediate priority incidents are more stringent, therefore prioritization would improve compliance. This and other policies are tested through simulation to evaluate their impact in terms of effort required per ticket and SLA compliance. Lee *et al* (2007) developed a system dynamics model for incident and problem management. Problem management is a service delivery process for root cause analysis and resolution of errors, which cause incidents. The purpose of this paper is to identify variables in the system which would cause significant fluctuations in overall system behavior.

Based on our literature review and domain knowledge, we established a set of feedback loops for command center as service delivery and supporting processes are inter-twined through cause and effect. These feedback loops and process inter-relations are not captured by available literature on discrete-event IT service operations simulation. We use system dynamics to capture these relationships and complex feedback structure. The available literature on system dynamics modelling of IT service operations is sparse and abstract in nature. It does not absorb requisite variety, which limits its

application in a real-world engagement. To improve the accuracy of our model in terms of structure as well as behavior, we uniquely use system dynamics as the combination of discrete and continuous simulation tool. We exploit capabilities of state-of-art system dynamics simulation packages (Vensim and Stella) to accommodate variety. Since, the lifecycle span of a ticket is expressed in minutes therefore we use a low time-step (1 minute) for simulation. Higher time-steps would generate inaccurate behavior. For example, if the resolution time for an alert is 10 minutes and time-step is 30 minutes, the system would use resolution time as 30 minutes. This would consequently affect SLA compliance, total effort spent, resource utilization and costs. Apart from modelling the feedback structure, we use system dynamics modelling to enable functionalities for what-if analysis and policy design. We developed a comprehensive model to accommodate a large set of scenarios and policies, which users require to govern a command center engagement thereby extending its applicability. In addition to exploiting power of latest system dynamics simulation tools we also need to sensitize ourselves to the domain of IT outsourcing, production support and production management. Rai (2013) constructs a viable framework for production support based on Viable System Model (VSM) of Stafford Beer (Beer, 1985); Espejo and Harnden (1976). Rai (2012) outlines a system of requisite knowledge for service engagement transformation. Rai *et al* (2010) discuss framework for portfolio management at Offshore Development Center (ODC). Rai and Mehta (2012) outline a systems thinking based methodology for assessing the As-Is state of a service engagement so that policy intervention could be made to ensure service engagement reaches the desired (To-Be) state. Rai and Swaminathan (2010) describe a system of systems approach to construct a framework for program management.

The next section describes the model in detail as a set of inter-related processes which constitute the overall feedback structure.

3.2 Command centre model

This section provides detailed description of the command centre model. We have divided this section into multiple subsections each representing a sector in the overall model. The table below provides an overview of all sectors.

Table 3.Command centre model sectors

Sector name	Sector description
Control data	This sector is a container for all data parameters to initialize the model.
Ticket generation and response	This sector models processes required to generate and respond to tickets. Tickets are generated using input data. Ticket response requires associates to acknowledge tickets and assign them for resolution.
Ticket resolution	This sector models the ticket resolution process. Tickets are resolved by associates (or escalated) with or without assistance of a knowledge base.
Release management	This sector models the release management process. Since a release is different from tickets, it has been modeled separately.
Monitoring and event management	This sector models monitoring for a set of configuration items. Configuration items are monitored to detect and resolve any abnormal behavior (event). An event may pertain to an alert or a batch incident which is resolved through the resolution process.
Knowledge management	This sector models improvement in knowledge base through creation and validation of knowledge articles.
Resource management	This sector models the availability and utilization of resources for various processes.

The model consists of a set of arrays. The purpose of these arrays is to represent multiple concepts within a particular variable. Arrays may be construed as classes of which array elements are instances. This enables disaggregation of variables into less abstract concepts. For the command centre model, we described a set of arrays to represent ticket types, ticket priorities, ticket resolution type (with or without assistance of knowledge base) and ticket occurrence frequencies. Any

variable, depending upon its type and usage may have zero or more combinations of elements of each array. Following table lists the arrays as well as their elements defined in the model.

Table 4.Arrays in command centre model

Array: Ticket type	
Array elements (Refer Section 2.1)	
Alert, Batch, Scheduled and Delegated.	
Array: Ticket priority	
Array elements (Refer Table 2)	
Immediate, High, Medium and Low.	
Array: Ticket resolution type	
Array element	**Element description**
KEDB	Tickets resolved using knowledge base.
Non KEDB	Tickets resolved without using knowledge base.
Array: Ticket occurrence frequency	
Array element	**Element description**
User defined classification of frequencies of occurrence for tickets into 5 classes – 'Very high', 'High', 'Frequent', 'Low', 'Very low'	
Unknown	Ticket occurrence frequency is unknown

As shown in table 4, four arrays have been defined. Each array is composed of array elements which allow a variable to represent multiple concepts. For example, the average effort required to resolve a ticket may be different for alerts, batch incidents, scheduled tasks and delegated tasks. A variable – 'Average effort required to resolve tickets' therefore may be initialized to contain different values for different ticket types. Array elements may also be combined to represent multiple concepts. For example, service level agreements are defined on the combination of ticket type and priority. Therefore, a variable – 'Response time SLA' would contain 16 (or less, if all combinations are not used) combinations of ticket type and priority. The SLA for an immediate priority alert may be different from a high priority alert or an immediate priority batch incident. This argument is also extended to equations, wherein different equations may be defined for different combinations of array elements within a particular variable.

Ticket type, priority and resolution type arrays are self-explanatory. Ticket occurrence frequency array classifies tickets into different categories based on number of occurrences of a similar ticket in a given period of time. The period of time is user defined. It may be a day, a week or a month. We defined five classifications – 'Very high', 'High', 'Medium', 'Low' and 'Very low' to classify tickets based on number of occurrences. A sixth category, 'Unknown' is also defined to accommodate tickets for which occurrence frequency is unknown (or not calculated).

3.2.1 Control data

This sector comprises of data parameters required to initialize the model for simulation. These data parameters are organized as shown in table 5. The table is divided into 3 columns – Parameter, Description and Arrays. Parameter column outlines the parameters in the model. Parameters are grouped together in the column as concepts and may not necessarily have the same names in the model. For example, the parameter 'Ticket response time' corresponds to four parameters in the model – 'Average ticket response time [Alert]', 'Average ticket response time [Batch]', 'Average ticket response time [Scheduled]' and 'Average ticket response time [Delegated]'. The arrays column outlines the arrays across which the parameter may be further sub-divided in the model. For example, priority distribution corresponds to – 'Priority distribution [Immediate, Alert]', 'Priority distribution [Immediate, Batch]' and other possible combinations of array elements for priority and type arrays.

Table 5.Data parameters in control data sector

Data parameters		
Parameter	**Description**	**Arrays**
Priority distribution	Percentage distribution of each ticket type across all priorities.	Priority, Type
Event distribution	Percentage distribution of events into alerts and batch incidents.	Type (Alert and Batch)
Ticket type distribution	Volume for all ticket types averaged out for all hours within a week. A graph of ticket volumes for all ticket types is plotted against time of week (0 to 167 hours).	Type
Response time	Time required responding to tickets for each ticket type.	Type
Resolution time	Time to resolve a ticket with and without assistance of knowledge base	Type, Resolution
Service level agreements		
Parameter	**Description**	**Arrays**
Response time	SLA for response time defined across all ticket types and priorities.	Priority, Type
Resolution time	SLA for resolution time defined across all ticket types and priorities.	Priority, Type
Resource management		
Parameter	**Description**	**Arrays**
Resource distribution	Weekly schedule for resources in service desk and resolution team.	None
Cost per resource	Cost per hour for resources in service desk and resolution teams.	None
Resolution resources used for ticket response	Percentage of resources in the resolution team used for ticket response.	None
Knowledge management		
Parameter	**Description**	**Arrays**
Effort spent	Percentage of total effort of resolution team spent on creating knowledge articles.	None
Required effort	Effort required (in person-hour) to create and validate knowledge articles.	None
Utility	Extent to which a knowledge article reduces effort required to resolve a ticket.	None
Ticket-to-problem ratio	Number of tickets arising due to a particular problem (or root cause).	None
Release management		
Parameter	**Description**	**Arrays**
Release schedule	Schedule of releases for the period of simulation.	None
Effort required per release	Total effort (in person-hour) required for a release.	None
Target release time	SLA for time required to deploy a release.	None
Event management		
Parameter	**Description**	**Arrays**
Event generation schedule	Average volume of events occurring at a given point of time in a week.	None
Resources required per event	Number of resources required to monitor and detect an event.	None
Events to incidents	Percentage of non-intercepted events missed to create potential incidents.	None
Events to tickets	Percentage of events which are converted to tickets.	None
Cost per incident	Incident management and service disruption cost.	None
Scenario modelling		
Parameter	**Description**	**Arrays**
Change in ticket distribution	Percentage change in ticket volume distribution (increase or decrease).	Type
Change in resource distribution	Change in resource distribution in service desk or resolution team.	None
Process improvement		
Parameter	**Description**	**Arrays**
Frequency distribution	Distribution and classification of tickets into frequency-based classes.	Type, Frequency
Frequency band	The actual frequency of occurrence for a specific classification.	Type, Frequency

Process improvement schedule	Schedule for process improvements over the duration of simulation.	Type, Frequency

3.2.2 Ticket generation and response

Ticket generation is divided into two sections – a) Tickets generated directly from input distribution, b) Tickets generated from events. Scheduled tasks and delegated tasks are generated from input distribution provided by the user. The input distribution is a plot of average ticket volume against hours in a week. It is our contention that ticket volume pattern is repetitive over time. The duration of repetition is a week (168 hours). Configuration items generate tickets and their support schedule is defined over a week such as, '24x7' (24 hours, 7 days), '16x7' (16 hours, 7 days) and '8x5' (8 hours, 5 days). Therefore, we input distribution as average ticket volume at a given hour across all weeks. The underlying assumption here is the number of configuration items remains constant. Else, percentage change in ticket volume is to be provided as input. Equation 4 computes number of tickets generated per unit time. The equation applies only to ticket types – Scheduled and Delegated.

*Tickets generated per unit time [Type] = Max (Ticket distribution [Type] (Modulus (Time in hours, 168)) * (1+Percentage change in tickets generated [Type]) – Reduction in number of tickets [Type]*, 0) (4)

As shown, tickets generated per unit time is a combination of 3 factors:

- Average number of tickets at a given point in time within a week is input as a table function which returns the corresponding ticket volume for the input hour. For a given point in time, the modulus function returns the corresponding hour-of-week, which is provided as input to the table function.
- Percentage change in tickets generated, as input by user at the start of simulation or during interactive simulation. It is used to model scenarios such as addition / discontinuation / change of configuration items.
- Reduction in number of tickets is a result of process improvements. One purpose of process improvement, which is addressed in our model, is to reduce repeated occurrences of similar tickets which reduces the number of tickets generated per unit time (Section 3.3.7).

Alerts and batch incidents are generated from events, as discussed in section – 3.3.3. They are consolidated in a single variable – 'Tickets generated per unit time'. Tickets generated are assigned priority based on priority distribution as.

*Prioritized tickets [Priority, Type] = Tickets generated per unit time [Type]*Priority distribution [Priority, Type]* (5)

Prioritized tickets accumulate as 'Tickets arrived'. Tickets are assigned to associates for response and acknowledgement. The underlying assumption is that an associate can address one ticket at a time. In an ideal scenario, all tickets arrived would be assigned to associates for response. However, it is constrained by number of associates available for response. A backlog is created if the number of tickets arriving are greater than the effort available to acknowledge them. An accumulator 'Tickets response pipeline' forms a buffer for tickets in the response process. This creates a lock-in of certain associates (equal to the number of tickets) for duration of response. Tickets are responded post a delay equal to 'Average ticket response time' (Figure 2).

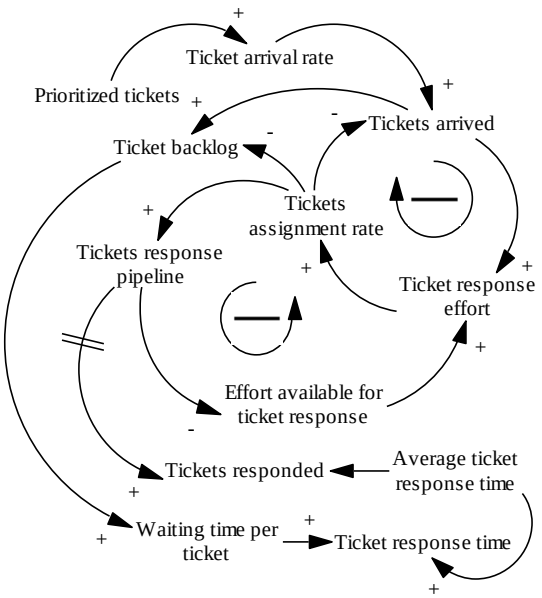

Figure 2.Ticket response causal diagram

Ticket response effort' computes the actual effort expended for response of tickets. Since, 'Ticket response effort' is computed at every time step, it essentially represents the number of tickets responded at a given point in time. At this stage of the model, tickets are classified into type and priority. Herein, the model defines the following rule, as shown in figure 3 for addressing tickets, either for response or resolution (Section 3.3.4).

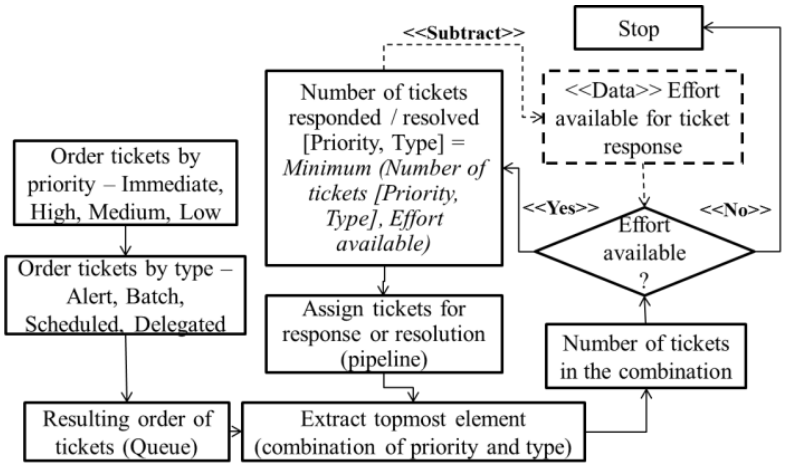

Figure 3.Ticket allocation rule (for response or resolution)

Ticket response effort [Priority, Type] = Minimum (Tickets arrived [Priority, Type], Effort available for ticket response) (6)

'Effort available for ticket response' for a given priority and type is computed in equation 7.

Effort available for ticket response [Priority, Type] = Initial effort available – Sum (Effort spent for ticket response for prior tickets in queue) (7)

Tickets are queued as they arrive and create a backlog if they are not acknowledged due to lack of effort. Average time spent by a ticket waiting in the queue is computed as,

$$\textit{Average waiting time[Priority, Type]} = \int_{time=0}^{time=t} \textit{Maximum (Tickets arrived [Priority, Type]} - \textit{Tickets acknowledged [Priority, Type], 0)} \div \int_{time=0}^{time=t} \textit{Tickets arrived [Priority, Type]} \quad (8)$$

At any given time step, the number of tickets in queue are equal to the positive difference between tickets arrived and acknowledged. Vice-versa, it can also be interpreted as 'n' tickets waiting for a time step, where 'n' is the number of tickets in queue at that time step. The cumulative sum of number of tickets at each time step (numerator) is equivalent to cumulative waiting time for all tickets. Average waiting time, is therefore, a ratio of the cumulative waiting time to cumulative number of tickets generated till that time. The overall response time per ticket is defined as:

Overall response time [Priority, Type] = Average response time [Type] + Waiting time [Priority, Type] (9)

Here, average response time is provided as input.

Compliance to service level agreements is computed based on comparison between overall response time and SLA response time as defined in equation 11. It is to be noted that we use the 'If' function as defined in equation 10.

If (Condition, Value if true, Value if false) (10)

$$\textit{Response SLA compliance [Priority, Type]} = \int_{time=0}^{time=t} \textit{If (Overall response time [Priority, Type]} < \textit{SLA response time [Priority, Type], Number of tickets responded at time instance, 0)} \div \int_{time=0}^{time=t} \textit{Tickets arrived [Priority, Type]} \quad (11)$$

For every time step, the number of tickets responded within SLA is accumulated. Percentage compliance is the ratio of cumulative tickets responded within SLA to total number of tickets generated till that time.

3.2.3 Monitoring and event management

In addition to scheduled and delegated tasks, events are generated by configuration items. These events may be informational (such as task completion notifications), warnings (such as memory usage warning) or exceptions (errors in configuration items). Monitoring is required to detect and respond to events. If not acknowledged and resolved, an event may cause a potential incident. Events are classified into alerts and batch incidents. Events are generated randomly from a weekly distribution provided as input. Similar to tickets, event generation is defined in equation 12. Equation 12 is applicable for ticket types 'Alerts' and 'Batch incidents'.

*Events generated per unit time [Type] = Max (Event distribution (Mod (Time in hours, 168)) * (1 + Percentage change in events generated) * Event classification [Type] – Reduction in number of tickets [Type], 0)* (12)

Here, 'Percentage change in events generated' is part of a scenario which user may configure at start of simulation or during interactive simulation. Event classification is the distribution of events in to alerts and batch incidents. Events (alerts and batch incidents) may also be reduced through process improvements (as described in section – 3.3.7). Events generated are logged and intercepted by resources (employees). Effort available for monitoring is computed as described in section – 'Resource management'. The number of events intercepted is computed as:

Events intercepted [Type] = Minimum (Effort available for monitoring [Type] ÷ Resources required per event, Events logged [Type]) (13)

Here, 'Resources required per event' is the inverse of the number of events that can be intercepted by an employee per unit time. Events not intercepted are queued and would cause potential incidents if not acknowledged within threshold time.

$$\textit{Event queue time [Type]} = \int_{time=0}^{time=t} \textit{Maximum (Events generated [Type] - Events intercepted [Type], 0)} \div \int_{time=0}^{time=t} \textit{Events generated [Type]} \quad (14)$$

If the queue time is greater than threshold time, then the existing queue is transferred to the accumulator variable – 'Potential incidents'. Events registered as 'Potential incidents' are removed from queue.

$$Potential\ incidents\ [Type] = \int_{time=0}^{time=t} If \begin{pmatrix} Event\ queue\ time\ [Type] > Threshold\ time, \\ Events\ in\ queue\ [Type], \\ 0 \end{pmatrix} \quad (15)$$

Potential incidents have a cost attached in terms of incident management, problem management and service disruption costs. These costs are averted through monitoring as shown in causal diagram (figure 4).

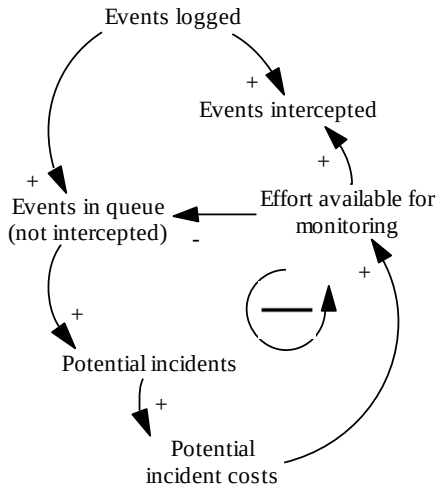

Figure 4.Monitoring and event management causal diagram

The negative feedback loop is a decision loop, wherein 'Effort available for monitoring' is a lever which would reduce overall costs due to potential incidents. A fraction of events are converted to tickets classified into 'Alerts' and 'Batch incidents'. Classification is based on a distribution provided as input by the user. Tickets generated go through the ticket response and resolution process as described in section 3.3.2 and 3.3.4 respectively.

3.2.4 Ticket resolution

Tickets responded are queued for resolution. As stated earlier, the resolution team may be same or different from the response team. In the context of command center, tickets are generally responded and resolved by the same team. Figure 8 shows a causal diagram for the ticket resolution process. Responded tickets are queued as – 'Tickets to be resolved'. Tickets may be either resolved with or without referencing the KEDB. 'Number of tickets resolved using KEDB' is determined by the 'Number of articles present in the KEDB', 'Ticket-to-problem ratio' and 'Ticket occurrence frequency' (per unit time) as.

*Tickets resolved using KEDB [Type] = Number of knowledge articles [Type] * Ticket to problem ratio * Ticket occurrence frequency per unit time * Average utility of KEDB* (16)

The resolution team may not be able to resolve all tickets without using KEDB and therefore a fraction (0 to 1) of tickets not resolvable using KEDB are escalated to a higher level. It is to be noted that 'Average time to resolve a ticket using KEDB', 'Average time to resolve a ticket without using KEDB' (No escalation) and 'Average time to resolve a ticket through escalation' are different in values based on context. Also, 'Average time to resolve a ticket using KEDB' would always be less than resolution without using KEDB. Therefore, an effective KEDB reduces ticket resolution time. KEDB effectiveness is computed as a ratio of the number of tickets resolved using KEDB to total number of tickets.

Effort available for resolution of tickets is computed at every time step. Tickets are allocated for resolution based on priority and type. The rate at which tickets are resolved is moderated by two factors – 'Time to resolve the tickets' and 'Effort available to resolve tickets' (Figure 5). KEDB is considered a factor in determining time to resolve tickets.

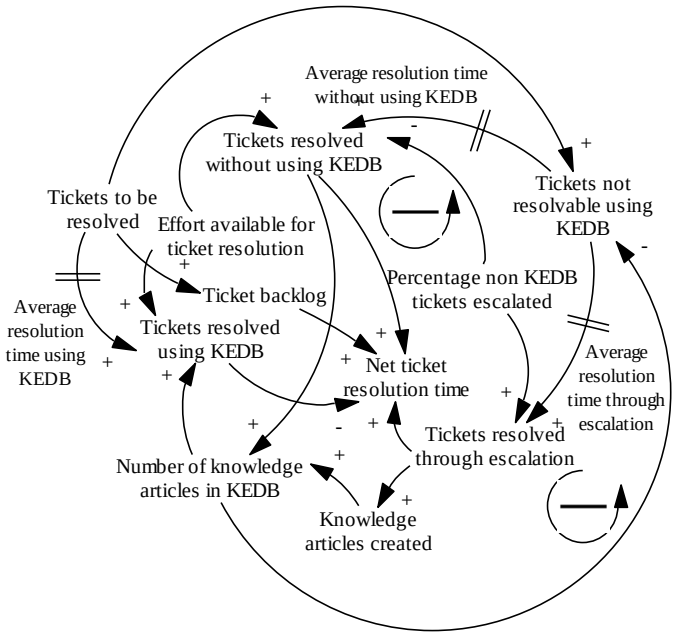

Figure 5.Ticket resolution causal diagram

Tickets not resolved add to backlog and increase average waiting time for tickets to be resolved. Waiting time for tickets is determined similarly.

$$\textit{Average waiting time for ticket resolution [Priority, Type]} = \int_{time=0}^{time=t} \textit{Maximum (Tickets to be resolved [Priority, Type]- Tickets resolved [Priority, Type], 0)} \div \int_{time=0}^{time=t} \textit{Tickets responded [Priority, Type]} \quad (17)$$

Overall resolution time is the sum of waiting time and resolution time, which is further a weighted sum of resolution type fractions (Using KEDB, Non KEDB, and Escalation) multiplied by respective resolution time.

*Ticket resolution time [Priority, Type] = Waiting time [Priority, Type] + Fraction of tickets resolved using KEDB * Average ticket resolution time using KEDB + Fraction of tickets resolved without using KEDB * Average ticket resolution time without using KEDB + Fraction of tickets escalated * Average ticket resolution time through escalation* (18)

Ticket resolution time for ticket types and priorities are compared to respective SLAs to compute SLA compliance.

3.2.5 Release management

Release deployment and checks are a part of command center activities. Releases are generated based on a schedule (as input by the user). Releases are assigned to employees. It is to be noted that more than one employee is dedicated to handle release deployments.

Releases assigned = Minimum (Releases arrived, Effort available for release management ÷ Resources required per release) (19)

Resources are locked-in for the time it takes to deploy and check a release. In event of lack of resources, releases are queued and waiting time is computed. Release deployment time is computed as the sum of time for release deployment and waiting time. SLA compliance is computed based on the comparison between release deployment time and release deployment SLA.

3.2.6 Knowledge management

Knowledge management module captures process of creation and validation of knowledge articles. Tickets resolved without using KEDB are analyzed, documented and validated for future reference (as knowledge articles). In actual terms these articles correspond to known errors, standard operating procedures and process documentation. One or more articles may be referred for resolving a particular ticket. Also, one article may be linked to one or more tickets, depending on ticket-to-problem ratio. Figure 6 shows the causal diagram for knowledge management process.

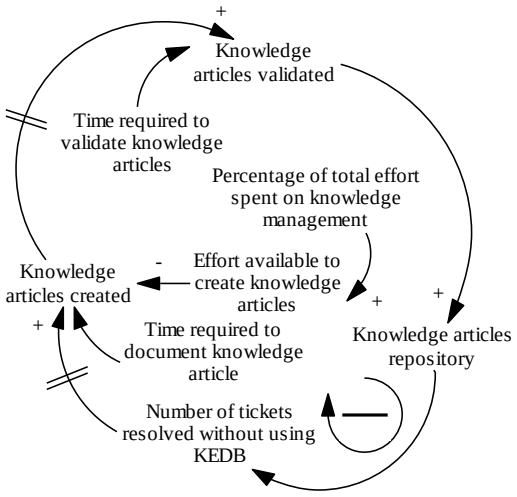

Figure 6. Knowledge management process

Knowledge articles are documented by team members. The effort provided is regulated by the user through a parameter – 'Percentage of total effort spent on knowledge management'.

*Knowledge articles created per unit time = Maximum (Tickets resolved without using KEDB – (knowledge articles created but not validated + knowledge articles in pipeline) * Ticket occurrence frequency per unit time * Ticket to problem ratio, 0) ÷ (Ticket to problem ratio * (1 + Ticket occurrence frequency per unit time))* (20)

For every ticket resolved without using KEDB, number of knowledge article created would be,

*1 ÷ ((Ticket to problem ratio) * (1 + Ticket occurrence frequency per unit time)* (21)

The expression computes the probability of a ticket corresponding to a unique knowledge article. For example, if a knowledge article is applicable for 5 tickets on average, then a set of 10 tickets would on correspond to 2 unique knowledge articles. Also, each ticket has duplicates due to repeated occurrences, which is factored through the parameter – 'Ticket occurrence frequency per unit time'. Since pipeline and created articles do not reflect in the repository, therefore they are subtracted to add only new articles to pipeline. Knowledge articles in the pipeline are documented based on effort available at each time step. Knowledge articles are then validated and added to the repository after a validation delay, which is the time taken by stakeholders to validate a knowledge article.

3.2.7 Process improvement

Process improvements are enhancements which reduce number of tickets or effort required to respond to and resolve a ticket. In this paper, we model the effect of process improvement towards reducing repeated occurrences of tickets. Frequency of occurrence is the number of times a similar ticket recurs per unit time. Reduction in frequency of occurrence translates to elimination of tickets. An example of process improvement is elimination of false alarms for alerts and batch incidents. It reduces number of tickets arriving and effort spent leading to cost savings and enhanced SLA compliance levels. To model process improvements, we assign frequency bands for all ticket types (Column) and frequency sets (Row) as given in table 6.

Table 6. Frequency band template (Example for alert and batch incidents)

	Very high	High	Frequent	Low	Very low
Alert	45, 50, 55	35, 40, 45	25, 30, 35	15, 20, 25	5, 10, 15
Batch incidents	25, 15, 12	12,9, 8	8,7, 6	6,5,3	3,2,1

Each cell contains a triplet – {Minimum, Average, and Maximum} which constitutes a band for a particular ticket type and frequency set. Frequency is expressed as a number of similar ticket occurs per month. An initial distribution is provided for all ticket types and frequency sets (including unknown).

Table 7.Frequency distribution (Example for alert and batch incidents)

	Very high	High	Frequent	Low	Very low	Unknown
Alert	5%	20%	30%	20%	5%	20%
Batch incidents	15%	25%	35%	20%	5%	0%

A parameter – 'Percentage reduction in frequency' provides the extent to which recurrence is reduced when process improvement initiatives are put in place, for all ticket types and frequency sets. Example – The frequency of a batch incident with very high frequency initially can be reduced by 80% or the frequency of a batch incident with low frequency cannot be reduced further. A process improvement schedule (policy decision) is provided as input. The schedule comprises of improvement levels for each frequency set for a specified time interval (monthly, quarterly). Figure 7 shows the causal diagram for process improvement.

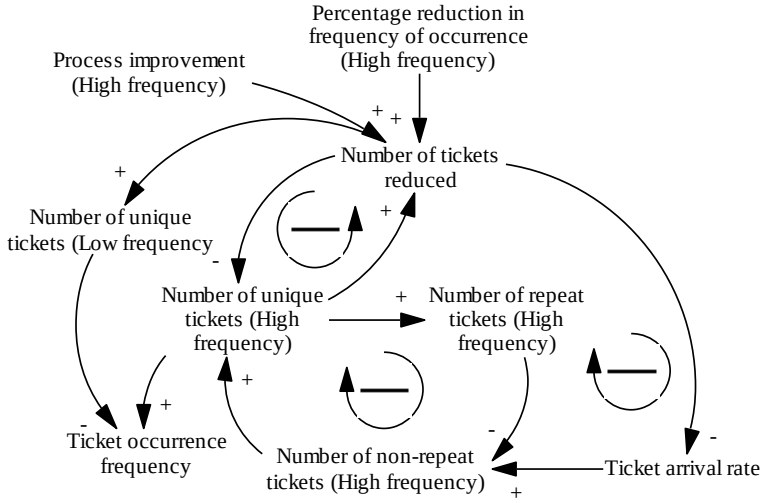

Figure 7. Process improvement causal diagram

The causal diagram above shows process improvement for high frequency set for any ticket type. Similar diagram and set of equations described below are applicable for all frequency sets and ticket types. Ticket arrival rate is computed from user inputs. 'Unique tickets' is a set of tickets across ticket types and frequencies without recurrences. At the start of simulation, the number of known unique tickets is zero. The number of tickets which would occur at each time step due to repetition is computed using known unique tickets as.

*Number of repeat tickets [Type, Frequency] = Number of unique tickets [Type, Frequency] * Frequency of occurrence per unit time [Type, Frequency]* (22)

New tickets are a difference of tickets arrived and tickets repeated. These are added to the number of unique tickets, which is an accumulation variable.

$$\textit{Number of unique tickets} = \int_{time=0}^{time=t} \left(\frac{\textit{((Number of non-repeat tickets [Type, Frequency] ÷}}{\textit{(1 + Frequency of occurrence per unit time [Type, Frequency])}} \right)$$

+ Change in frequency distribution [Type, Frequency]) (23)

Non-repeated tickets are divided by the occurrence frequency to arrive at the corresponding number of unique tickets. We model process improvement as the percentage of unique tickets, which are affected by the improvement leading to a reduction in its frequency of occurrence. At every improvement cycle, the model executes following steps to re-evaluate frequency distribution. Through re-evaluation, the model reduces the proportion of high frequency tickets while increasing the proportion of low frequency tickets. We describe re-evaluation process below.

a. Compute resulting frequencies for each frequency set as:
   Resulting frequency [Type, Frequency] = Frequency average(1 – Percentage reduction in frequency of occurrence [Type, Frequency])* (24)
b. For each frequency set (f), if the resulting frequency does not belong in its range, then compute the number of tickets to be re-assigned.
   *Change in frequency distribution (f) = -1 * Number of unique tickets (f) * process improvement level (f)* (25)
   Search for frequency set (f') to which resulting frequency belongs and re-compute distribution for f' as,
   *Change in frequency distribution (f') = Number of unique ticket (f) * process improvement level (f)* (26)
c. Change in frequency distribution is added to the number of unique tickets as shown in equation 23. It is to be noted that the frequency band for f' will always be less than or equal to f.

Reduction in number of tickets per unit time is the negative sum of change in distribution for each frequency set multiplied by the average frequency for the set across all frequency sets (Equation 27). Reduction in occurrence frequency reduces number of tickets generated per unit time.

*Reduction in number of tickets per unit time [Type] = -1 * ∑ (Change in frequency distribution [Frequency]) * Average frequency per unit time [Frequency])* (27)

Ticket occurrence frequency per unit time is the weighted average frequency of all frequency sets, weighted by the number of unique tickets (Equation 27).

*Ticket occurrence frequency per unit time [Type] = (∑ (Average frequency per unit time [Frequency]) * (Number of unique tickets [Type, Frequency])) ÷ (Total number of unique tickets for each type)* (28)

3.2.8 Resource management

The purpose of this module is to allocate resources to activities described above and keep track of assigned resource values (Figure 8).

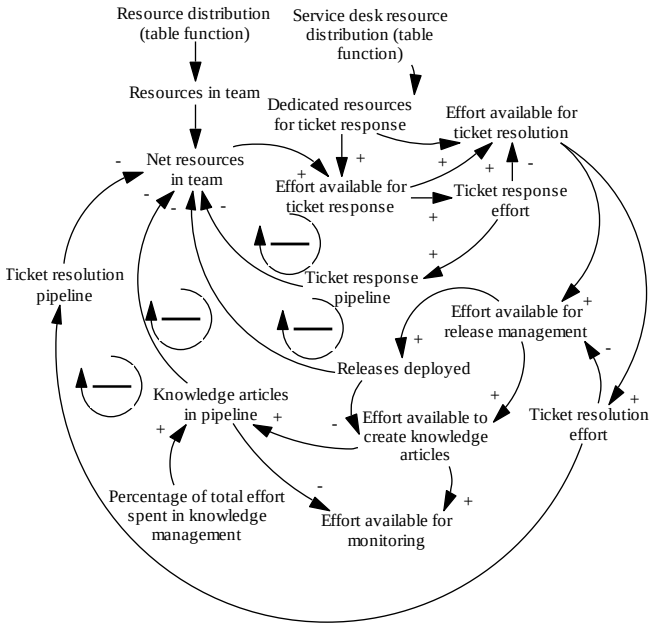

Figure 8. Resource management causal diagram

At a given point in time, net resources in team is computed as.

Net resources = Maximum (Resources in team – Resources being used for ticket response – Resources being used for ticket resolution – Resources being used for releases-Resources being used for knowledge articles, 0) (29)

Net resources are assigned to activities in the order – 'Ticket response', 'Ticket resolution', 'Release management', 'Knowledge management', 'Monitoring and event management'. Resources may be unavailable for one or more activities which would lead to backlogs and potential incidents in case of monitoring. It is to be noted that effort spent in a particular activity is also a function of corresponding volume of work within the activity such as tickets arrived for response; tickets to be resolved; releases arrived; and knowledge articles in pipeline (Not shown in figure 11).

4. MODEL SIMULATION

4.1 Initialization

Initialization of a system dynamics model implies providing input for all data parameters and table functions. We initialized the command centre model using data generated from a command centre engagement within our organization. We extracted values for the following data points.

Table 8. Data points extracted from given data

Data parameters
Ticket type distribution, Event distribution, Priority distribution, Resolution time and Response time.
Service level agreements
Resolution time and Response time.
Resource management
Resource distribution. Costs per resource assumed.
Knowledge management
Initial effectiveness of knowledge base.
Release management
Release schedule and Effort required per release.
Event management

Event generation schedule. Other parameters – Resources required per event and cost per incident are assumed.
Process improvement
Frequency distribution (Batch incidents) and Percentage reduction in frequency (Batch incidents)

4.2 Policy Configuration

Model configuration implies providing values to parameters which represent scenario and policy related parameters (not derivable from data). We simulated two runs with conditions as shown below.

Table 9.Run parameters

	Base run	**Scenario & Policy run**
Scenario parameters		
Initial number of knowledge articles	25, 50, 35, 45	5, 10, 7, 9
Threshold time to intercept events	15 Minutes	10 Minutes
Increase in events over time	20%	40%
Policy parameters		
Additional resources in team	0	1
Process improvement	0	Batch incidents – 100%,100%,100%, 75%, 0
Process improvement interval	NA	15 days
Percentage resources in KM	25%	100%

Base conditions correspond to parameter values derived from data. We perform what-if analysis by defining scenarios (changes in environment) which would affect the model. We considered 3 scenario parameters.

a. Initial number of knowledge articles represents the effectiveness of KEDB at start of engagement. We reduce it by 75% for scenario and policy run.
b. Threshold time to intercept events represents the lag between event generation and event manifesting as a potential incident. If intercepted within lag, an event would be logged and resolved. Else, it would incur incident management and service disruption costs.
c. Increase in events over time – We observed from data, that events increased by 20% over 3 months' time. This is provided as base value. We set scenario value at 40%.

Impact due to scenarios is mitigated by policies. We configure following policy parameters in the model.

a. Addition of resources in team.
b. Percentage spent on creating knowledge articles.
c. Process improvements to reduce ticket volume.

Table 9 outlines the configuration for scenarios and policies. For parameter – 'Initial number of knowledge articles' the values are in ticket type sequence – 'Alert', 'Batch', 'Scheduled' and 'Delegated'. For parameter – 'Process improvement' it implies that 100% of 'Batch incidents' are analyzed for possible reduction in frequency of occurrence. Section 4.3 shows simulation results and discussion.

4.3 Simulation results

We simulated 3 runs for the model for 3 months based on given configuration (Table 9). The runs are as follows

a. Base – Base values (derived from data).
b. Scenario (Only scenario impact) – Values for scenario parameters are changed using configuration table. Values for policy and other parameters remain unchanged.
c. Policy – Values for policy parameters are changed using configuration table to mitigate scenario impact.

We display the simulation results through time series plots for metrics critical to the engagement. It is to be noted that simulation time-step is 1 minute (shown in x-axis of graphs). Please refer to legend below for graphs.

Scenario run Base run Policy run

Figure 9 shows averaged time to resolve tickets (in hours). Time to resolve a ticket is impacted negatively (increases) in scenario conditions. Policy implementation reduces impact and time decreases to base conditions by end of simulation.

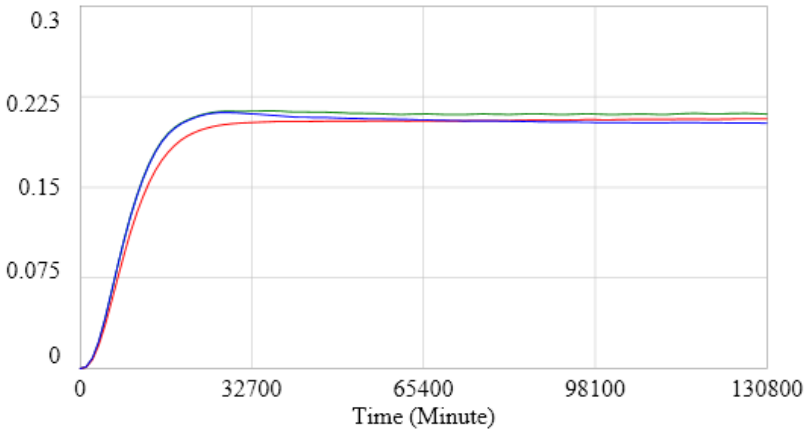

Figure 9. Ticket resolution time series

One factor attributing to reduction in resolution time is investment in knowledge management. Number of knowledge articles in repository increases the percentage of tickets being resolved using KEDB which reduces overall resolution time (Figure 13). In the base run, the initial effectiveness of KEDB is set at 98% from data. Percentage of tickets resolved using KEDB culminates at 90% in scenario run and 95% in policy run, at higher rate.

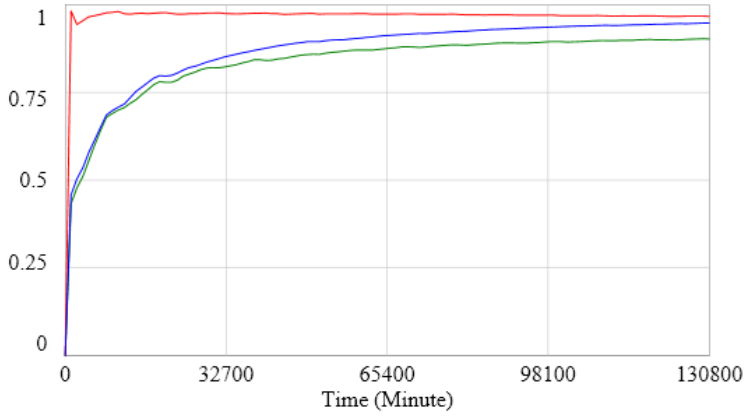

Figure 10. Fraction of tickets resolved using knowledge base.

The other factor contributing to resolution time is waiting time per ticket, which is lower in policy run due to additional resources (Figure 11) and reduction in effort through process improvements.

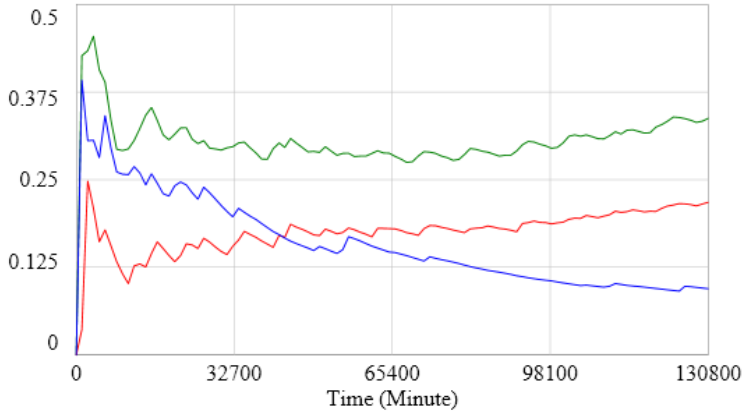

Figure 11. Waiting time per ticket (minute)

Resolution time affects service level compliance. Service level compliance increases as resolution time decreases. We observed > 95% compliance by the end of all runs (Figure 12). Highest compliance was observed at base values followed by policy runs. However, in policy and scenario runs, compliance is < 90% for first 15 days, which is unfavorable as it would result in penalty due to SLA non-compliance.

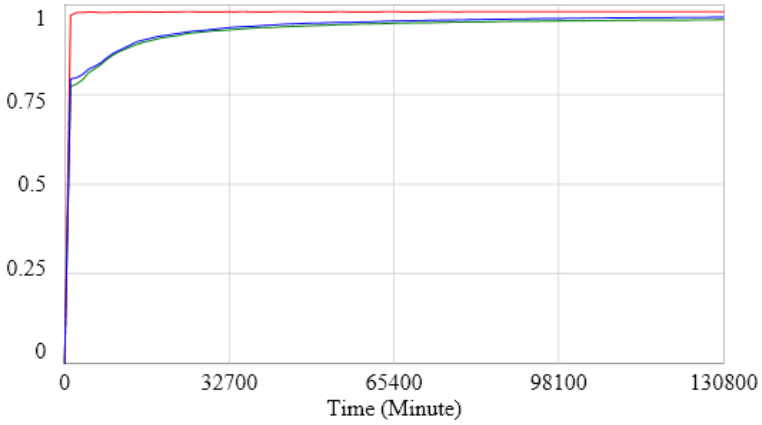

Figure 12. Service level compliance (fraction)

One of the benefits of process improvements is reduction in number of events and therefore potential incidents. Figure 17 shows the number of potential incidents for each run. Due to same number of resources as base run and low effort spent in knowledge management, scenario run requires longer to resolve a ticket. This leads to a resource deficit for monitoring (Figure 13). Percentage resource deficit is the ratio of cumulative deficit to the cumulative effort required for monitoring over simulation period. Scenario run records the highest number of potential incidents.

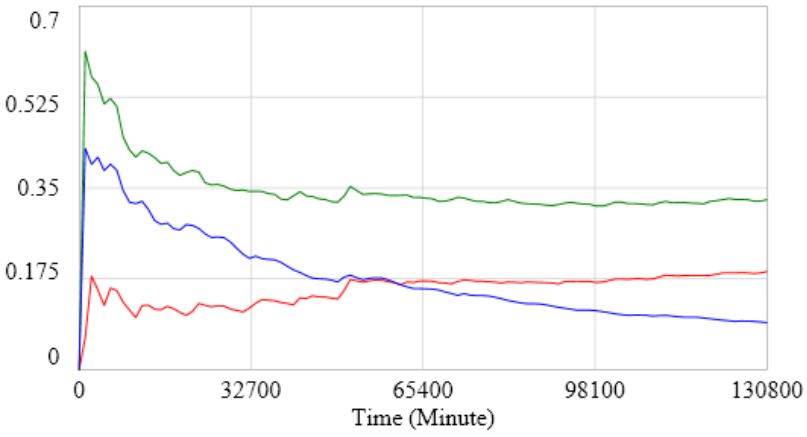

Figure 13. Average resource deficit (fraction) over time

As shown in figure 14, the number of potential incidents continuously increases in scenario run. In policy run, potential incidents increase early on but plateaus over time.

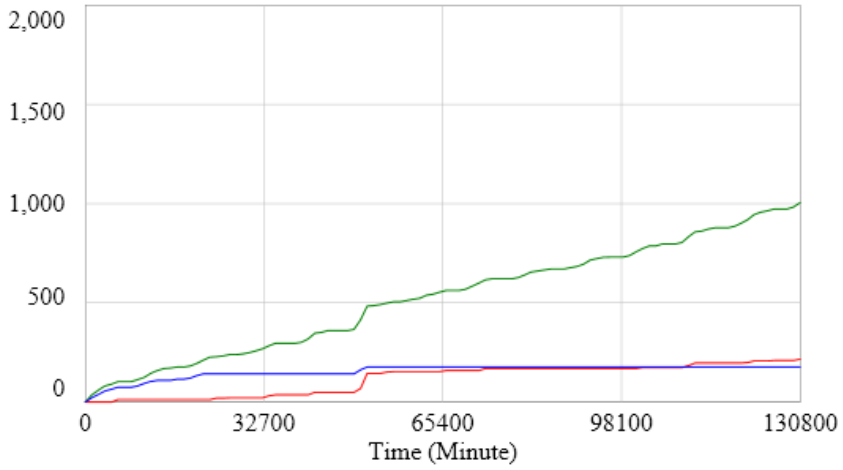

Figure 14. Number of potential incidents

Potential incidents would lead to incident management and service disruption costs. Quantification of potential incidents into costs is not a part of this paper. Costs per incident may be estimated using organizational data and context (Rumber, 2013 and George, 2008). We define a parameter – 'Costs of not monitoring' to represent risk exposure if monitoring efforts are not put in place. 'Costs of not monitoring' is defined as the product potential incidents and cost per incident.

Process improvement leads to reduction in effort over time. Figure 15 shows optimal resource costs for all runs. Optimal resource costs are calculated assuming 100% resource utilization and cost per resource is assumed as 6 units / hour. Reduction in effort leads to lower costs. Figure 16 shows effort (in person-hour) spent on resolving tickets over time. Effort reduces post process improvement in the policy run. Owing to an additional resource, costs initially are higher in the policy run. However, policy run cost rate decreases over time and overall costs (optimal) in the policy run are lower than base run. Total costs are higher in policy run than base run and scenario run due to additional resource.

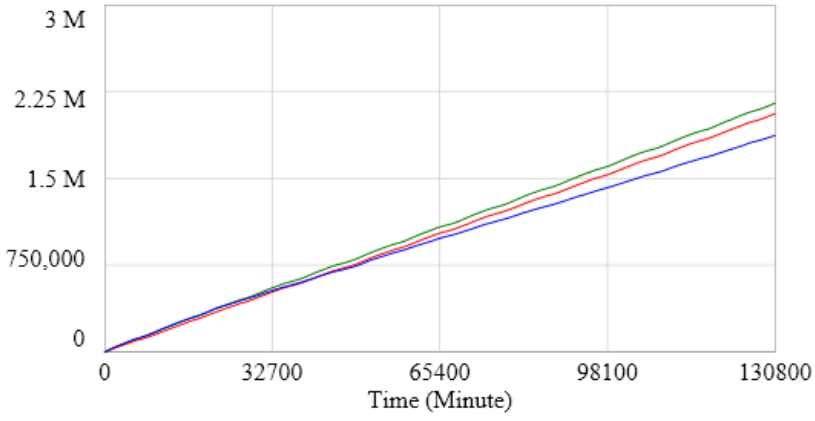

Figure 15. Optimal resource costs

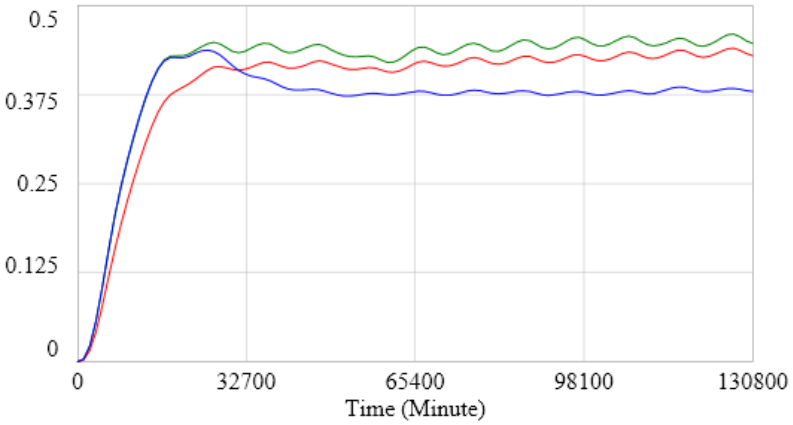

Figure 16. Effort spent (in person-hour) in resolving tickets over time

Reduction in effort leads to lower resource utilization in policy run as compared to base and scenario run, as shown in figure 17.

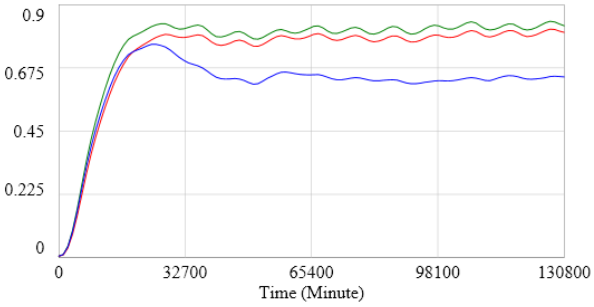

Figure 17. Resource utilization (fraction) over time.

Process improvement costs are assumed as 100 units / ticket. Total cost for process improvement is shown in figure 18.

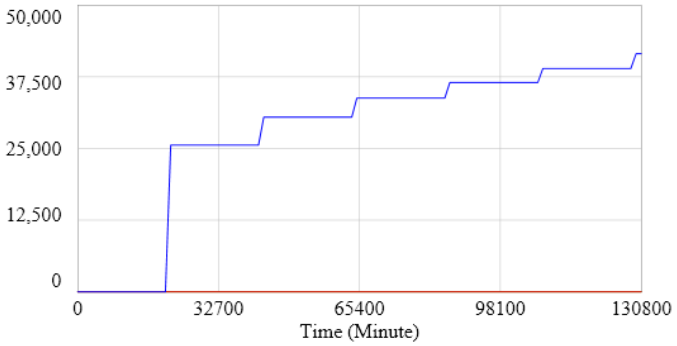

Figure 18. Process improvement costs

5. CONCLUSION

This paper described and demonstrated the modelling and simulation of an IT operations command centre. It used system dynamics approach to model the behavioral dynamics of a command centre. Complexity of command center asks for modelling and understanding it in terms of feedback loops. System dynamics modeling has evolved as a result of new modelling tools. Advancements in terms of array definition, calibration and model analysis features in system dynamics modelling tools facilitate detailed modelling of systems. As detailed in section 3 (model description), we exploited these features to define multiple arrays to capture variety without proliferation of parameters. We modelled sector-wise discrete processes such as ticket response, ticket resolution, monitoring and event management aggregated with other elements and processes through causal links and feedback loops. Based on this study we infer that investment in monitoring, process

improvement and knowledge management reduces volume of tickets, effort required per ticket and saves costs in the long run. These policies are required to mitigate negative impact due to scenarios such as change in ticket volume, ticket distribution, resource distribution etc. and meet SLAs. Combination of scenario analysis and policy design is instrumental in governance of a command center engagement. This study showcases a management flight simulator for command center, which allows a manager to interactively input scenarios and policies during simulation. This facilitates a manager to formulate explicit and tacit policies and generate roadmaps to achieve business objectives.

6. References

Assunçao, M. D., Cavalcante, V. F., de C, G., Maira, A., Netto, M. A., Pinhanez, C. S., & de Souza, C. R. (2012). *Scheduling with pre-emption for incident management: when interrupting tasks is not such a bad idea*. Scheduling with preemption for incident management: when interrupting tasks is not such a bad idea. Winter Simulation Conference 2012: 403

Bartolini, C., Stefanelli, C., & Tortonesi, M. (2008). *SYMIAN: A simulation tool for the optimization of the IT incident management process.* In Managing Large-Scale Service Deployment (pp. 83-94). Springer Berlin Heidelberg.

Beer, S. (1985). *Diagnosing the System for Organizations*, Wiley, Chichester.

Forrester, J. W. (1969). *Urban dynamics.* M.I.T. Press

George R. (2008). *Estimating the cost of incidents – The incident management process. Serio IT Service desk and helpdesk blog*. http://www.seriosoft.com/Blog/?p=240.

Goo J., Kishore R., Rao H.R., & Nam K. (2009). The role of service level agreements in relational management of information technology outsourcing: An empirical study. MIS Quarterly, 33(1), 119-145.

Information Technology Infrastructure Library (2013). *http://www.itil-officialsite.com/AboutITIL/WhatisITIL.aspx*. Last accessed November 2013.

Lee, J. H., Han, Y. S., & Kim, C. H. (2007). *IT Service Management Case Based Simulation Analysis & Design: Systems Dynamics Approach.* In Convergence Information Technology, 2007. International Conference on (pp. 1559-1566). IEEE.

R. Espejo and R. Harnden, '*The Viable Systems Model – Interpretations and Applications of Stafford Beer's VSM'*, Wiley, Chichester (1989), NY, 1976.

Rai V. K. (2012). *System of Requisite Knowledge for Service Engagement Transformation*. In proceedings of 9th International Conference of Service Systems and Service Management (ICSSSM 2012).

Rai V. K. (2013). *Systems Oriented Approach to Production Support- A Viable Framework*. In Proceedings of 2013 IEEE International Conference on Systems, Man and Cybernetics.

Rai V. K. and Mehta Sanjit (2011). *Systems Approach to As-Is State Formation of an Engagement: A case Study Illustration.* In proceedings of 6th IEEE International Systems Conference (IEEE Syscon 2012).

Rai V. K. and Swaminathan N. (2010). *Constructing Program Management Framework- A system of Systems Approach*. In proceedings of 4th IEEE International Systems Conference (IEEE Syscon 2010).

Rai V.K., Vijayasaradhi B., Subramanian K and Umamaheswari S. (2010). *ODC Portfolio Management Ensuring Sustainability in the Face of Growth Surge- A Case Study*. In proceedings of 4th Annual IEEE International Systems Conference (IEEE Syscon 2010).

Stella modelling and simulation software (2013). *http://www.iseesystems.com/softwares/Education/StellaSoftware.aspx*.

Rumber J. The true cost of desktop support: Understanding the critical cost drivers of desktop support. MetricNet, LLC. http://www.metricnet.com/. Last accessed November 2013.

Sterman, J.D., (2001). System dynamics modelling: Tool for learning in a complex world. California management review 43(4), 8–25.

Vensim (2013). *http://www.vensim.com/*.