Inter-Dimensional Traceability and Intelligence Mining for Complex System Improvement

Farha Mukri, Jay Ramanathan, Rajiv Ramnath & Kelly Yackovich
Center for Experimental Research in computing Systems
The Ohio State University
Columbus, Ohio
OSU-CISRC-4/09-TR14

1. Abstract

Complex service-oriented Business-IT systems are difficult to improve locally and the related challenges contribute to the high rate of 'unsuccessful' IT projects. This paper discusses the motivation for establishing *inter-dimensional traceability* and identifies the enterprise elements and associations that are necessary to analyze and improve. These issues are addressed here by proposing an 'Adaptive Complex Enterprise' framework for performance improvement of service-oriented enterprises based on enterprise-level traceability. This framework allows us to use available modeling, monitoring, simulation and mining tools for the different aspects of complex enterprise systems in an integrated fashion based on the enterprise ontology. The resulting traceability across the different dimensions in a complex enterprise is leveraged through a cycle of continuous improvement addressing the numerous service Requests with services provided, in turn, by a large number of associated enterprise elements. This also leads to an integrated context for decision making. The City Strategic Planning-and-Execution example is used to illustrate the framework and how it supports continuous improvement using traceability.

2. Introduction to the Business Problem

Traceability Concepts and its Different Uses

Traditional definitions of traceability and the focus of research have primarily been on the management of software life-cycle artifacts and the use of various underlying associations between artifacts and their attributes[1][2][3]. One limitation of all this, from a Business-IT complex system perspective, is that traditional traceability research has

focused mainly on the creational phase of software development. Recently, goal-centric traceability has received attention because it helps in dealing with non-functional Business-IT requirements[3]. Recognizing the importance of Business-IT alignment, this, goal-driven requirements engineering has been used to abstract the important relationships between the multiple business goals and their impact on associations and actions as in i* [5]. Requirements traceability itself, however, has other issues which has also prompted research identifying tool gaps and lack of emphasis on pre-Requirement specification traceability [6].

Other emerging work related to traceability of *complex systems* is in the context of ISO20000 (ITIL [i] [7]) where the traceability of installed configuration items within the CMDB is critical to the IT services such as incident resolution, problem and change management, service and operations management. A complementary body of knowledge in TOGAF stresses the importance of traceability between the conceptual business, logical system, and physical architecture layers to facilitate their continuous improvement [8]. That is, the emphasis is on *inter-dimensional* traceability as discussed in [9].

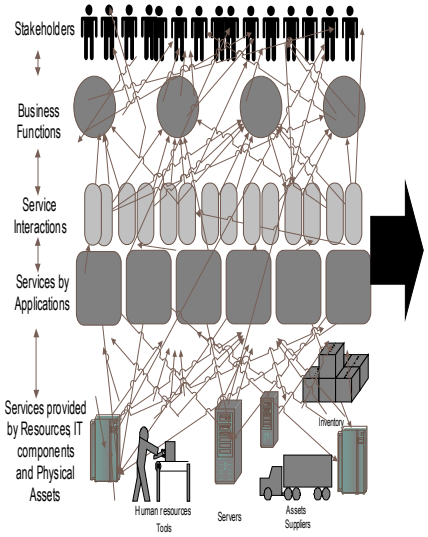


Figure 1: Challenges of the many-to-many interactions across the service layers of the externally-driven enterprise.

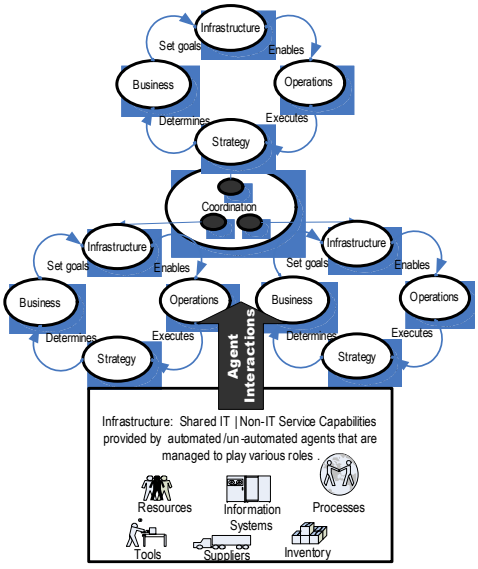


Figure 2: Adaptive Complex Enterprise BioS conceptualization and the underlying infrastructure of Interacting Agents.

Here, realizing that in a complex system improvement is achieved not by simply creating a new service but by an incremental repetitive process of 'tweaking' different layers of services to achieve the desired improvement, we focus on traceability between *Planning-and-Execution*. (More background on complex IT systems management is in [9][**Error! Reference source not found.**].) Thus research focus here extends the goal-directed requirements engineering work further by abstracting the elements and associations in the *Business, Infrastructure, Operations, and Strategy (BioS)* dimensions of complex systems. In particular, we focus on those services that are based on *shared Agents* within the infrastructure and contribute to Interactions and overall Goals. In its essence the questions we ask here are:

- "How can the BioS performance of a complex system be concurrently improved?"
- "What existing methods and tools can be integrated to support the required decision-making for continuous improvement?"

Conceptualization of a Complex System

To answer these research questions we begin with a *to-be* conceptualization of a complex service-oriented enterprise as a collection of *goal-directed organizations* and *sub-organizations* (Figure 2) responding to external Requests. We also contrast this with typical *as-is* approaches and existing view of services (Figure 1). In the as-is view there are many service layers each contributing to other services. Note that important *value-chain associations* are not easy to identify since they are not visible and 'declarative'.

To address this, we propose the *Adaptive Complex Enterprise* organization view on the right. Here each organization (and sub-organization) that handles a collection of Requests is represented in turn as an internal value-chain and cycle of adaptation characterized as follows:

BioS - ***B****usiness value achieved through an* ***I****nformation infrastructure enabled* ***O****perations to deliver on* ***S****ervice Strategy*.

The overview of the associations and traceability due to this conceptualization is illustrated by the Figure 2. Further details of the value-chain associations are given in Figure 3 at the top with the ontology elements of the value chain given at the bottom.

The underlying definitions of the *BioS conceptualization* of an *Adaptive Complex Enterprise* system are as follows:

- *Non-routine Requests:* We use the term 'Request' to generally refer to requirements from the customer in a wide range of enterprise business forms - custom orders, routine orders, service calls, proposal Requests, new application requirements, incidents, emergencies, technology changes, defect reports, and even new product/process requirements. Requests from the customer require a response. At a high level, Requests can also be generally classified as routine or non-routine, and each can in turn be of many types. It is important to note that non-routine Requests have associated requirements that need more knowledge-based responses and routine Requests are typically handled well within existing enterprise systems.
- *Shared Agents:* The responses to non-routine Requests must also enlist the needed services of underlying business Agents (i.e. internal /external organizations and workers, primary[ii] and secondary processes, IT systems and assets). Typically, each Agent (or a group of Agents) has a certain collection of skills or capabilities to play different Roles at specific cost. Knowledge Agents are higher cost and thus often shared. In these cases, an understanding the service requirements of each Request before assigning tasks becomes more important for effective processing. Finally, given the uncertainty in the arrival of Requests, the availability of the Agents also becomes uncertain. In these circumstances 'triage' or dynamic assignment based on *business rules* becomes important. This is one way in which the responses can *adapt* to incoming Requests.
- *Coordination and Triage:* Request Triage examines the Requests coming into the organization and classifies the types and requirements and assigns available resources so that their processing becomes more predictable. For example, satisfying a customer service Request may require the use of multiple physical and electronic resources to produce the deliverable. Although it is simple enough

to capture a generic Request though the use of an IT system, the actual response by the organization and all of the required resources may not be known a-priori. Note that the Request, the executed response, and the deliverable have both electronic and physical manifestations that have to be kept consistent and traceable. This element of uncertainty adds a layer of complexity to the overall system.

- *Interactions and Value Metrics:* Transactions and resulting value creation - as in order fulfillment - are the basis for any business [12] and occur *in-the-large*. Transactions also occur *in-the-small* as for example when the order fulfillment in turn causes hundreds of data base transactions. Thus transactions represent how value is added either at the level of an IT service, or at the level of the business process, or across the supply chain, or across market segments.

  Here, we use the word 'Interaction' to avoid confusion with many similar words like 'task', 'transaction' or 'activity' as in database transaction, business transaction, business activity and so on. Using a single term like "Interaction" also allows us to think of the organization as a fractal [13]. Finally, an Interaction can represent and abstract both *primary* business and revenue generating processes like patient handling, engineering change, etc. and enabling *secondary* processes like IT service support, returns, etc.

  Some of Interactions in the organization are *in-the-small* and others are *in-the-large* and business oriented. Interactions therefore form a base or canonical concept that relates the enterprise activities at all scales. The term *dimension* is used group all the stakeholders with the similar time and scale perspectives on an Interaction. Specifically, strategy, business and operational performance is typically viewed in the aggregate or in-the-large. The metrics for all Interaction instances is aggregated at the type level – either with a Request type in the strategy and business dimensions or with its Interaction type in the operational dimension.
- *Planning-to-execution Traceability across BioS dimensions:* The end service is delivered as a result of an interaction of Agents that produce the deliverable. At

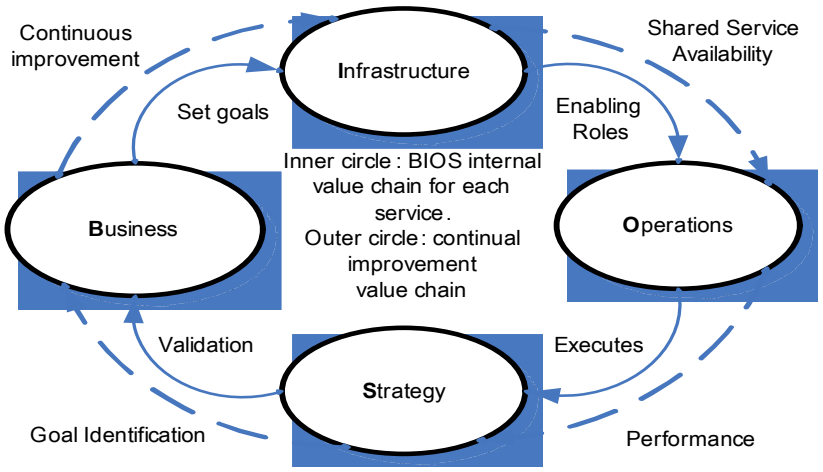
this execution dimension, the metrics is actually captured at an instance level. Service response metrics to a Request employing other services by shared resources are also recorded. All results of execution including variations and exceptions are reported for future improvement. Also at execution (typically enabled by IT) the actual value contributions are made in-the-small. These are then aggregated in-the-large dimensions. Metrics are aggregated and synthesized to reflect performance and goal achievement trends that provide input into decision-making for the different BioS stakeholders.

Business Problem Related to Traceability in a Complex System

Most enterprises have several service organizations that are complex. A familiar example of a complex system includes "Hospital Emergency" where every incoming patient is a non-routine Request for service. Here the *triage* nurse makes an initial *assessment of requirements* and makes an *initial assignment of resources*. Resources change as diagnostics reveal new requirements for additional medical experts.

Another familiar example is a complex organization like the City that is pressured to be more-and-more service-oriented with fewer resources [14]. The City example used here has an expanding service area, fluctuating revenue, and a growing population that combine to place stress on existing response systems. The Mayor has instituted a “covenant” that includes a guiding principle that technology will be a key tool to achieve City objectives and improving service responses. To this end, the City consolidated its IT operations under Department of Technology (DoT) and began implementing a series of IT improvements. Looking to the future, the DoT wishes to develop a plan more strategically aligned to the City departments - that is DoT's customer.

Conceptual View of the BioS Value Chain (inner associations) and continuous improvement of the BioS Value Chain (outer associations)



Related Model Elements

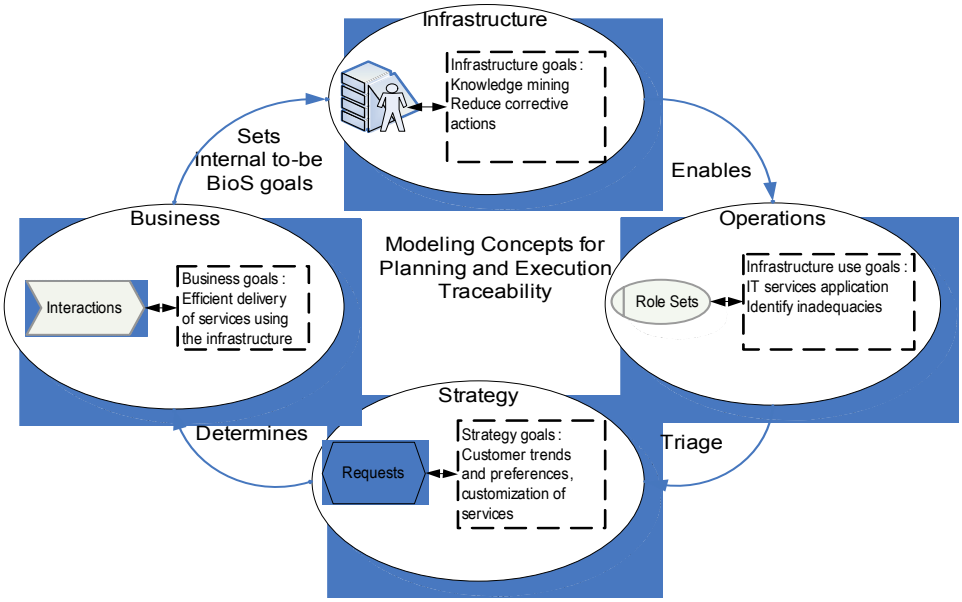


Figure 3: Business, Infrastructure use, Operational, and Strategy (BioS) goals, elements, with traceability associations.

These two examples are systemized in the Adaptive Complex Enterprise ontology presented next. Before proceeding we use the illustrative examples next to generalize the reasons why efficiency in a complex system is challenging:

- Complex systems cannot be improved through one action alone, many inter-related actions are required. Further the emergent behavior of such systems is not entirely predictable.
- Continuous Improvement of Complex Systems requires detailed monitoring of as-is performance to achieve to-be improvements. Thus they require on-going traceability between *Request planning and Request execution*.
- The increased difficulty of capturing traceability in processing one-of-a-kind non-routine Requests which also require multiple agents that are not known a-priori (i.e. processes that involve humans in the decision-making loop).
- Difficulty of gathering knowledge insights on how the Agents collectively deliver a Request / service.
- Missing monitoring and audit tools that relate operational performance and performance of shared Agents thus making decisions related to efficiency more difficult.

These issues are addressed by the research goals here which identify missing methods and tools for planning-and-execution traceability and provide a solution approach to better management of complex systems.

Research Contributions to Model-Driven Engineering of Complex Systems

In reality, we have isolated tools that monitor different aspects of elements within a complex system. However their application to Complex System improvement is not integrated and well understood. We do not have tools that allow us to understand the overall behavior of a complex system and manage by objectives. To address this:

- We present a conceptual modeling framework to identify and associate the elements of a complex system
- We use this to identify gaps in tools and methods, and at the same time we use this as a basis for integrating tools into an integrated monitoring, simulation, intelligence mining and decision-making tool
- We will illustrate the methods using the City Strategic Planning and Execution Effort

Thus the overall research objectives met are:

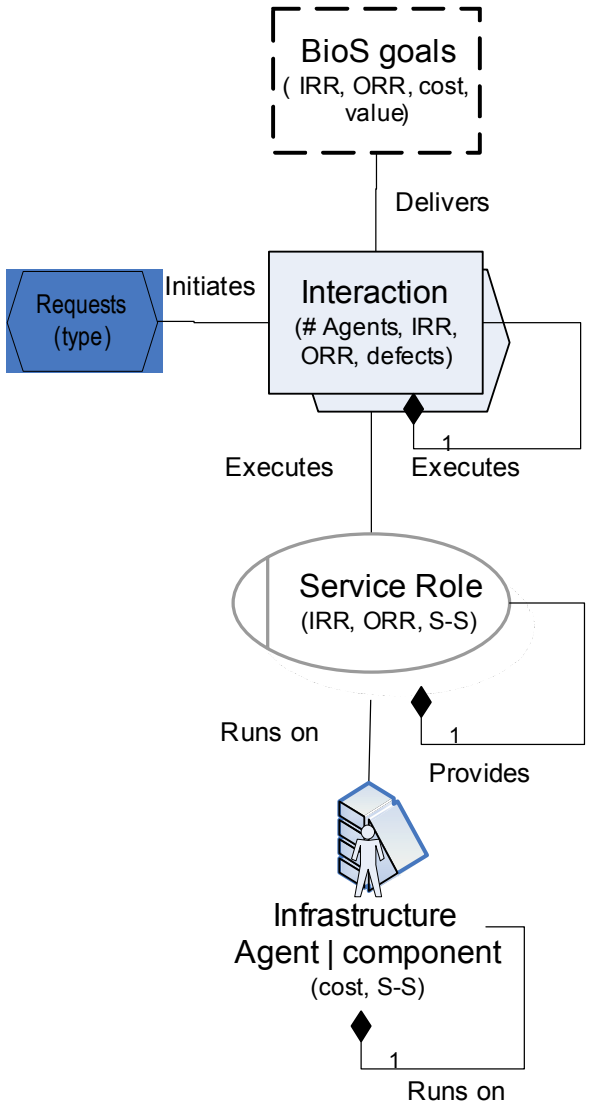
- An Enterprise Ontology for conceptualizing Complex systems
- A Continuous Improvement (CI) method for Complex systems based on an integrated method and toolset for
    - Infrastructure Agent Monitoring using CI Agents that we call observers,
    - Simulation,
    - Intelligence Mining, and
    - Service Interaction(s) performance enhancement

3. Complex System Ontology

Figure 4 illustrates the five types of high-level elements that we assume in any ACE – BioS goals, Requests (and associated deliverable), Interactions, Service Roles (or Role Sets), Agents (artifacts, components, etc.) [9][15][16]. Progressing from the top to bottom these elements represent the enterprise conceptually as follows:

1. *BioS dimensions* are the Business, Infrastructure, Operations, and Strategy goals and stakeholders that are grouped based on performance interests. There can be many goals and stakeholders in each BioS dimension.
2. *Request (and deliverable) represents* the related event from a 'customer' that initiates an Interaction. It also has associated requirements and deliverable details.
3. *Interactions* engage agents to execute and 'deliver' on BioS goals. Interactions are also abstractions that represent the performance milestones of end-to-end business processes (and sub-processes) that are 'executed' to fulfill the Request. An Interaction allows us to measure the customer-facing *Service Level*.
4. *Service Roles (also Role Set)* are the prototypical Agent(s) requirement(s) that are all needed for the Interaction. These Role Sets are filled by actual Agents at triage. The actual Agents are 'part of' or 'run on' the infrastructure. The Role Set and its space and processing speed needs are thus provisioned by the infrastructure; or else the Service Roles 'provide' services to other Roles. The requirements also identify the *Operating Levels* needed of the Agents.

5. *Infrastructure of Agents:* The shared infrastructure consists of the physical machines, routers, and essential operating software. It also includes software components that provide information services. We will also place human Agents in this category.



BioS Goals:

IRR or Input Request Rate: Number of Input Requests of a type queued per time period.

ORR or Output Response Rate: Number of Requests serviced per time period.

Cost of achieving the ORR.

Quality of deliverable as perceived by the customer.

Interactions to achieve the ORR and SL.

execution #'s that is performance attributes associated with Agents - time used, amount used, wait time, and *defects.* These are monitored during the Interaction.

Requests (Initiates Interactions and has an associated deliverable at the end of the interaction).

Instance specific attributes like requirements, customer details, arrival time, assigned to etc. Also includes classification like *type of requirement, type of customer, processing needs etc.*

Service Role set S-S and OLs.

S-S collection of space and speed of processing attributes: This collection includes derived attributes such as current available capacity, cost, monitored actual, projected future etc. We shall often also refer to these as the S-S non-functional attributes. Finally, these are also the attributes that define the Operating Levels. In addition each Role has its own IRR, ORR.

Infrastructure Agent | component

S-S - certain processing speed and space (or capacity) attributes that determine the Request throughput of the service running on or using it.
For a human Agent the processing speed depends on knowledge, skills, and training. In addition each Agent has its own IRR, ORR.

Figure 4: Adaptive Complex Enterprise ontology - elements, associations and example attribute collections.

The associations in Figure 4 further detail the Planning-and-Execution associations of the enterprise elements and value chain associations that underlie the different BioS goals as in Figure 3. Figure 4 is a more detailed view that highlights the associations and attributes that we want to monitor for traceability along these value chain associations.

The attributes associated with each element above are identified in Figure 4. Starting from the lowest row, for a service component we have S-S or 'speed and space' collection of attributes. These attributes can describe any agent. For example, for a human resource this includes attributes like speed and capacity (which in turn may depend on set up time, training and so on). This collection of attributes also refers to characteristics of other types of systems. Generally speaking, the S-S attributes are also of two types - intrinsic and extrinsic S-S attributes. The intrinsic S-S attributes are determined by the characteristics of the Agent. For a software component example, they are those determined by running a database service component in a test infrastructure machine environment, on benchmarks and in isolation. Extrinsic attributes are those S-S attributes that are affected by business Requests from other elements along associations that we will explore later. We will henceforth use the function C, as in C_x(S-S), to reflect the intrinsic and extrinsic S-S values of Agents.

What do S-S attributes convey differently for infrastructure components and service components? To answer this let us first think of service component X that runs on an infrastructure component or machine. In its essence the hardware has a certain available space and is able to process at a certain speed. However, the machine can accommodate a component such as a data base only if the machine's available S-S is greater than the data base own needs or $C_{database}$(S-S). The needed data base S-S also increases with the number of Requests serviced from different process transactions. To accommodate this increase the hardware must in turn have the extra needed S-S available. That is, more generally, the software has a certain fluctuating C_x(S-S) footprint due to extrinsic circumstances that it needs from the hardware. The process transaction element and its attributes introduced next allow us to compute the needed extrinsic S-S values due to external Requests.

Interaction attributes include #Users, IRR, and ORR. Here IRRs are incoming business Requests distinguished from the Request output rate or ORR by the fact that the

former is initiated and queued by the customer while the latter is the actual number processed (and could include many internally generated Requests serviced by the lower level entities).

Finally, Business goal attributes include IRR and ORR costs in addition to aggregated metrics. Note, in addition to the essential attributes identified above there are additional attributes that reflect target and actual values for costs and resource S-S used for processing a Request, and so on.

We can now also introduce specific *Service Levels* (SL) and *Operating Levels* (SL). The SL of an Interaction is reflected by its Request throughput - ORR and quality attributes. This is in turn achieved by provisioned Agents and their own S-S attributes at certain ORR Operating Levels (OLs) to achieve business ORR targets.

Finally, we present our formal *definition of traceability* as the cause-and-effect along the elements, attributes and associations of the Complex System representation based on the above ontology.

4. Existing Methods and Tools Gaps

Many different types of tools are available to monitor attribute values related to the service ontology defined in Figure 4. These are summarized below for background.

Summary of Existing Models, Tools and Gaps

Element type	Functions available and vertical traceability gaps	Examples of applicable methods, commercial and open source tools
Requests; Business, Information use, Operations, and Strategy goals *IRR, ORR, Cost, Quality*	Business applications maintain Request information. With these Requests, there is information on value provided to the customers and business. The business value is mainly indicated by the revenue. Gap: Many Interactions may be needed to deliver on a single Request. Within current enterprise systems it is however difficult to associate all the Interactions that deliver on the primary Request and its associated business value. The dynamics of how multiple organizations can impact each other when Agents are shared is also difficult to obtain. For example, how do different ORR rates of interactions and sub-Interactions impact the ORR of the original Request?	Enterprise data warehousing, CRM, CSC (Customer Request Logs), and so on. Systems dynamics tools like Vensim[iii] could be used to create simulations of Request cycle times for Request types and define throughput of the Requests (ORRs for IRRs).

Interactions *Request type, IRR, ORR*	Agents used to achieve Interaction IRR-ORRs for each type of Request can be estimated by analyzing the results from the software applications and assets used. Gap: Interaction metrics, that is, how much of the underlying Roles and capacities were used in the current Interaction and sub-Interactions? This is difficult to estimate completely since the Interactions make use of shared resources. Thus the Roles and capacities for a particular type of Interaction are difficult to isolate. There is a gap in available technologies for activity-based costing.	Time and project management systems obtain high-level information on the use of resources; it is difficult to relate this to the actual Interactions.
Service Roles *Agents SS, ORR, Knowledge needed*	The tools here monitor the isolated Agent performance and also help apply knowledge when serving the Interactions in the operations layer. Many tools have features to identify the IRR and ORR of a component. Knowledge mining can be used to relate Agent instance performance within an Interaction. Gap: The gap is in identifying the Role Set ORR performance related to a specific Interaction. There is a gap in available technologies.	Component monitoring tools. These are tools (like the funnel analyzer[iv], database monitoring[v], web server monitoring, etc.). Monitoring of Roles can be accomplished by tools that mine/synthesize the overall based on the performance of Agents contributing to a Role. On-line training and work instructions are examples of Role enabling systems. Starlight is an example that can relate Agent performance to Role performance.
Infrastructure *Infrastructure used, capacity*	The tools/methods in this layer monitor the specific IT infrastructure components and the traffic between them to give us the current state (SS) of the system. For example, tools like HPs SiteScope help in monitoring the performance of distributed IT infrastructure assets. It is web-based software that provides a centralized view of the entire infrastructure in real time. The highlight is that you can gain the real-time information you need to verify operations, stay apprised of problems and quickly address bottlenecks in your system. For example, if a database server is not running and this leads to bottlenecks in the system when the application server starts issuing Requests to the database server, the deviation in the performance of the database server can be identified proactively (during reduced availability itself as opposed to when the server completely shuts down) Thus the problem can be diagnosed and corrected early resulting in performance improvement. This database also serves as the primary point of accountability for the life-cycle management of information technology assets throughout the organization. Dynamic operations monitoring tools are focused on monitoring the live system while it is in execution and providing that information to the user to improve the performance of the system. Gap: Maximizing the performance and lifetime value of complex assets can be done by improving return on assets, decrease cost	Monitoring of IT infrastructure and operations. InsightETE[vi] provides solutions for measuring and troubleshooting IT system performance by monitoring flow of information to and from IT users. It measures availability and response in real time and helps track service levels and reduce outages. Another example is IBM's Asset management solution called Maximo™ Consists of six key management modules — asset, work, service, contract, materials and procurement management

	and risk, increase productivity and improve asset-related decision making. This helps closely align assets with overall business strategy. However, the ability to relate capacity to the individual Request types and Request fluctuations that originate business Interactions is missing. This is needed to have precise accountability between the capacity and its point of use.	

Vertical Traceability Gaps

Generally speaking, the tools/methods identified above allow us to monitor each *Business, Infrastructure, Operations, Strategy* dimensions and the constituent Agents individually. However, they do not allow us to easily relate the use of multiply Agents vertically to Interactions and BioS goals. For example, we cannot easily relate the capacity of an Infrastructure component to the Business Interaction capacity that it facilitates. This requires a capability to trace Requests across the layers. We refer to this as *vertical or BioS traceability*. With this ability we can begin to align capacity and S-S attributes more closely to fluctuating IRRs and desired ORRs at the business layers.

However, this vertical traceability is missing. One approach is to dynamically propagate the identifier attribute of the initiating Request to retain full traceability [18]. Another is to apply data mining strategies (with some starting points in [19][20][21]). Finally, if this vertical traceability is made available, organizations can also be armed with predictive knowledge that will allow them to manage by objectives.

Specifically, the tool requirements for vertical traceability area can be divided into 1) knowledge mining, and 2) predictive capability for continuous improvement. These categories and their state-of-art are explored further in the table below.

Knowledge mining	Starting with operations monitoring logs from the different layers of systems we can analyze to gain useful insights into Operations. For example, these logs can be analyzed using tools like Starlight which gives a conceptual | visual understanding of relationships across the layers. Gaps: The weakness that remains is the fine granularity of execution traceability needed to truly understand how goal increments in the higher dimensions will be achieved.	A tool example is Starlight [17]. Other specific types of tools also exist for particular type of logs, for example web logs can be analyzed using tools like web funnel analyzer to give more insights into user behavior in the web system.
Predictive methods	While system dynamics and simulation tools can help in analyzing to-be performance based on future needs, there are	Analysis methods that justify investment based on increased future

	few principles that relate current value to future value of services. For example, how will increased availability/capacity actually increase the BioS value of an Interaction? Gaps: The theory of service value is missing. For example, we really do not know why or how retaining certain architecture options will increase the value of future Interactions. These sorts of questions need to be answered for improved EA governance.	BioS value are relevant.

The example simulations and knowledge mining illustrate the advantages of integrating these features and methods.

5. Model-Driven Engineering Environment

The method for achieving an ACE is proposed as a Continuous Improvement (CI) environment for complex systems based on an integrated method and toolset for:

- Execution monitoring of Agents using CI Agents,
- Simulation,
- Intelligence Mining, and
- Continuous Service Interaction(s) Improvement.

Execution Monitoring

Using automated or manual instrumentation Continuous Improvement (CI) Agents determine the IRR/ORR rate for each Agent in the Role Set associated with the Interactions. Additionally the S-S attribute values are captured at execution using a variety of monitoring tools mentioned above.

Simulation

We next show how system dynamics tools can use monitored information to model a specific scenario in the City which receives customer Requests through the Customer Service Center (also known as the Help Desk). The focus of the simulation is the traceability of the response path that Requests take. This is of particular interest because different paths provide different Agents to provision the Interaction at different

costs. We will show how the simulation provides us with overall throughput rates and decision-making insights even though the variables are based on *average* monitored values.

The simulation is based on Vensim[vii] where the *stock* (IRR) and *flow* (related to ORR) variables in the diagrams are based on the following key principles:

- **Stock:** Any entity that accumulates or depletes over time (IRR).
- **Flow:** Rate of change in a stock (determines ORR).

Based on the City case study we identified the following stock-and-flow variables related to the basic Interaction pattern in Figure 5.

- **Incoming Requests:** This is the number or rate of Requests flowing into the organization.
- **Request queue:** This is a stock of all the Requests that have come in and are waiting in the queue to be executed. If the Requests get executed faster than the rate at which they arrive then the Requests in the queue will decrease over time.
- **Requests delivered:** This is the stock of Requests that were allocated the Agent resources (infrastructure, people etc) and hence have completed their execution.
- **Execution:** This is a flow of Requests from the Request queue to the stage that they are completed. This flow is affected by the factors like the Agents available and their productivity.
- **Rework:** This is the reverse flow from the Requests that have been completed to the Requests that are queued. Some Requests may not have been resolved correctly (due to lack of skills and knowledge) and hence they come back into the Request queue. This is affected by variables like error fraction and the average time to discover errors.

The values of these important variables and flows under consideration are as follows:

Variable name	Units	Value/Equation	Rationale/Comments
Incoming Requests Rate (IRR)	Request	Initially assumed as 4000 (constant) based on City data	-
Average time to discover errors	Months	2	-
Help Desk staff	Person (Agents)	45	
Average Help Desk productivity	Request/(person*Month)	50	Actually multiple Agents are involved - help desk, CRM system and so on

% of unresolved Requests	-	Starts from 0.5 and then reduces by amounts of 0.05, 0.07, 0.1 after 12, 24 and 36 months respectively	As the time passes, the staff gets better at resolving the Requests and hence this value decreases over time
Error fraction	-	Is directly equal to the % of unresolved Requests	As the % of unresolved Requests decreases over time, the error fraction also reduces by same amount

Aspects of the ontology and the underlying Vensim simulation are illustrated in Figure 5 and are based on the ontology in Figure 4. The BioS goals are also illustrated in the top half of Figure 5. The following is an analysis of the City Help Desk and how the triage pattern was used to simulate the improvement to the throughput of Requests in the City. We will look at progressive applications of the pattern to make decisions about the levels of achievable improvement.

Case 1: First, consider the case when all Requests go through the same route in the Help Desk and there is *no classification* of the Requests and internal attributes that reflect the complexity of its requirements. (It is helpful to think of the emergency room analogy mentioned earlier).

The results of the simulation are in Case 1: Level 0. Here we note that both routine and non-routine Requests are processed in the same manner. Consequently - routine ones take longer to execute and non routine ones not getting executed properly.

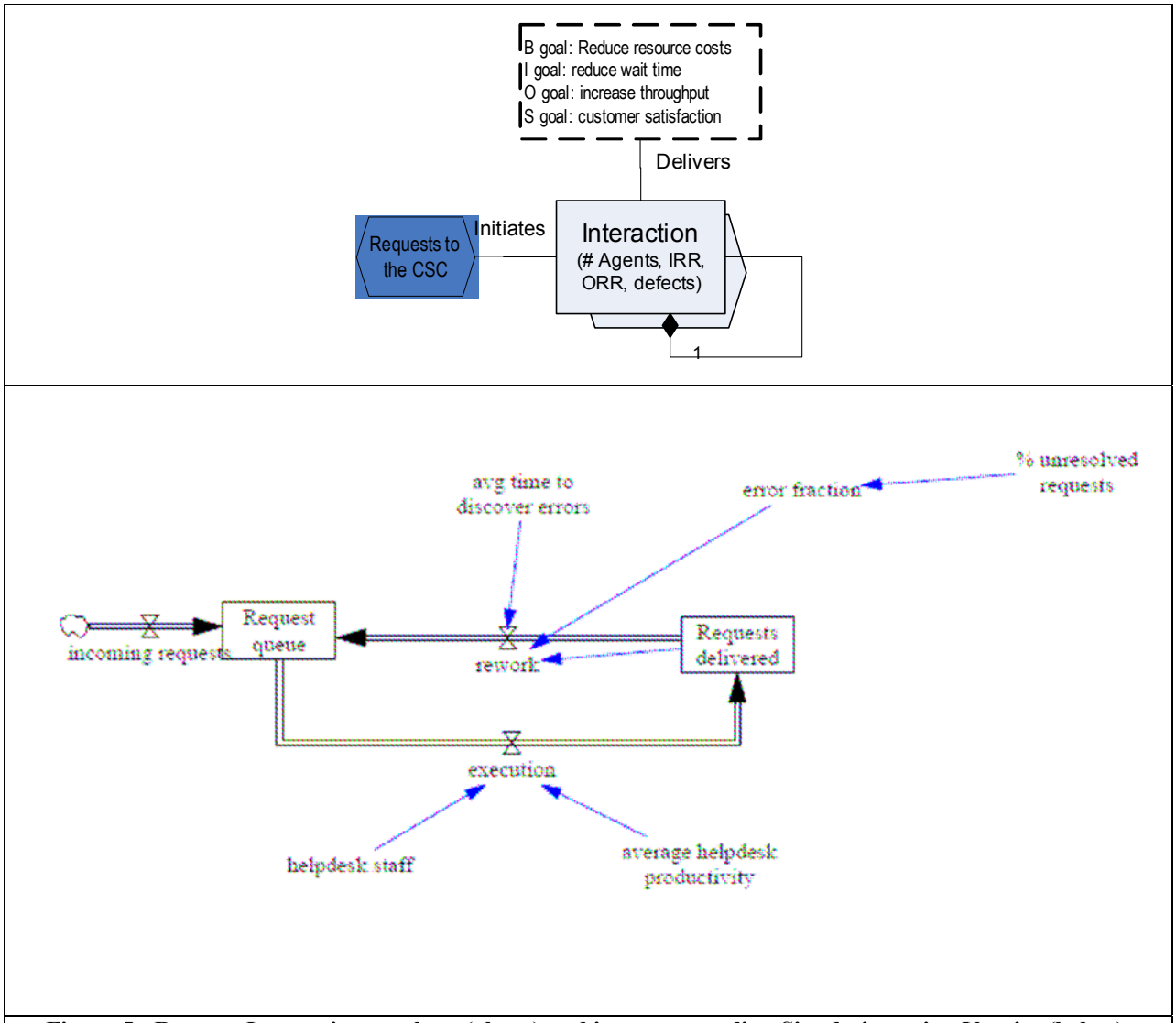


Figure 5: Request-Interaction ontology (above) and its corresponding Simulation using Vensim (below).

Intelligence Mining

To improve the performance of this complex system we next ask the questions:

1. What specific types of Requests take longer?
2. Can we identify the non-routine Request characteristics and resources to improve throughput?
3. How can we improve throughput of the Interactions, keeping the resource numbers the same?

The first two questions are *not directly* answered by the simulation but by the execution details of individual Requests. For this we need a different technology which

can help us in analyzing each individual Requests, the correlations between the different elements and their attributes, and identify any trends. We can use the tool Starlight (developed by Battelle) for this purpose. Starlight ingests large quantities of data and generates visualizations of structured values, conditioned with color and shape in a manner that make data easy to interpret, analyze and gain effective insights. For example, in the case of the City Help Desk we could ask interesting questions like :

- Which Requests have which non-routine attributes that are taking the longest time and which Agent assignments do they receive as they go through triage?
- If there are maximum errors or delays in the Requests serviced by a particular Role, then does that indicate misclassification at triage or perhaps even the need for another potential triage level to be added?
- Do these Requests cluster around specific non-routine characteristics?

A related example is shown below using helpdesk data. Figure 6 shows an example of the many interesting views called 'Link Array view' generated using Starlight. Here the interrelationships among the properties like RoleName, BrowserType and ErrorCount for the different requests coming into the helpdesk are visible.

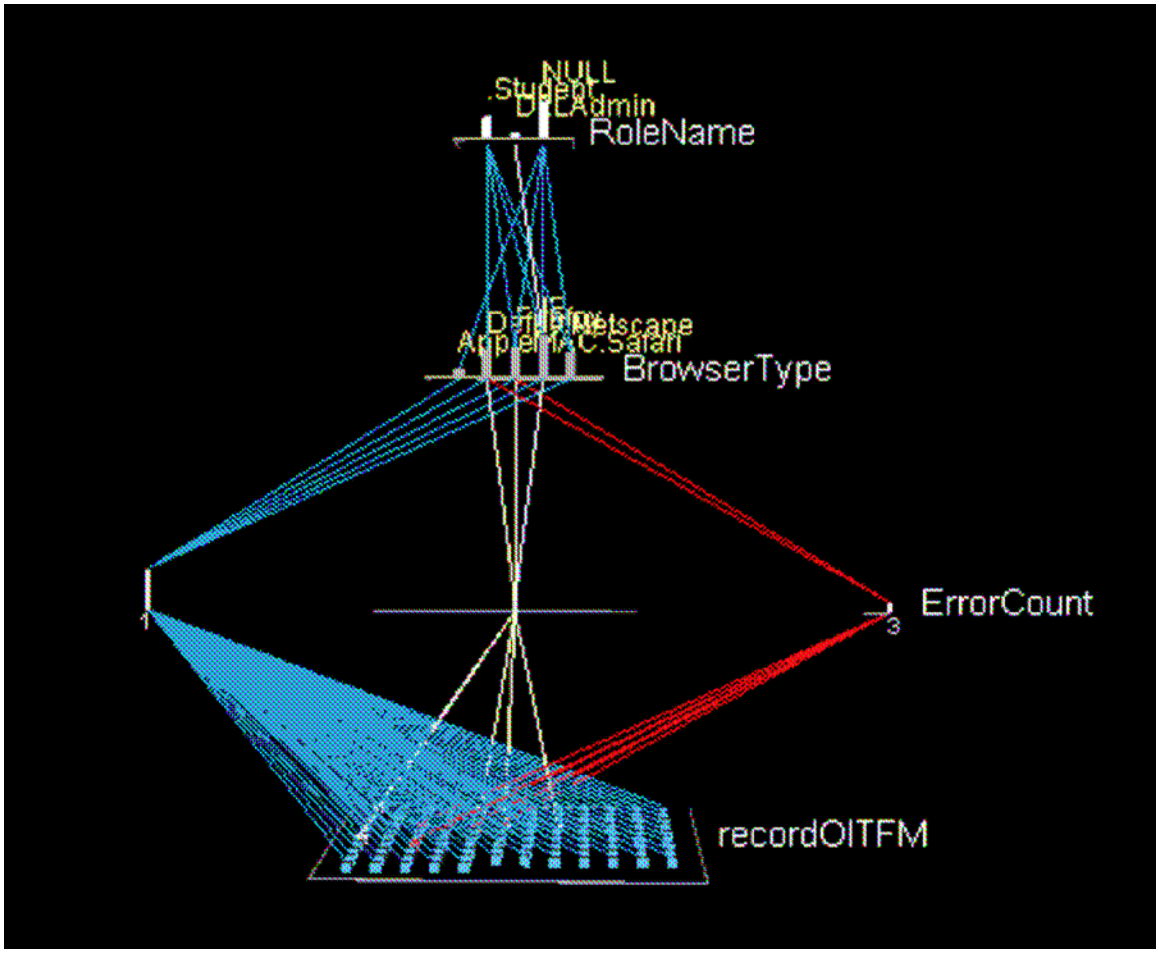


Figure 6: Starlight example with Office of Information Technology, OSU helpdesk data

With such visualization we can begin to get insights like:

- Which roles (Student/Admin/Other) encounter how many errors? This could identify the potential (training or quality) problems in the sections of the system that could in-turn merit system specific triage rules.

- Which browser types have the most errors? Could indicate an incompatibility with browser types - an incident that can be quickly resolved at Level 0.

With this type of intelligence we can next ask if we can simply classify incoming Requests into routine (standard or regular) and non routine (non standard) ones to get an advantage in terms of response time and performance. This leads us to the simulation case 2.

Case 2: In Figure 6 we add a new level 1 of triage, with the Requests now being classified as routine and non-routine. The stocks and flows in Figure 5 are retained but the following changes take place for the Level 1 triage variables as in the following table.

The Requests are now classified as routine and non-routine on the basis of the average Request complexity. Requests of higher complexity move to the path that assigns higher skills commensurate with request complexity. As the execution of Requests is now separate, we have separate staff and productivity variables for both Request types. This also reflects the distinction between the types of resources that are assigned to work on particular Requests. As Requests are explicitly classified into new categories, the Agents servicing the Requests should be of appropriate type (e.g. staff skill and experience level). From the City case study, we know that this does not hold because requestors often bypass the Help Desk and directly contact higher level staff about Requests without evaluation of the Requests. This is problematic because higher cost resources are being drawn in to work on less complex Requests. In this triage scenario, the Requests of higher complexity are now explicitly moved up to level 1 (and higher levels in subsequent scenarios).

The error fraction now gets influenced by not only the unresolved Requests but also the misclassified Requests for example a non routine Request which got triaged as routine and hence did not get executed correctly and landed back totally or partially as a new Request in the Request queue. The 'total Requests delivered' is an additive variable that adds the Requests serviced at all layers. Thus the values of the important variables and flows under consideration for Level 1are now as follows:

Variable name	Units	Value/Equation	Rationale
Incoming Requests	Request	Initially assumed as 4000 (constant) based on City data	-
Average time to discover errors	Months	1.25	Since there is at least one level of triage the average time to discover errors reduces since now the Requests are matched and executed so errors can be discovered earlier
Help Desk staff	Person	30	The total staff from fig 5 to fig 6 should remain same since no additional resources should be added. So 15 staff members are moved from the Help Desk to the next level for handling more complex Requests
L1 (level 1) staff	Person	15	Same as above

Average Help Desk productivity	Request/(person*Month)	60	The productivity increases from fig 5 to fig 6 since the staff is now getting those Requests that it is more capable of handling
Average L1 productivity	Request/(person*Month)	60	-
% of unresolved Requests	-	Starts from 0.4 and then reduces by amounts of 0.05, 0.07, 0.1 after 12, 24 and 36 months respectively	As the time passes, the staff gets better at resolving the Requests and hence this value decreases over time. Also the initial value decreases from 0.5 in fig 5 to 0.4, since the addition of one triage level leads to better servicing of Requests since right Requests go to the right people to some extent and thus the unresolved Request% decreases
% of misclassified Requests	-	Starts from 0.35 and then reduces by amounts of 0.05, 0.07, 0.1 after 12, 24 and 36 months respectively	Similar rationale as for unresolved Requests. This variable did not exist in fig 5 since there was no triage and hence no question of misclassification
Error fraction	-	Is equal to the sum of the above 2 percentages	As the above percentages decrease over time, the error fraction also reduces by same amount

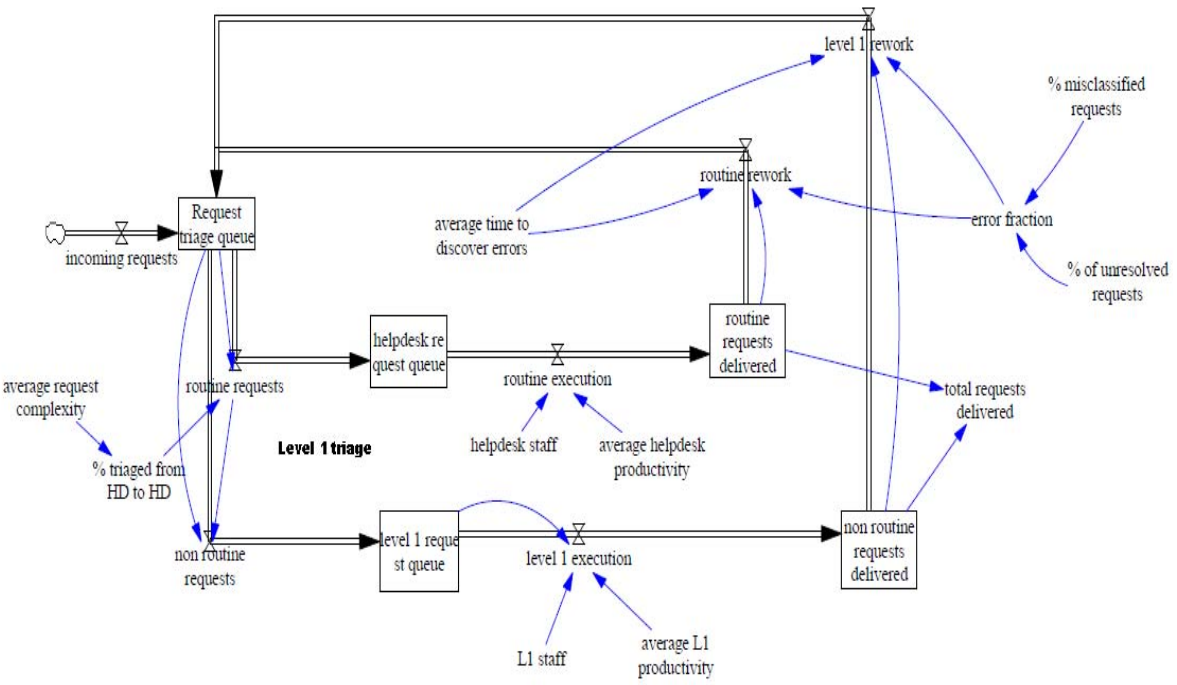


Figure 6: Triage at the Customer Service Center - simple 1-level model

Case 1: Level 0

Case 2: After adding Level 1

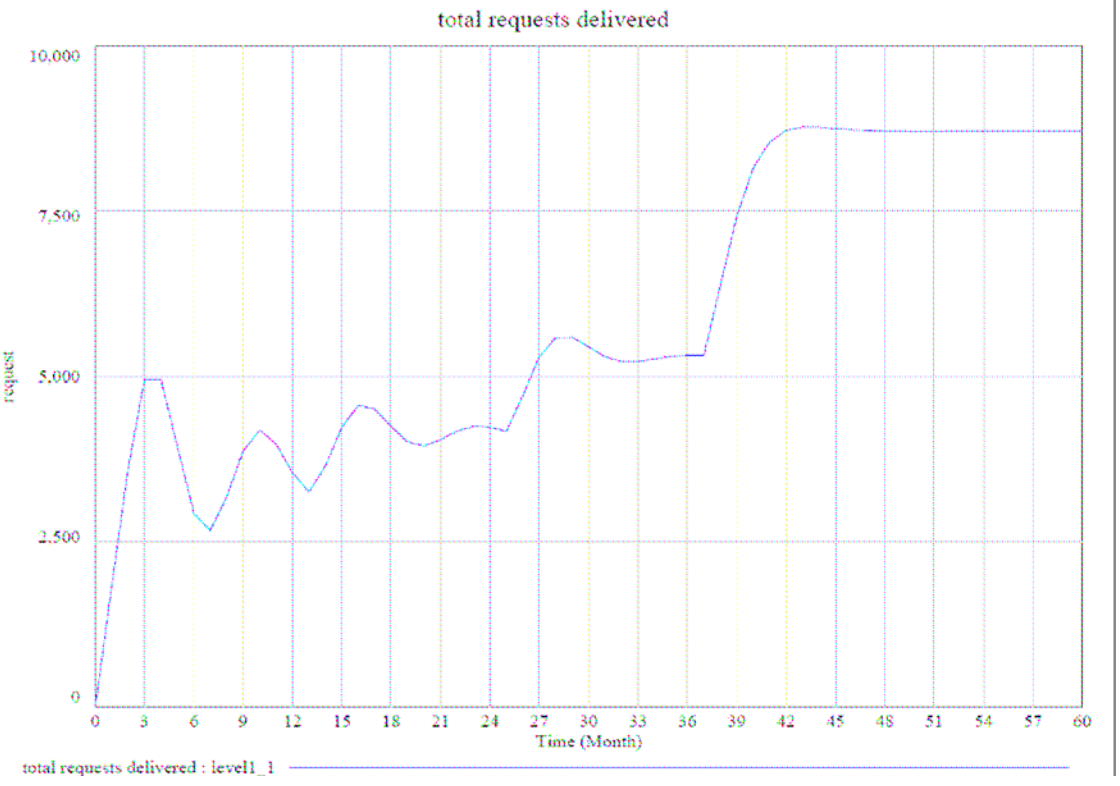


Figure 7: Before and after the addition of Level 1 triage.

The simulation in Figure 7 supports the hypothesis that the Help Desk productivity would improve from what it was without the Level 1 triage since the Help Desk staff is going to handle fewer Requests and also specifically those routine Requests that it is capable of handling instead of being overwhelmed with all the Requests that need higher skills. The graphs illustrate a throughput improvement of about 8% (from 8050 to 8710 Requests).

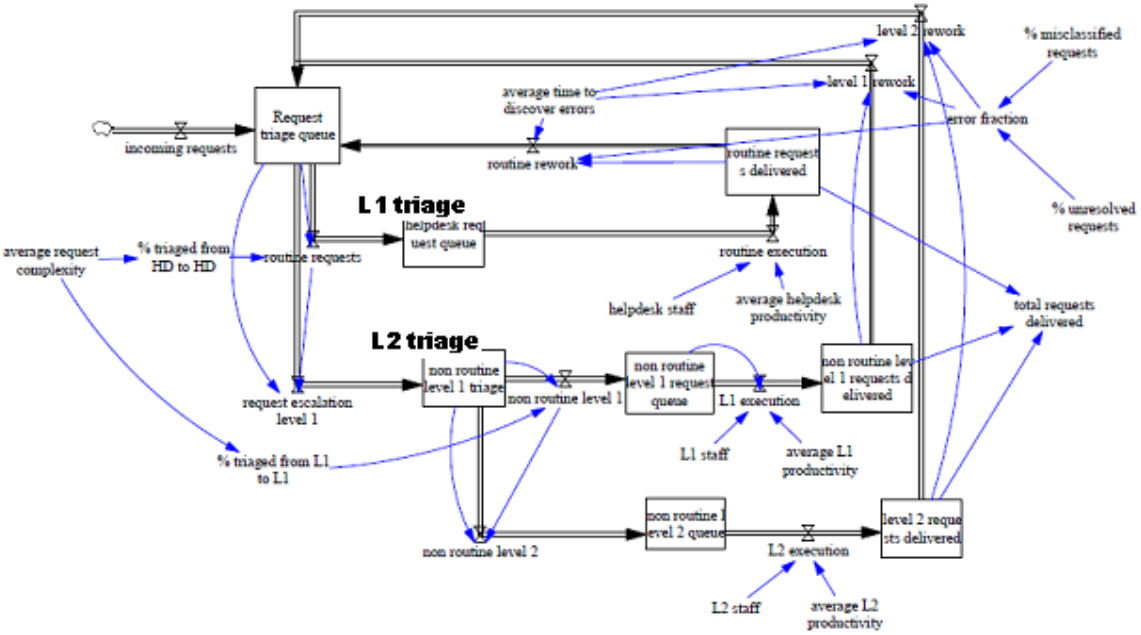


Figure 8: Triage at the Customer Service Center - 2-level model

Continuous Service Interaction(s) Improvement.

Continuous Service Interaction Improvement questions now become:

- How much further can we improve throughput of the Interaction?
- How can we further identify the non-routine Request characteristics to do better?

Case 3: With new intelligence about Request and Agent execution metrics we can develop better triaging rules and explore whether the BioS goals can be better achieved with additional Level 2, Level 3, Level 4 etc. for continuous improvement.

We choose to illustrate this through re-applying the pattern to an example where the stock and flows that were modeled in Level 1 triage are simply extended to

accommodate 2 new levels of triage. The hypothesis at Level 2 triage is that the Level 1 productivity (Figure 6) would improve since the Level 1 Agents are going to handle fewer Requests and also specifically only those Requests that they are capable of resolving. However it may still not be the most optimal since there are only 2 levels of triage and some Requests could come here and still they may not have an appropriate level dedicated to service them.

This sort of insight and goal validation is achieved by monitoring the actual execution - using the CI agents - to 1) validate the variables used in the simulation and 2) see if other new information is arising from the individual Requests and Interactions.

In the City example the mined Request classification levels at the Help Desk corresponded to:

- Level 0: Incident Requests dealing with passwords, move, add, change etc.
- Level 1: Incidents not reproducible or requiring more skills.
- Level Projects: staff managers for large projects (not including the project implementation resources)
- Level 3: Reproducible failures with networks, software etc.
- Level EA: staff addressing Special small service projects.

At this point it is important to note that the skills at Level 1 triage are not always adequate to do more advanced classification and triaging. Hence the triage cannot be made into 5 levels from the start. The helpdesk agents just possess enough knowledge to triage the requests into routine versus non routine. Similarly, the level 1 staff knows just enough about the request to know if it can handle it on its own (level 1 queue) or if it needs to be escalated (level 2). It is at the level 2 of the triage that the agents are knowledgeable enough to know if it can handle the request on its own, or if it needs to be triaged further. Thus, the Level 2 triage in Figure 8 is actually expanded with nested sub-triage levels as in Figure 9. This also requires more attributes to be used in characterizing the request and associated skill level to actually apply the triaging. Thus the Level 2 triage needs more experienced engineers.

The values of the important variables and flows are now as follows:

Variable name	Units	Value/Equation	Rationale
Incoming Requests	Request	Initially assumed as 4000 (constant) based on City data	-
Average time to discover	Months	1	Since there more levels of triage, the

errors			average time to discover errors reduces since now the Requests are matched and executed so errors can be discovered earlier
Help Desk staff	person	6	The total staff from fig 7 to fig 8 should remain same since no additional resources should be added. So only 20 remain at this level while the others move to the next levels for handling more complex Requests
L1 (level 1) staff	person	10	Explained above
L2 (level 2) staff	person	10	Explained above
EA staff	person	2	Explained above
Project staff	person	12	Explained above
L3 (level 3) staff	person	5	Explained above
Average Help Desk productivity	Request/(person*Month)	120	The productivity increases from fig 7 to fig 8 since the staff is now getting those Requests that it is more capable of handling
Average L1 productivity	Request/(person*Month)	100	Similar as above
Average L2 productivity	Request/(person*Month)	100	Similar as above
Average EA productivity	Request/(person*Month)	(1/6)/2	Similar as above
Average project productivity	Request/(person*Month)	1	Similar as above
Average L3 productivity	Request/(person*Month)	10	Similar as above
% of unresolved Requests	-	Starts from 0.3 and then reduces by amounts of 0.05, 0.07, 0.1 after 12, 24 and 36 months respectively	As the time passes, the staff gets better at resolving the Requests and hence this value decreases over time. Also the initial value decreases from 0.35 in fig 7 to 0.3 in fig 8, since the addition of one triage level leads to better servicing of Requests since right Requests go to the right people to some extent and thus the unresolved Request% decreases
% of misclassified Requests	-	Starts from 0.3 and then reduces by amounts of 0.05, 0.07, 0.1 after 12, 24 and 36 months respectively	Similar rationale as for unresolved Requests.
Error fraction	-	Is equal to the sum of the above 2 percentages	As the above percentages decrease over time, the error fraction also reduces by same amount

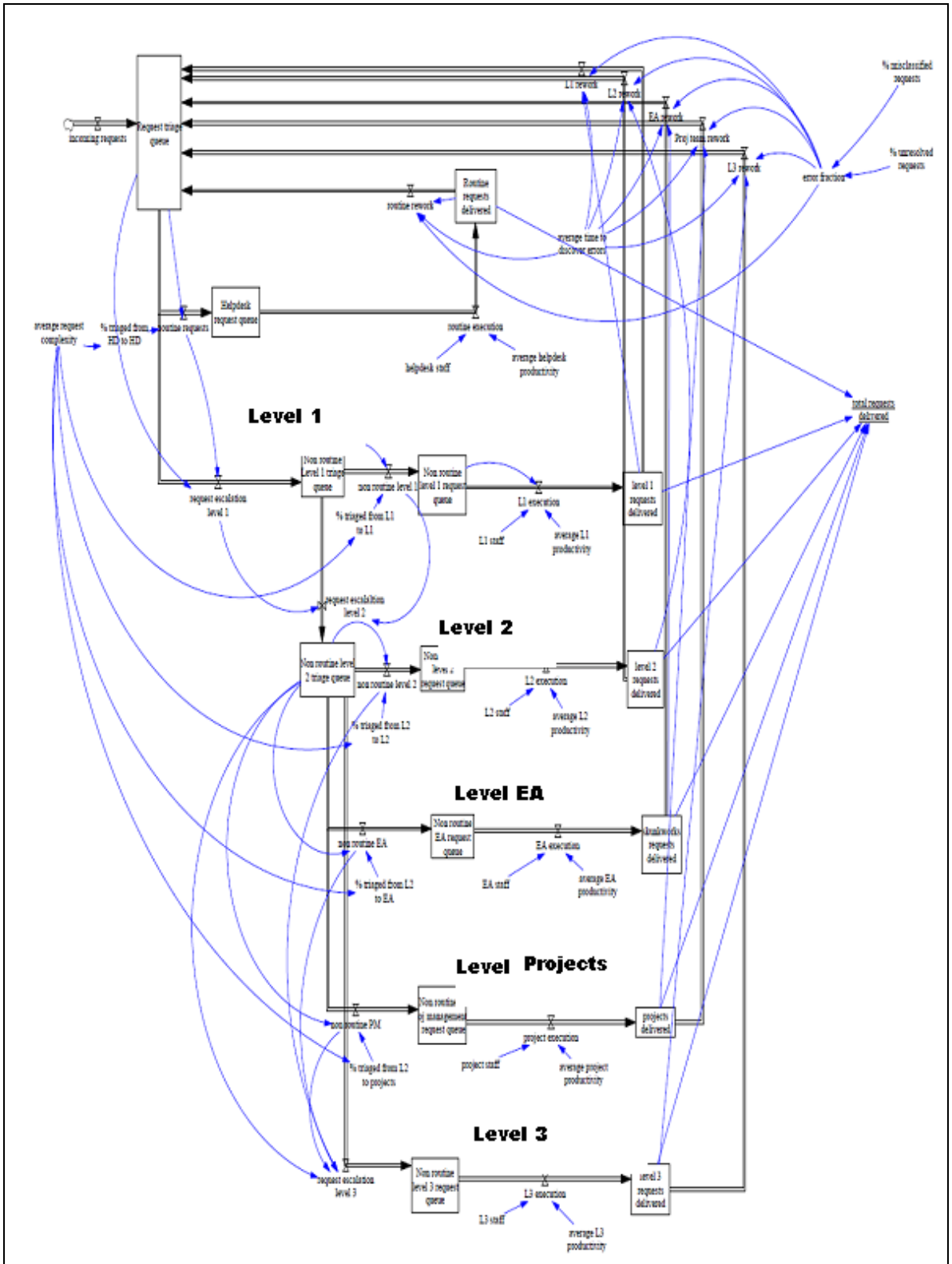


Figure 9: Case 3 and Multi-level triage to improve request throughput.

The resulting simulation gives us the final table to illustrate the predicted achievement of BioS goals based on four levels of Request complexity that existed.

Case 3 Simulation Results		
Level of triage	**Throughput** **(number of Requests delivered in 60 months)**	**Improvement (%)**
Level 0 (no triage)	8050	NA
Level 1	8710	8.2%
Level 2	10865	24.7%
Nested triage (level 2 triaged into 4 levels)	12408	14.2% better than level 2 42.5% better than level 1

6. Conclusions and Future Research

This example with real-world basis illustrates that the monitoring-simulation-mining-continuous improvement methodology based on the enterprise ontology shows that:

1) The use of the triage levels based on the classification of Request attributes internal to the Request can be used to improve throughput and identify the SLs achievable while maintaining the level of resources.

2) The SLs are predicted based on OLs attributes used as productivity numbers and so on.

3) If we mine the Requests for variation, we can dynamically triage these to direct them to receive the Agent services that, in turn, show performance improvement.

Thus we can begin to predict improvements based on mined knowledge, identification of new skills, additional Request classification levels and eventually making the non-routine Requests more routine in their processing. The challenges that remain in generalizing the methodology for managing the Adaptive Complex Enterprise based on objectives can now be identified.

Insights from Vertical Traceability

Assuming that we can capture all the information needed to build the associations and traceability among the elements of an enterprise above, what types of knowledge can we mine for decision making in the real-world? What integrated model-driven engineering tools and 'dashboard tools' can we build? What types of ACE objectives can we manage?

To answer these questions, we note that with the underlying ACE ontology presented above and the resulting vertical traceability, we can develop run-time tools and an integrated ACE environment that:

- Integrates business and system simulation tools to better pin-point Interactions needing improvement to meet Goals,
- Monitors Interaction between Business users and IT services in order to predict desired SLs and Agent capacity requirements without needing human intervention,
- Identifies Agent service improvements that meet OLs and better relate to stakeholder Goals,
- Identifies achievable OL performance in the context of the target environment and architecture,
- Provide trade-offs and costs associated with improving different Roles and Role sets,
- Quantifies the impact of a new services to be deployed into an existing environment, and
- Extends traceability, through mining and simulation capability, to new applications such as availability, and security.

We have provided a conceptual starting point for model-driven engineering with the above features along with a toolset for continuous improvement across BIOS.

Future Research

The immediate next step for research in this area is the development of a prototypical dashboard which is able to monitor and suggest improvements across

dimensions. Detailed requirements have to be developed starting with a standard XML based form for representing data that is shared between tools. The long term research is related to field deployment using Web Services as a way to implement CI Agents that provide enterprise wide visibility.

7. Acknowledgements

This research is supported in part by the National Science Foundation under Grant No. 0753710 and Members of the CERCS IUCRC Center for Enterprise Transformation and Innovation. We also wish to acknowledge members of Center and other graduates students and partners that contributed many insights.

8. References

1. Lucia, A. D., Fasano, F., Oliveto, R., & Tortora, G. (2007). Recovering traceability links in software artifact management systems using information retrieval methods. *ACM Transactions on Software Engineering and Methodology*, *16*, 4 (Sep. 2007), 13. DOI= http://doi.acm.org.proxy.lib.ohio-state.edu/10.1145/1276933.1276934
2. Kelleher, J (2005). A reusable traceability framework using patterns. *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE '05, ACM, New York, NY*, 50-55. DOI= http://doi.acm.org/10.1145/1107656.11076684
3. Asuncion, H. U., François, F., & Taylor, R. N (2007). An end-to-end industrial software traceability tool. *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering , ESEC-FSE '07. ACM, New York, NY,* 115-124. DOI= http://doi.acm.org.proxy.lib.ohio-state.edu/10.1145/1287624.1287642
4. Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., & Christina, S. (2005). Goal-Centric Traceability for Managing Non-Functional Requirements. *Proceedings of 27th International Conference on Software Engineering*.
5. Eric, S. K. Yu (1997). Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE'97)*.

6. Gotel, O. C. Z., & Finkelstein, A. C. W. (1994). An Analysis of Requirements Traceability Problem. *Proceedings of the First International Conference on Requirements Engineering*.
7. Bon, J., Pieper, M., & Annelies, V. (2005). *Foundations of IT Service Management Based on ITIL*. Van Haren Publishing.
8. Blevins, T. J., Spencer, J., & Waskiewicz, F.(2004). TOGAF ADM and MDA. *The Open Group and OMG*.
9. Anquetil, N., Grammel, B., Galvão, I., Noppen, J., Khan, S. S. , Arboleda, H., Rashid, A., & Garcia, A.(2008). Traceability for Model Driven, Software Product Line Engineering. *ECMDA Traceability Workshop Proceedings*.
10. Ramanathan, J., & Ramnath, R. Co-engineering the Adaptive Business and Technologies. Collaborative Service-Ontology and Enterprise Architecture Principles for Best Practice Deployment and Performance.
11. Ramnath, R., & Ramanathan, J. (2008). Integrating Goal Modeling and Execution in Adaptive Complex Enterprises. *Proceedings of the ACM Symposium of Applied Computing*, *ACM,* 523-539.
12. Sampson, G. (2003). The myth of diminishing firms. *Communications of the ACM*, *46*(11), 25-28.
13. Ramanathan, J. (2005). Fractal Architecture for the Adaptive Complex Enterprise. *Communications of the ACM*, *48*(5), 51-57.
14. Gupta, V., Mukri, F., Ramanathan, J., Ramnath, R., & Yackovich K. (2009). CitiScapes: Architecture for eGovernment Effectiveness. To appear in the *42nd Hawaii International Conference on Systems Sciences, Waikoloa, HI: IEEE Computer Society*.
15. Kumar, A., Raghavan, P., Ramanathan, J., & Ramnath, R. (2008). Enterprise Interaction Ontology for Change Impact Analysis of Complex Systems. *IEEE Asia-Pacific Services Computing Conference (IEEE APSCC)*.
16. Hartung, R., L., Bolinger, J., Ramanathan, J. (2008). Ontology for Enterprise Modeling. *Knowledge-Based Intelligent Information and Engineering Systems 12th International Conference, KES*.

17. http://www.directionsmag.com/article.php?article_id=775&trv=1

18. Agarwala, S., Alegre, F., Schwan K., & Mehalingham, J. (2007 June). E2EProf: Automated End-to-End Performance Management for Enterprise Systems. *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Washington DC, IEEE Computer Society*, 749-758.

19. Belkin, N.J. & Croft, B. (1992 December). Information Filtering and Information Retrieval: Two Sides of the Same Coin. *Communications of the ACM, 35*(12), 29-38.

20. Bonaceto, C. & Burns, K. (2005 November). Using Cognitive Engineering to Improve Systems Engineering. *The MITRE Corporation*, Retrieved November 23, 2008, from http://www.mitre.org/work/tech_papers/tech_papers_05/05_1361/.

21. Ding, Xiaowen. Liu, Bing. Yu, Philip S. (2008). Web Search and Web Data Mining archive. *Proceedings of the international conference on Web search and web data mining*, *ACM,* 231-240.

[i] http://www.itil-officialsite.com/home/home.asp

[ii] A primary process directly satisfies a business need while a secondary process is generally any process with the sole purpose of enabling a primary process.

[iii] http://www.vensim.com/software.html

[iv] http://linux.wareseeker.com/Internet/funnel-web-analyzer-5.0.zip/319940

[v] http://www.uptimesoftware.com/

[vi]Insightete: http://www.insightete.com/

[vii] http://www.vensim.com/software.html