Integrating Research Root Cause Analysis Tools into a Commercial IT Service Manager

by

Xiaochun Li
B.Sc., University of Victoria, 2008

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

Department of Computer Science

Supervisory Committee

Integrating Research Root Cause Analysis Tools into a Commercial IT Service Manager

by

Xiaochun Li
B.Sc., University of Victoria, 2008

Supervisory Committee

Dr. Hausi A. Müller (Department of Computer Science)
Supervisor

Dr. Alex Thomo (Department of Computer Science)
Departmental Member

Dr. Ulrike Stege (Department of Computer Science)
Departmental Member

Abstract

Supervisory Committee
Hausi A. Müller (Department of Computer Science)
Supervisor

Alex Thomo (Department of Computer Science)
Departmental Member

Ulrike Stege (Department of Computer Science)
Departmental Member

IT environments are turning more complex by the day and this trend is poised to rise in the coming years. To manage IT resources and maximize productivity better, large organizations are striving for better methods to control their current environments. They also have to prepare for future complexity growth as their environments cater to the growing IT needs. In the current economic recession, organizations are not only threatened by the growing complexity, but also have to cope with limited personnel due to financial constraints. Organizations are ardent about obtaining new technology to have firmer control on different platforms, vendors, and solutions at a reasonable cost. At the same time, this new technology must deliver quality services that can effectively fulfill customer needs.

To deal with IT management challenges, CA developed Spectrum Service Assurance Manager (SAM), a product by CA Inc. (formerly Computer Associates) to solve complex IT environment service management problems. SAM can provide organizations with a

wide-ranging view of their multi-faceted IT environments by providing vital pieces of information that no other software can perceive. Thus, SAM can monitor and manage systems, databases, networks, applications, and end-user experiences. Although, this technology is able to detect many errors and problems, it still lacks a good mechanism to diagnose the detected problems and uncover their root causes for end users to fix.

Four research groups from Universities of Alberta, Toronto, Victoria and Waterloo—under the auspices of the Consortium for Software Engineering Research—built different tools for root-cause analysis and detection. To integrate these solutions, these research groups worked together with CA Inc. to produce a web-based integration tool to integrate these add-ons into the main SAM application. The resulting framework does not affect any of SAM's existing features as the additions only involve a new web communication layer that acts from the core of the software to detect and present root causes. The detection tools only parse the log files for vital information and thus the core functionality of the software remains unaffected.

My contributions to this research project are presented in this thesis. In the beginning of this thesis, I report on background research on SAM and describe how it is going to solve the increasing complexity problem in IT environments. Later on, I propose two software integration approaches to integrate root cause diagnosis tools with SAM and briefly describe CA's latest software integration framework Catalyst. Towards the end of this thesis, I compare our integration solution with Catalyst, and discuss advantages and disadvantages of these integration solutions.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to extend special appreciation to my supervisor Dr. Hausi A. Müller, who has not only provided me with an excellent research opportunity and environment, but also bestowed valuable advice, guidance, and support throughout my graduate studies.

Also I would like to thank Serge Mankovskii and Hamzeh Zawawy from CA Canada Inc., Qin Zhu, Ron Desmarais, Priyanka Gupta, Lei Lin, Alexey Rudkovskiy and all other members of the Rigi research group at the University of Victoria for their continuous support throughout my studies.

Chapter 1 Introduction

1.1 Motivation

The ongoing global economic recession has a huge budget impact on many of today's businesses. Companies have to deploy better and more complex business solutions which involve less human labour for their daily operation to help reduce spending and avoid overlapping their reduced budgets. These new complex solutions make the environments more intricate, thereby demanding more attention and resources. Figure 1 illustrates dependencies in an IT environment and shows how complex such an IT environment can become—it is difficult to figure out what is going to happen if we modify one of the applications. Thus, managing this environment is a real challenge. To pursue a solution to combat the rising complexity issues, organizations need to adapt new technologies, including web-based applications, to integrate IT management tools better to improve efficiency and minimize losses.

Besides the increasing complexity in IT environment, organizations are profoundly concerned about the scope to which the complexity can increase in the rapidly developing industry. It is imperative for these organizations to take steps now, in order to maintain quality standards and profound control over their operations in the future [4].

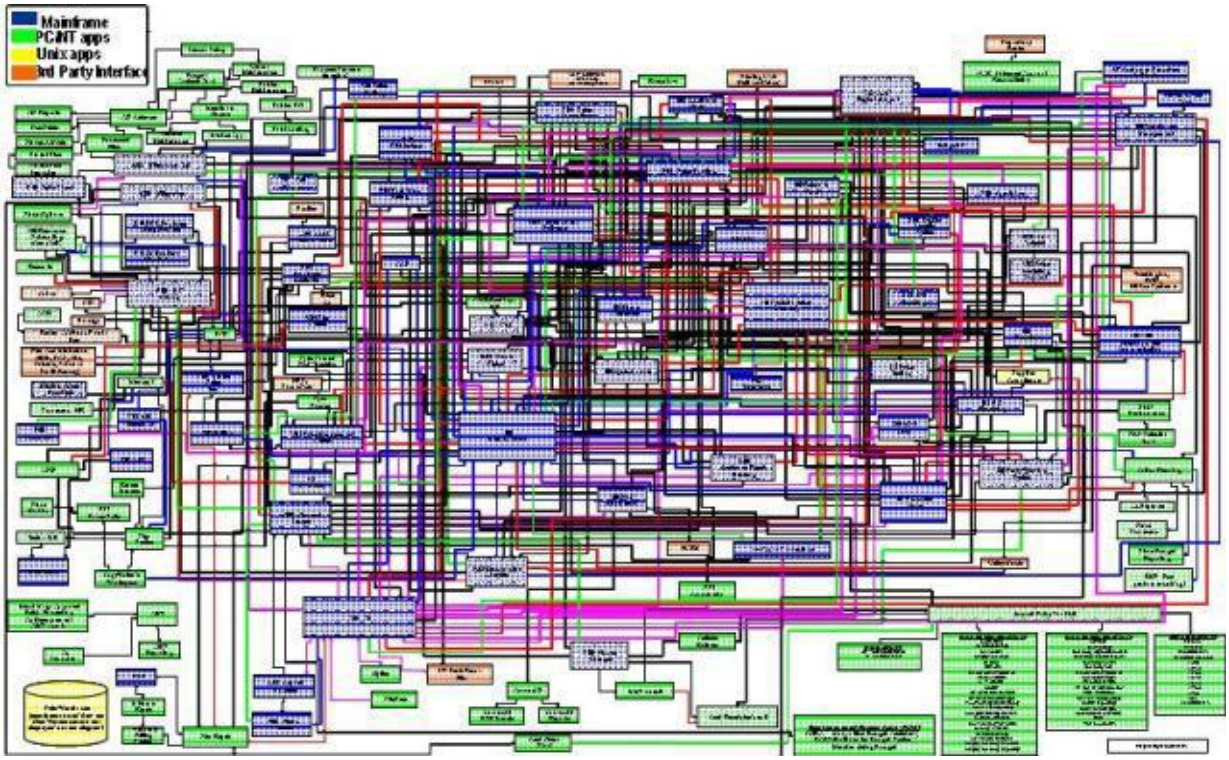


Figure 1: Actual point-to-point infrastructure [14]

Furthermore, organizations are witnessing an increasing number of competitors who are moving into the industry. Thus, they are strained to invest more to advertise their business and maintain their customers. This introduces more financial constraints for them in these tough economic times. With limited budgets, companies have to be even more careful to make the right choices to control their environments. A wrong choice can fatally derail them from their business model which can result in customers opting for other companies. Since, the issue of managing IT environments cannot be ignored, many organizations are investing more to monitor and allocate services rather than investing in growth, research and development. A simplified control over IT resources can lift a huge burden off these organizations and will allow them to focus on other parts of their expanding business.

As mentioned above, organizations are allocating more money to control IT complexity instead of spending that money on their growth and development. Business problems are mostly subjugated by implementing new technology to provide enhanced control to ease and monitor complex tasks. Although, these new technologies could bring countless advantages such as better data modelling and reduced fault alarm, they also require skilled personnel to manage these new interfaces. Thus, an organization has to also invest in the workforce that can operate the new technology and provide support if the systems are prone to downtime. To tackle this issue, CA Inc. has developed a new product, *Enterprise Information Technology Management (EITM),* through which the company aims to tackle IT management challenges and introduce a new level of control over IT assets.

1.2 Approach

"*Enterprise IT Management (EITM) is a strategy conceived and developed by CA Inc. detailing how organizations can transform the management of IT in order to maximize business value. As a strategy for increasing the business relevance of the IT function, EITM considers the need for IT organizations to start operating as a service-based business. That is, ensuring investments are prioritized according to business strategy and that operational efficiencies can be more quickly realized and costs reduced when IT processes are integrated and automated*." [1]

To solve IT management issues that have been initiated by current tools such as Nimsoft [39], ConnectWise [40], *CA Spectrum® Service Assurance Manager (SAM)* has been developed to enhance service quality while lowering the costs of IT management. As a result, SAM provides

users with an end-to-end view of all the services such as any computers, routers, hard drives, and network etc, that are monitored in real-time through a simple user interface called *Dashboard*. It also provides end users with a service-centric view of the IT environment they are monitoring. With this layout, users are always aware of the risk level of each service and can prioritize these risks so they can be dealt with efficiently and in a timely manner.

However, SAM still has does not solve all IT management issues. One of the most significant disadvantages of SAM is that it does not have a good mechanism to detect the root causes of detected defects automatically. In other words, if a problem arises such as a CPU outage in an IT environment, SAM will not be able to figure out the why CPU has all been occupied. End user will have to intervene to obtain more details and find out which processes caused the CPU outage. In many cases, such as the CPU outage example we mentioned above, SAM still requires human intervention to troubleshoot and scrutinize the root causes of defects. To address this concern, CA Inc. started an industrial collaborative research project to derive solutions to address the problem of automating root cause analysis. The NSERC Collaborative Research and Development (CRD) project entitled *Logging, Monitoring and Diagnosis Systems for Enterprise Software Application (LMD)* aims to investigate concepts and methods to evolve monitoring capabilities as well as develop prototypes to achieve more authoritative diagnosing [29].

In order to make SAM capable of analyzing errors and problems, different universities initiated development of software tools to parse information from SAM's error messages and log files and interpret that information to provide knowledge to analyze probable future defects. The

development of these tools was just the first phase of the solution. The second phase involved the integration of these solutions onto SAM without affecting SAM's existing functionality.

To integrate root-cause detection solutions into SAM's framework, we first looked at and purposed an Enterprise Application Integration (EAI) based integration solution, and after some thorough research, we figured out that a web-based software integration approach will work better with our root cause analysis tools and SAM.

1.3 Thesis Outline

Chapter 2 provides background on CA Spectrum® Service Assurance Manager and how it conforms to the goals of EITM. Chapter 3 describes details of the Enterprise Application Integration (EAI) approach to integrate third party solutions with SAM. Chapter 4 introduces our web-based solution for integrating solutions from LMD research project [29] into SAM. Chapter 5 compares two integration approaches and discusses the advantages and disadvantages of both approaches. Chapter 6 concludes the thesis with a summary of my contributions to the research and introduce to some possible future research work can be made for the work I presented in this thesis.

Chapter 2: Service Assurance Manager (SAM)

"IT faces the problem of cutting costs without affecting the productivity of business users. Ideally, enterprises will look at solutions that increase IT ops' productivity, reduce the time needed to correct problems, and eliminate wasted time: This should favour the latest and most innovative IT management software products. IT organizations will invest in management apps that effectively lead to increased staff productivity, but vendors will need to support their claims with solid proofs and testimonials." —*Forrester Research* [3]

2.1 Overview

This chapter introduces SAM as a major new solution to solve the raising IT management problem. It also comes up with detailed description about new IT monitoring technologies such as improved service modeling, prevention of service health risks, and root cause detection for service assurance SAM applied and how these technologies could help improve IT management efficiency with limited recourses.

2.2 Better Productivity with Less Effort

"*Is it possible for IT departments to achieve optimal results and greater economies of scale while lowering the cost of service delivery? Business and economic realities tell us we have to. Corporations of all sizes, across all vertical markets, are dependent upon their IT departments to efficiently deliver the services necessary to keep the business up and running to ensure the quality of those services."* [4]

IT environments including operating systems, databases, networks, and business applications have a direct impact on a company's efficiency. For seamless business operations, organizations necessitate IT management tools to help them maintain their IT resources and manage all possible defects in their IT environments. Furthermore, advanced IT management tools often allow businesses to allocate and prioritize their problem solving resources for defects more efficiently [4]. An effective analysis of operations in an IT environment allows the staff to prioritize tasks and allot contingency assets on time. With better management of IT assets, the system health can improve and the risks of breakdowns can diminish [4].

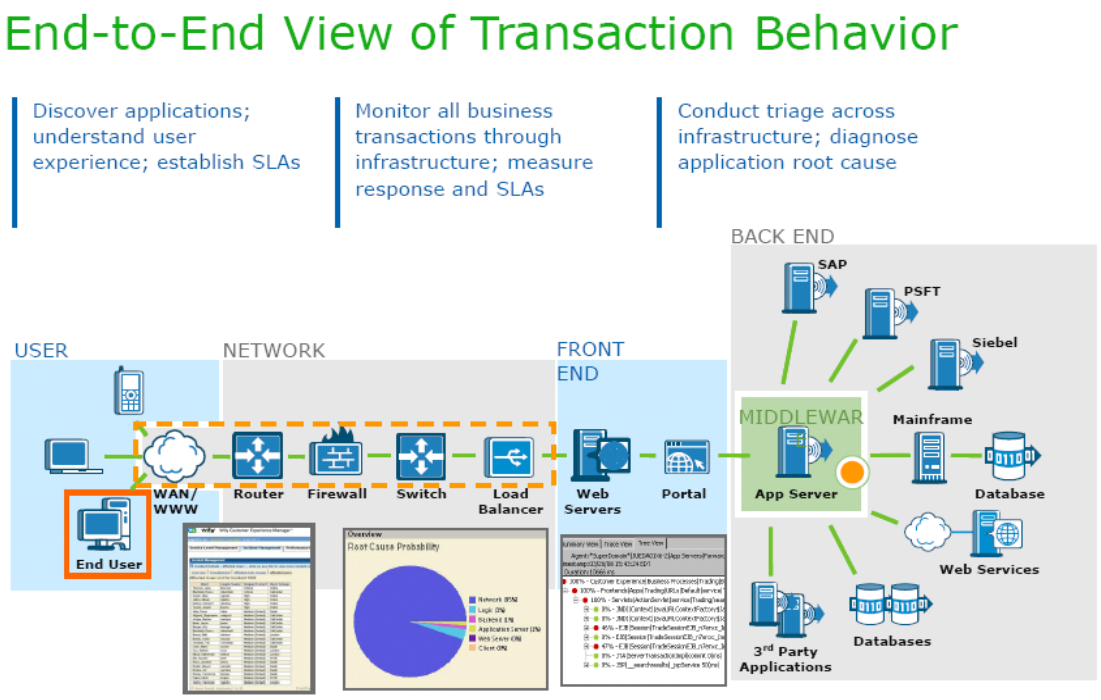


Figure 2: End-to-End View [38]

Currently, most organizations still manage their IT environments using multiple software tools rather than one universal interface. Moreover, a considerable number of IT staff is hired to

operate and work with multiple interfaces, thereby increasing the costs significantly. The staff monitors the wealth of information provided by the tools: “Recent surveys have uncovered IT’s increased desire to manage service performance, rather than mere availability. They also indicate an increased desire for predictive visibility so they can move from reactive to proactive IT management practices” [4]. Therefore, IT management requires a solution that can provide an end-to-end management view (cf. Figure 2) to fulfill the robust growth and demand. The solution should meet the following criteria:

- Expose applications under the environment;
- Enhance the user experience by providing a user friendly interface;
- Monitor every business transaction through the infrastructure;
- Conduct triage across different parts of the infrastructure; and
- Possess the ability to diagnose the root causes of defects.

For a successful and universal technological model, a strong integration technology is considered necessary to integrate all of the above mentioned modules into one solution [4]. Together with the most vital infrastructure upgrades, IT organizations are now pressing hard for new management solutions that can not only assure supreme service quality, but also allow them to condense IT staff workflow.

2.3 Challenges of IT management

As discussed in the previous section, IT managers are striving for efficient solutions within their budgets. The IT sector is often insufficiently connected to the strategic priorities of organizations because senior level executives tend to allot more resources to their core businesses assets rather

than IT assets. Thus, with limited budgets and person power, management teams struggle to keep up providing high quality services. Unfortunately, IT complexity is growing and the administration of environments requires more attention and resources. Many companies are unknowingly compromising their quality by dedicating their resources to other parts of the business. This is leading to a more serious problem in the future for the IT sector. If no new solutions are developed to restrict the impact of this problem, companies will have no choice but to adopt multiple software systems and employ more workforces to deal with the increasing number of issues and defects in their environments. To meet this challenge, IT companies have to develop new process-centric solutions. The approach should strive to deliver a user friendly interface to save end-user training costs. In the old days, a good IT solution was capable of monitoring an application's everyday life and also identifying service errors. However, this is no longer considered sufficient for businesses today. A high-quality IT management solution is also required to prevent errors to minimize business downtime. Ultimately, these looming issues lead CA Inc. to deliver their service centric IT management solution called SAM. The service-centric approach is discussed in more detail below.

2.4 Service-Centric Solution

Service-centric solutions focus on addressing common IT issues such as abnormal high load, client server is not running, and load balance running issue that are prevalent in businesses. By applying a service-centric approach, organizations can improve and adapt the IT infrastructure to different running services. The service interface allows IT staff to manage and prioritize tasks appropriately and according to quality of service requirements. Such a service-centric solution will help organizations predict and prevent IT issues in their infrastructure, thereby alleviating unexpected downtimes and loss of service. Other benefits of a service-centric approach include

higher service quality, more effective troubleshooting due to effective separation of concerns (i.e., services).

2.5 Important Aspects for Improving IT Management

Intelligent automation technologies have been used for decades to improve the efficiency of IT services. This section outlines critical technologies to facilitate quality IT service management according to industry demands. Our IT management research is based on the functionality of these technologies and we aim to reduce the complexities present in IT infrastructures while hopefully lessening the strain on the IT budget.

2.5.1 Service Modeling

The most important step to fabricate a solid IT management solution that can meet the essential quality criteria is to model the entire IT configuration of the enterprise accurately and effectively. Figure 3 shows IT infrastructures and services which consist of common elements such as *client systems, applications, databases, servers, storages,* and *networks*. CA Inc. stores the components of an entire enterprise configuration in the *Configuration Management Data Base (CMDB)* and provides tools to explore the configurations in structured and unstructured ways. Moreover, an Application Programmer Interface (API) allows users to query and access configurations programmatically.

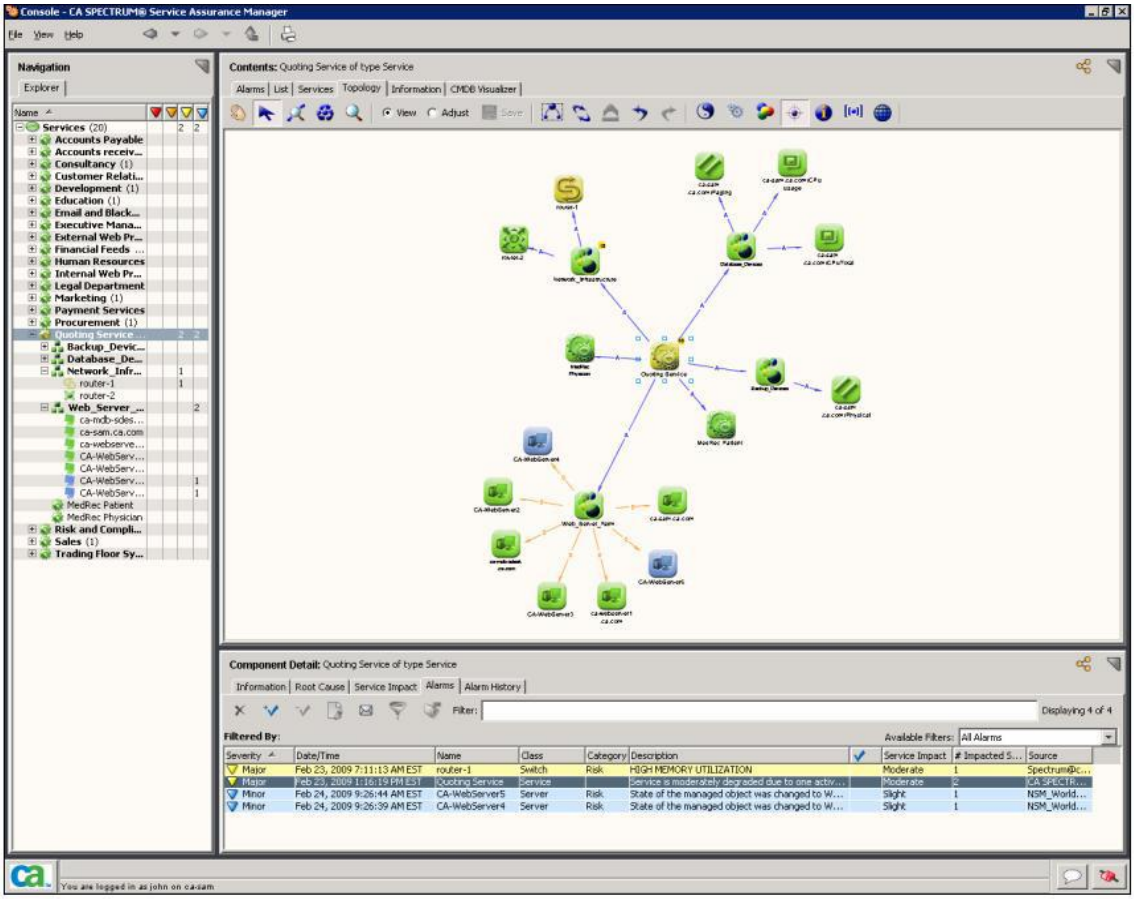


Figure 3: Model of a Sample IT Environment [38]

CMDB component are continuously updated to keep the model up to date and current. All the components are monitored in real time so they can relay the real time status of services. Thus, any removal or addition of a service is automatically managed by the solution. There is no human intervention necessary for this process and thus significantly reduces the likelihood for errors. To obtain the current status of a configuration and its components in the most effective way, the service models need to be integrated with CMDB. Domain management tools provide methods and tools to manipulate the IT infrastructure [4]. The next section discusses technologies for predicting IT service errors to reduce potential risks and damages further.

2.5.2 Prevention of Service Health Risks

New IT services need proactive management to predict and prevent any possible defects. IT services need to be robust so that "even an outage, such as a single server crash does not degrade or shut down an essential business service" [4]. However, this goal is hard to achieve with the current IT management technologies. For example, in a large IT firm, a single server outage poses no threat to the network as a whole, as the workload will switch to another server to provide high availability and prevent a service from collapsing. Nevertheless, in a small firm with only two servers, a single server shut down will be sufficiently critical to bring down the entire service. Hence, innovation is needed to assist organizations in managing different services in a unified manner. This will allow them to be more aware of risks or threatening factors to their infrastructure and give them the leverage to make critical fixes in time and before causing serious damage [4].

The new development should also provide organizations with an end-to-end visualization of the status of an IT configuration across entire service delivery paths from clients to suppliers. The goal for this new technology is to predict most service degradations so they can be recognized and addressed before they can inflict damages to local services and the user experience [4].

2.5.3 Root Cause Detection for Service Assurance

When a problem is detected in one of the configurations, it is critical to determine and comprehend the exact reasons for service degradations (e.g., identify a particular router or database causing the problem). The first priority should be to locate the root cause. Nonetheless, a good IT management tool should encompass functionality to prioritize detected problems by

severity level and present results in the form of a fishbone diagram that lists possible causes of service degradations [4].

2.5.4 New Solution Applies New Technologies

As mentioned before in this chapter, budgets for IT enterprises have been drastically reduced. Previous sections propose that, service-centric and other new technologies can reduce IT management's budgets and provide quality services to customers. By having a service assurance management tool that applies these new technologies, companies are not only able to define service impact quickly, but are also capable of prioritizing errors and address detrimental service impacts that require immediate assistance. Such a service assurance management tool can not only manage service quality, but also possible risks to service delivery. With the new tool, we can also manage our IT environment proactively which affords us the ability to predict possible service errors, and prevent our services and infrastructures from outages.

2.6 Service Assurance and Enterprise Approaches

2.6.1 Service Assurance

"Service Assurance (SA) is a procedure or set of procedures intended to optimize performance and provide management guidance in communications networks, media services and end-user applications. Service assurance is an all-encompassing paradigm that revolves around the idea that maximizing customer satisfaction inevitably maximizes the long-term profitability of an enterprise." —*Zone* [8]

Three major aspects of *Service Assurance (SA)* include *Quality Control (QC), Quality Assurance (QA),* and *Service Level Management (SLM).* Quality control emphasises testing and uncovering errors to prevent the release of unstable products. Quality assurance endeavours to improve quality of the products during the production process to eliminate possible defects. Service level management involves the management of *Quality of Service (QoS)* properties using *Key Performance Indicators (KPIs).* It also involves comparing real performance with expectations and determining proper actions according to the results.

As mentioned in previous sections, the performance and success of an IT department in a company is always measured by its relationship to revenue, person power, and business performance. To solve IT related issues, CA Inc. developed the SA solution to link user experience, transactions, and applications together to provide users with real-time information about their IT infrastructure. The major benefits accrued by SA include increasing efficiency, prioritizing actions based on business impact and *Service Level Agreements (SLAs),* and providing greater IT value and relevance to core business objectives [9]. The next section presents three different ways to implement SA.

2.6.2 Enterprise Approaches for Service Assurance

There are three major approaches to implementing SA in today's software industry. Many IT management products apply a top-down approach which executes the implementation with IT services definitions and service categories. These applications are often initiated within a Configuration Management Database (CMDB) application to track the state of the IT environment for the front line of IT support [2]. Engineers and researchers often apply a bottom-up approach for service management, which starts with a strong understanding of the IT

environment and predefines common threats in certain IT environments. The third approach is a middle-out approach, which is mainly based on some horizontal technologies for creating service definitions and assessing them by measurements that are outside the immediate infrastructure knowledge. Examples of this type of service management applications include passive web-based monitors that are not directly tied to the infrastructure of the network.

Although these approaches target specific circumstances, they all face the common challenge of capturing and translating relevant metrics for presenting service quality and integrity in real-time [6]. Moreover, these approaches do not keep at par with the rapidly growing IT environments because of the limited real-time monitoring services. Accordingly, these IT products are based on old approaches and their inability to evolve has deemed them unfit to operate in current IT environments. To solve these prevailing issues, CA Inc. developed CA Spectrum® Service Assurance (SAM). The next section discusses SAM's architecture and features.

2.7 SAM- The New Solution

SAM, a service-centric IT management solution, aims to address the rising complexity in IT environments by implementing new IT management aspect and applying enterprise service assurance technology. It guarantees to deliver a new level of service assurance technology that can manage both service quality and risks to service delivery using a real-time approach. SAM is also "a platform neutral solution that can always work fine regardless which platform it is on. It integrates with CA Inc. infrastructure domain management, application performance management, workload automation, and service desk tools as well as third-party management tools to leverage customers' existing investments in management tools and processes." [4] This product helps overcome the IT challenges, as discussed in the previous sections, using a service-

centric approach and allows out-of-the-box integration with other CA Inc. management solutions and third party applications.

By combining all three major development approaches, CA Inc. implemented its Spectrum® Service Assurance (SAM) solution to analyze events, faults, performance and manage information from IT domain management tools with a fresh end-user experience. It also collects transaction behaviour from application performance management tools to determine the impact of infrastructure on the delivery of service [4].

SAM initiates the monitoring of IT environments with a bottom-up approach from integrated infrastructure management products including CA eHealth Performance Manager, CA Insight DPM, CA Spectrum Infrastructure Manager, and CA NSM. It examines all transactions and infrastructure status with predefined risk threat patterns. Subsequently, SAM passes the monitored results to its top-down CMDB and the front line IT support to produce end-user support information. Furthermore, SAM also provides middle-out support through CA Wily Application Performance Management (APM), a web application for service health monitoring, to process application management data. The root cause analysis tools discussed in later chapters are to be integrated at the same level as Wily tools. The following sections discuss key services and features provided by SAM.

2.7.1 Role-based Service Dashboard and Console

SAM's Service Dashboard provides users with a real-time view of their IT environment status. From Figure 4 we can see that the interface categorizes the IT infrastructure information into *priority*, *health*, *risk*, *availability*, *quality*, and *SLA compliance*. It allows end-users to prioritize

and manage resources according to their needs. The Service Operation Console visualizes end-to-end service model descriptions to determine service impact and aid root cause analysis. Together, the dashboard and console allow SAM to monitor everything inside an IT environment including networks, databases, and applications in a real-time fashion.

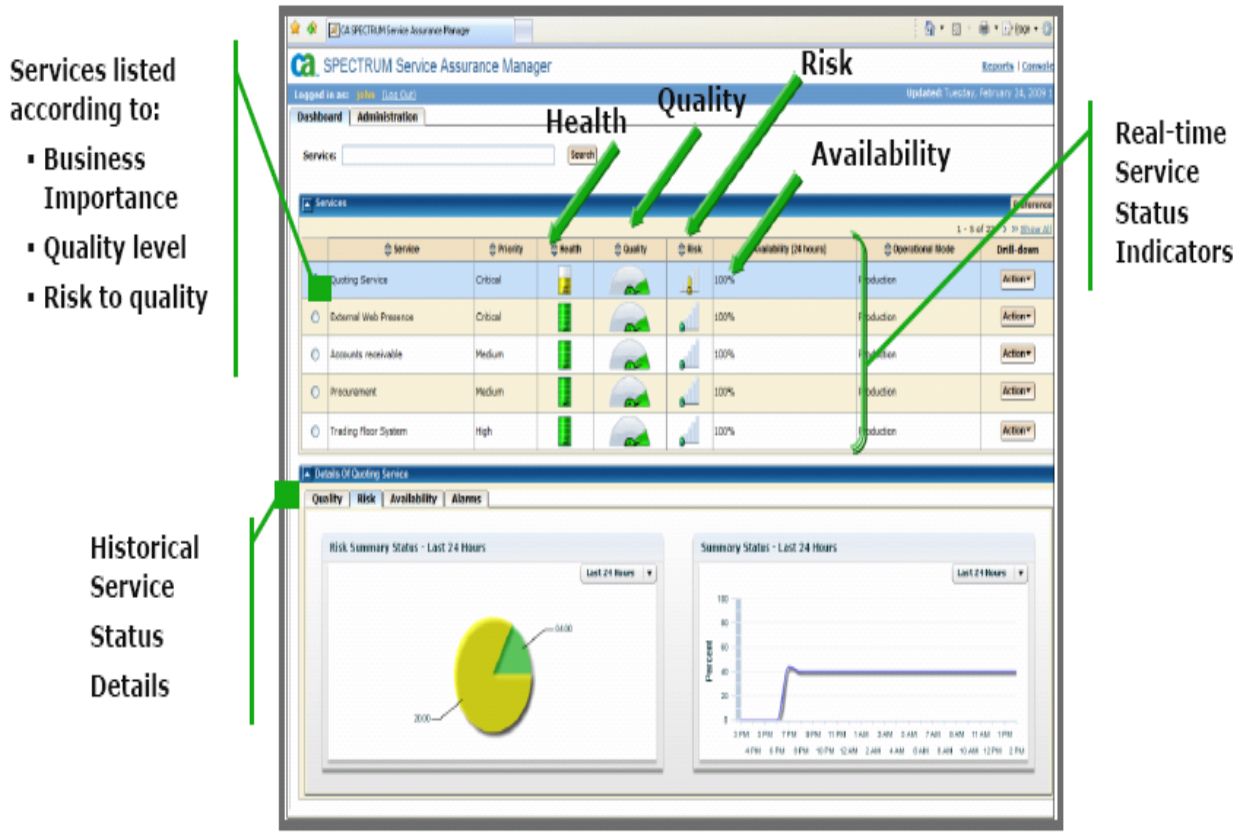


Figure 4: SAM Service Dashboard [38]

2.7.2 Visualization of Business and Service Status

SAM provides users with a visual feature that displays real-time impact of business activities on the current IT environment. From the lower section of Figure 4, one can see that SAM's Service Dashboard not only calculates and displays real-time service quality, but also integrates and displays information from business systems.

A side-by-side figure about the service status and financial impact section is also depicted in this diagram. Examples include ecommerce service quality and risk alongside the number of products sold and resulting revenue; online insurance agent service quality and risk alongside the number of new policies completed, incomplete policies, and revenue impact; online driver's license renewal service quality, number of licenses renewed, and revenue [4]. This visualization view can also include other information such as help desk tickets or calls per service.

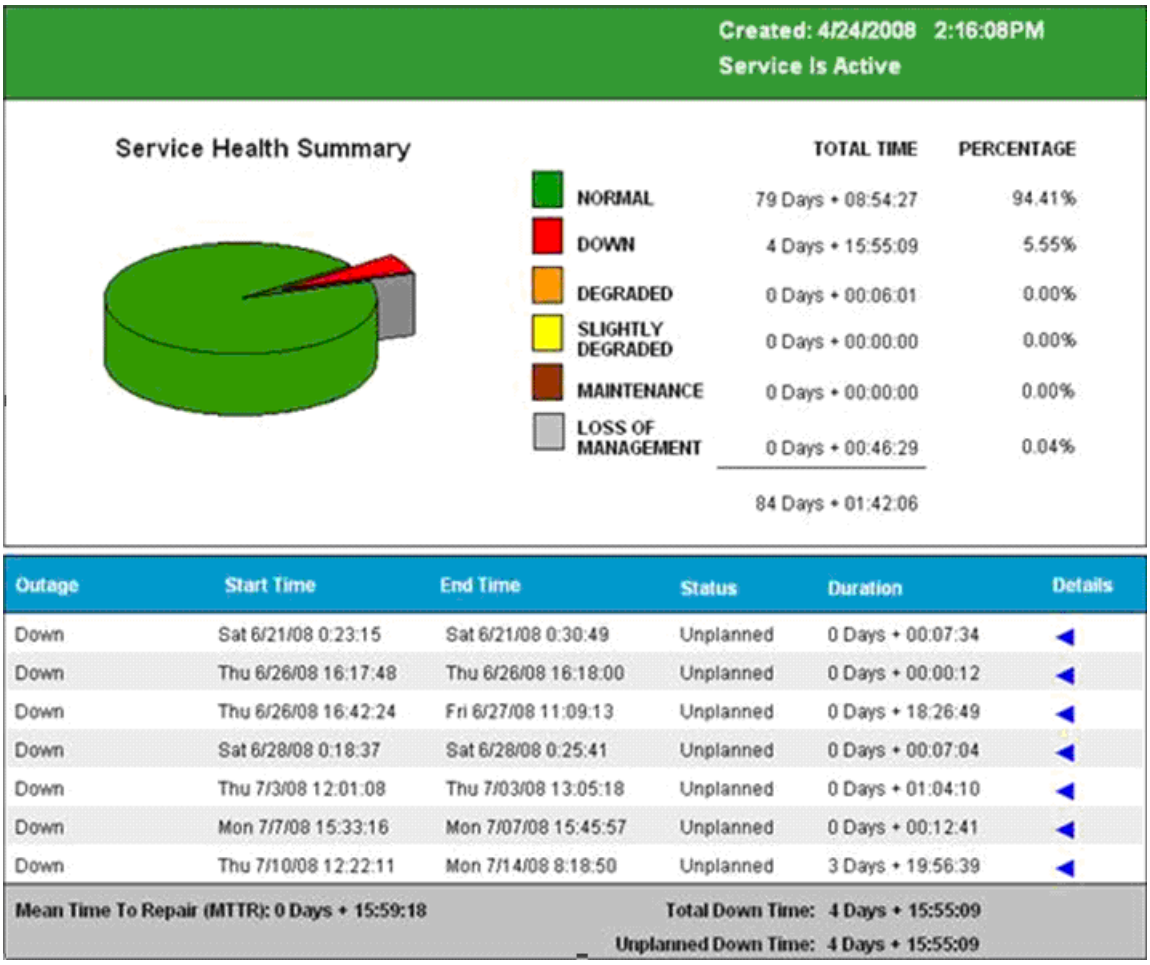


Figure 5: Risk Summary Report UI [38]

2.7.3 Service and SLA Reports

CA's Spectrum SAM is capable of pinpointing and reporting the root causes of IT service problems. This report can also be used to present the infrastructure status to business stakeholders so they can validate if the IT environment is at par with the SLAs. The report system also includes out-of-the-box contents such as SLAs, availability, health, quality of services, and service affecting configuration items (cf. Figure 5). Users can also customize the report to meet their specific business needs; however, the current report technology still requires human intervention to make many decisions during root cause analysis process. This issue lead in part to the root cause analysis research project with collaboration of University of Toronto, University of Waterloo, University of Alberta and University of Victoria.

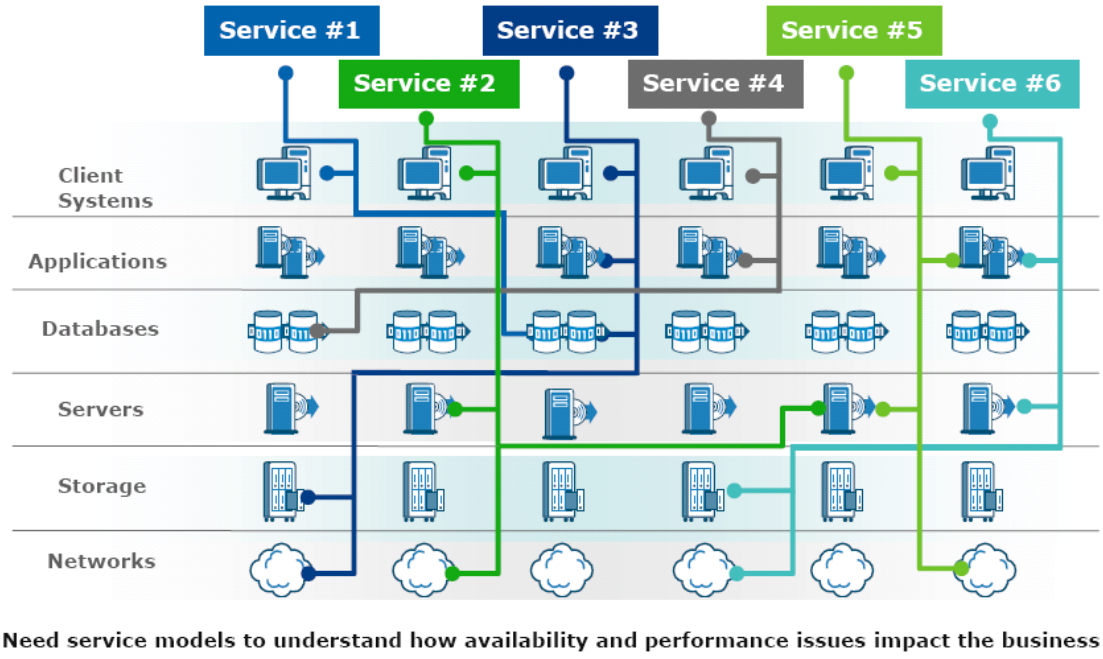


Figure 6: Services Modeling [38]

2.7.4 Intelligent Service Modeling

A model-based approach to recognize and track services is at the heart of SAM. CA Spectrum® Service Assurance (SAM) gives users the ability to drag and drop components and service constructs from different integrated CA applications, including CA eHealth Performance Manager, CA Insight DPM, CA Spectrum Infrastructure Manager, and CA NSM. To construct a new IT environment model, users can simply drag and drop available resources from Wily's Resource Family (cf. Figure 7) onto the console. Moreover, they can build relationships between different resources by defining them under SAM's Policy Editor (cf. Figure 8). Figure 6 shows an example IT model developed for a specific IT environment. Figure 9 is a more detailed view of an online ordering system's IT model.

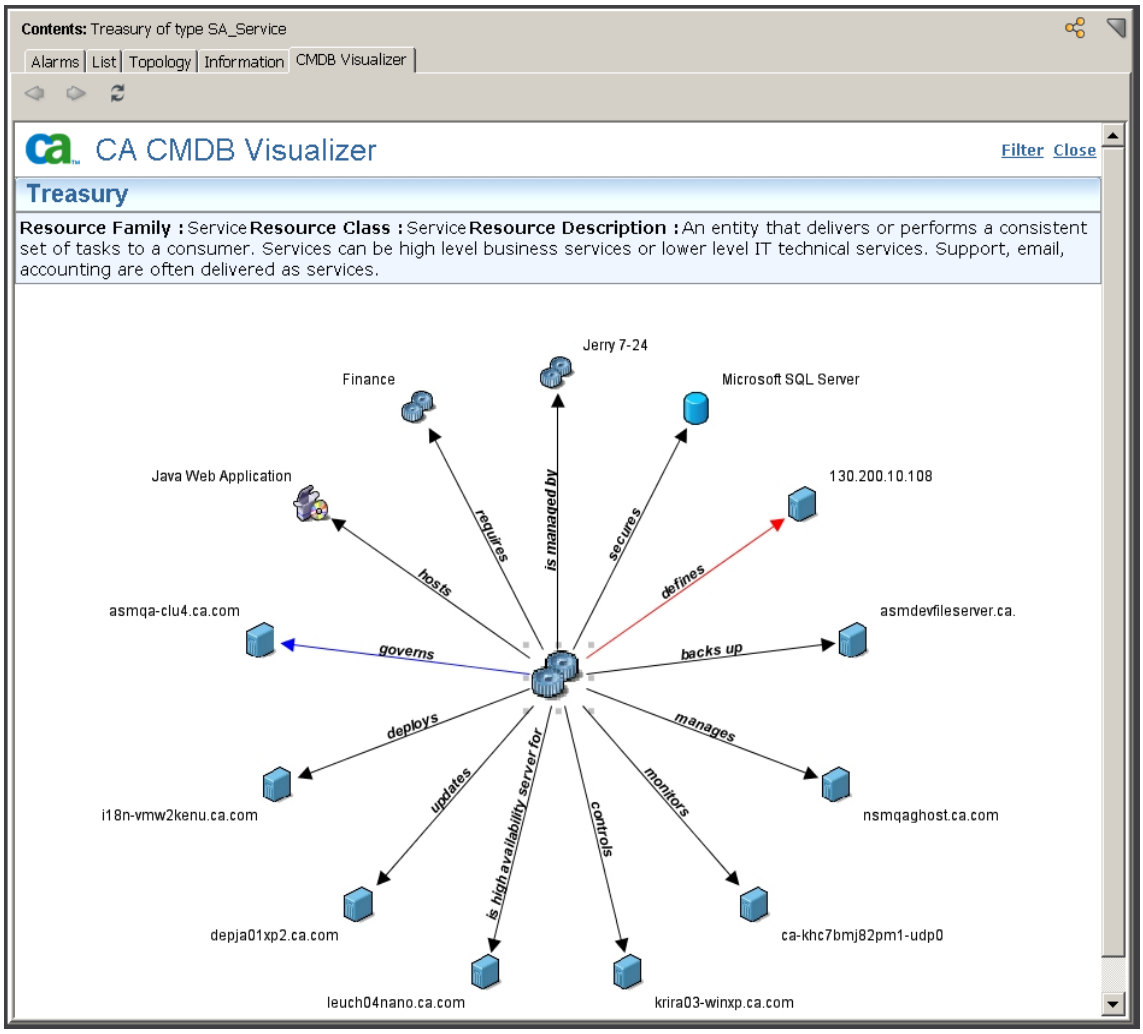


Figure 7: IT Resource Family [38]

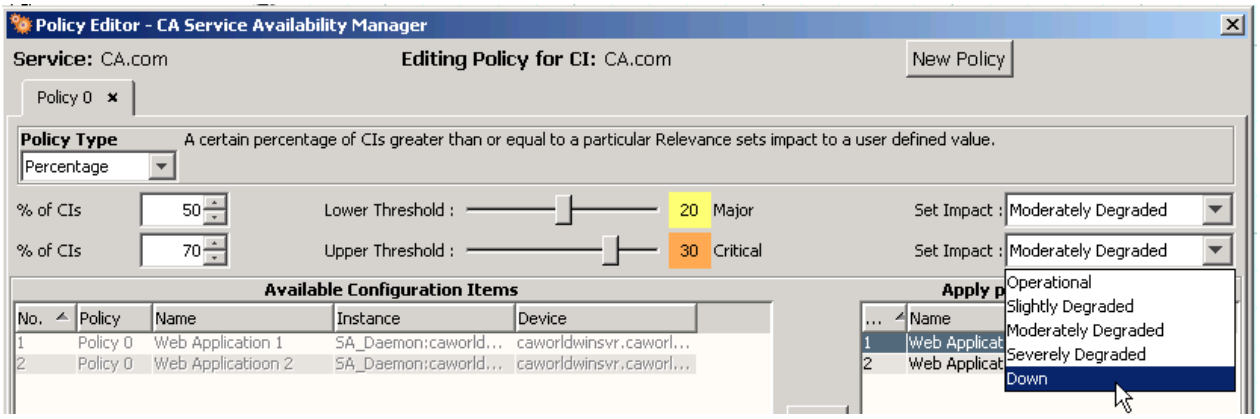


Figure 8: Policy Editor [38]

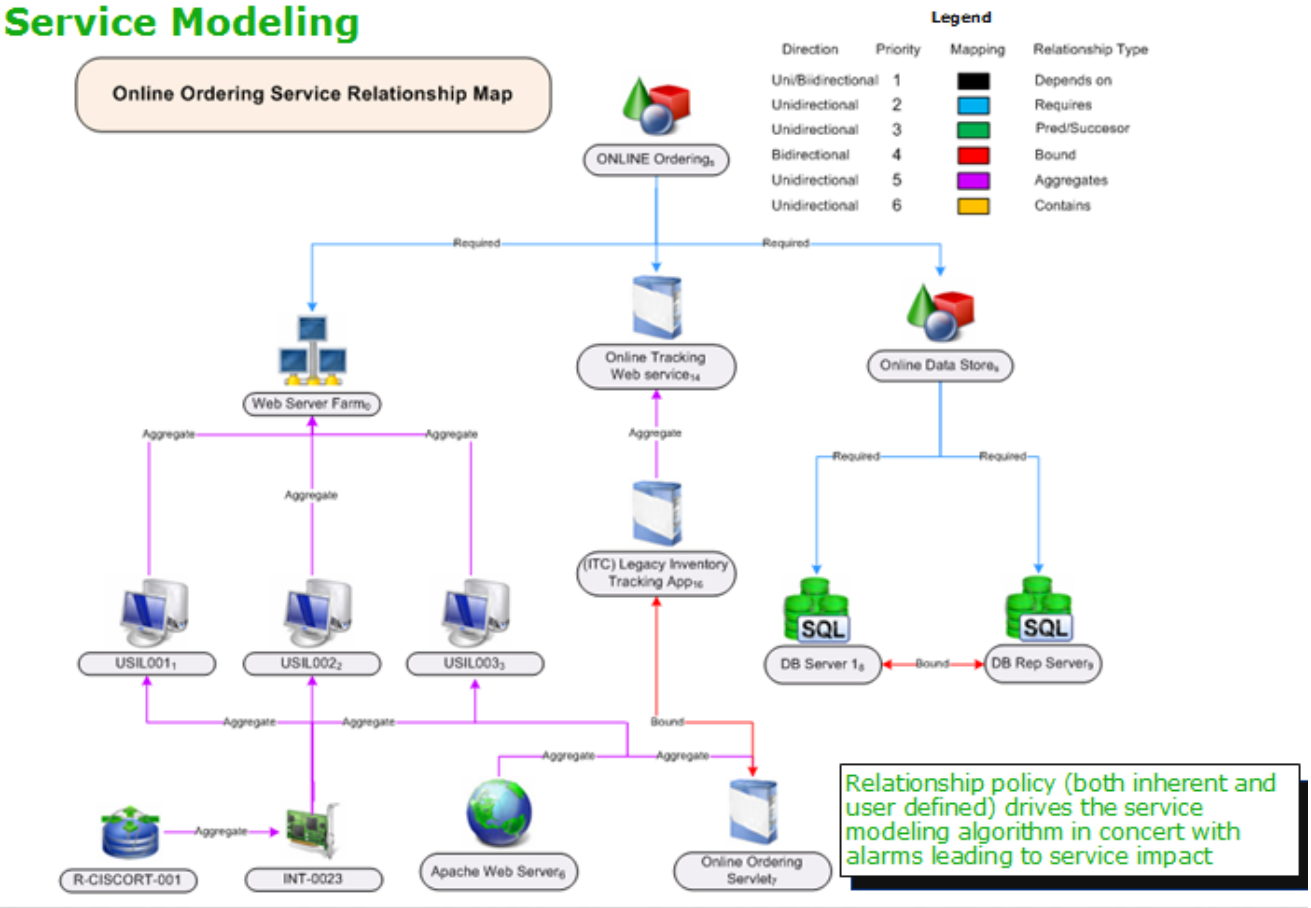


Figure 9: Sample Online Ordering Service Modeling [38]

2.7.5 Service Impact and Root Cause Analysis

SAM deploys service-centric technology so that it is capable of detecting IT components that pose a risk or have detrimental impact on the entire service quality. It can also dynamically calculate each component's impact on a particular service so they can be prioritized based on the amount of damage they can cause to services. The root causes are displayed in the console and actionable elements are provided for further analytics (cf. Figure 10).

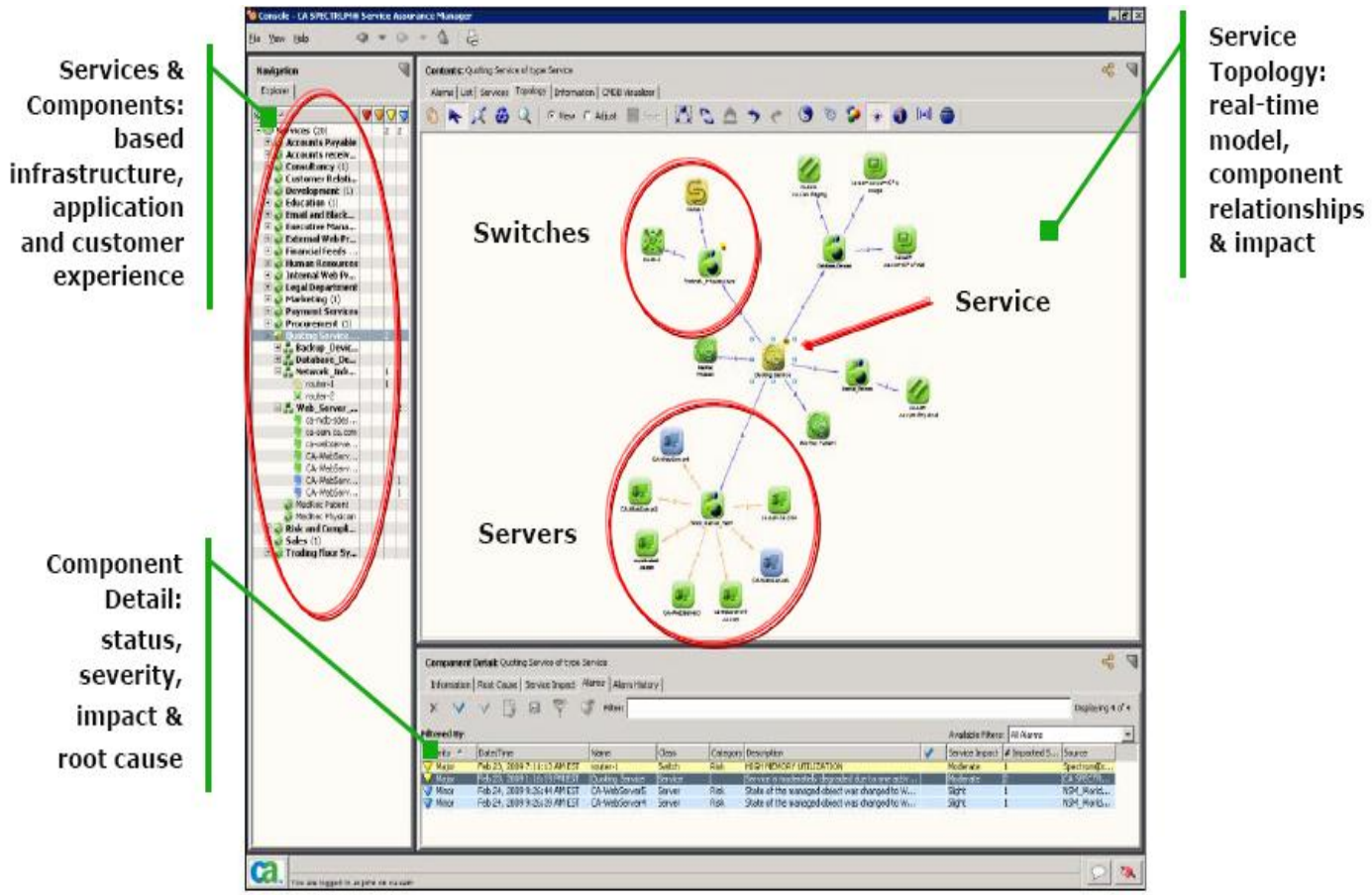


Figure 10: Analytics and Root Cause Technology [38]

2.7.6 SOA Integration Architecture

SAM connects different applications such as the performance manager, service desk, CMDB, and workload automation tools together with domain managers through intelligent connectors (cf. Figure 11). A universal connector, software development kit for data sources and an event integration tool are included in this architecture.

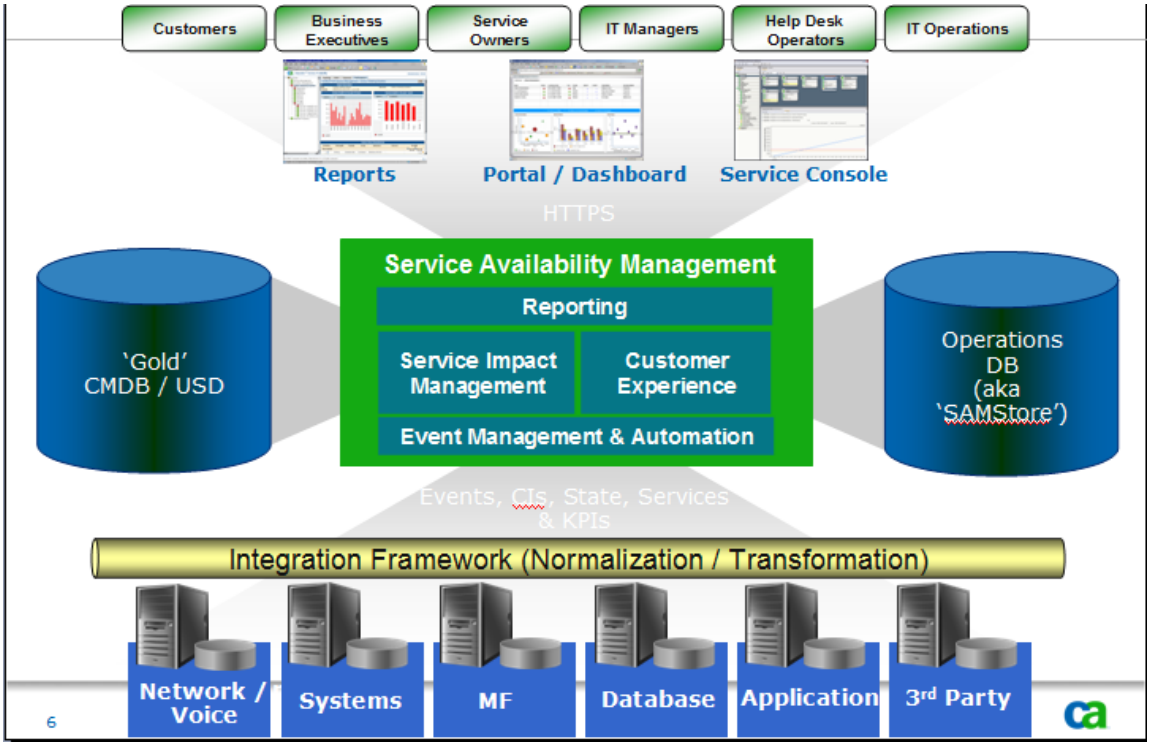


Figure 11: SAM Architecture [38]

2.8 Summary

As system downtimes can inflict significant damage to infrastructure and services, customers are often frustrated with inadequate quality. None of the management solutions in the current market can present customers with a complete, end-to-end view of their key infrastructure services. The lack of such visibility and transparency introduces further perplexity for IT staff who struggle to eliminate defects whose origin cannot be tracked.

To address these problems, CA Inc. developed Spectrum® Service Assurance (SAM) to unify the health and availability information from domain management tools in order to align it with IT services through a service-centric approach. This product introduced a new service management

layer to reuse and leverage old management solutions while also allowing users to customize the framework by extending it with new third-party applications.

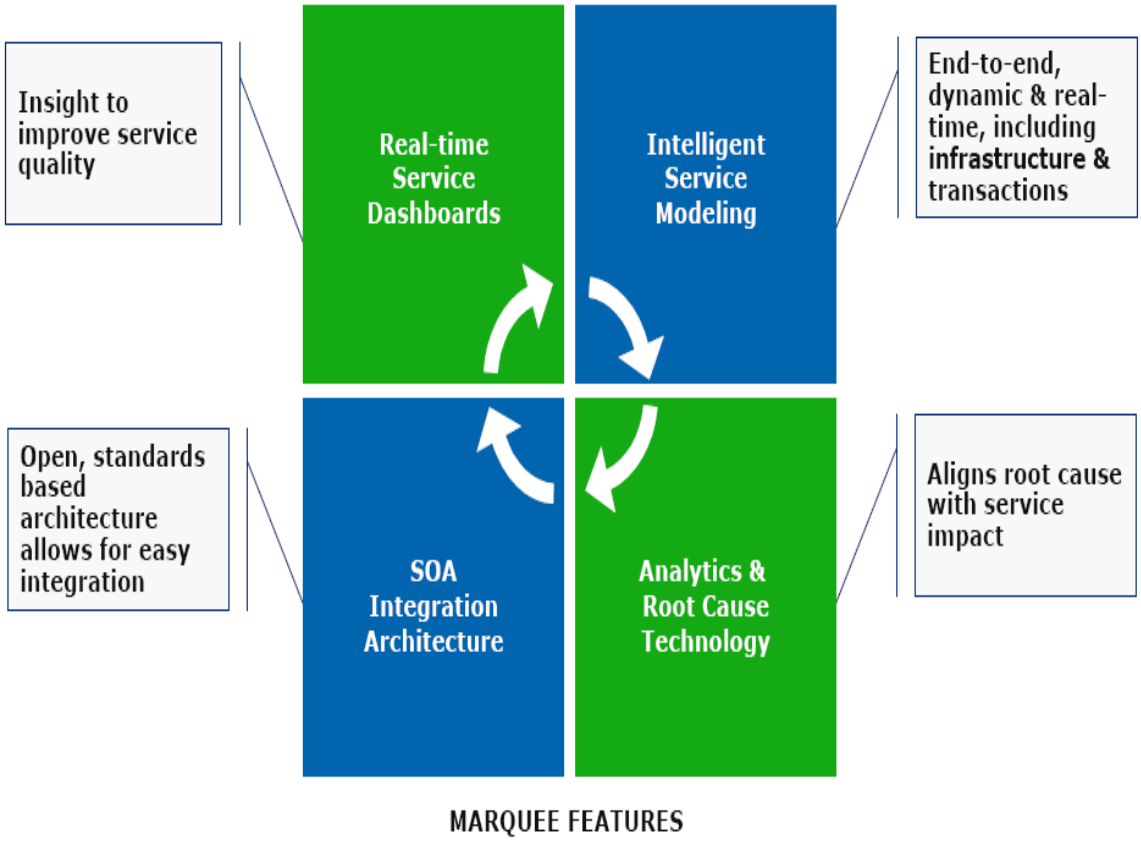


Figure 12: Benefits Brought by SAM [38]

As depicted in Figure 12, SAM increases IT environment's predictability, quality, and efficiency by providing customers with:

- An insight view of the system being monitored;
- An end-to-end dynamic and real-time view of infrastructure and transaction status; and
- An open standard based integration layer.

Although SAM offers a reporting feature to analyze root causes of defects, the current methodology requires significant human intervention to investigate and determine root causes

that triggered an error. As a result, the IT department hires more staff to deal with elements that are causing these errors. If more personnel are required for troubleshooting purposes, the financial stress on the IT department and company increases.

To take on this challenge, CA Inc. initiated the *Logging, Monitoring and Diagnosis Systems for Enterprise Software Application* research project to figure out ways to enhance SAM's root cause analysis capabilities. The next chapter discusses two integration approaches for SAM and sheds more light on the application of root-cause analysis tools.

Chapter 3 Enterprise application integration (EAI)

"The evolution of integration technologies is undergoing a major transformation with the introduction of new technology solutions as well as new methods, standards, and practices. The emergence of the Application Server Platform Suite and Web Services are poised to take the solutions focused on EAI one step closer to the dream of a complete enterprise integration model. —*LaFata* [10]

This chapter discusses the integration of third-party solutions such as root-cause detection tools into SAM while complying with EAI standards.

3.1 The Problem with SAM

As mentioned in previous chapters, SAM is a service-centric solution that monitors customers' IT environments. SAM's IT error detection capabilities are recognized in the industry and by customers. Yet, the company realizes that most of the root analysis procedure requires human intervention to find actual causes for future prevention. SAM's root cause analysis mechanisms need additional analysis capabilities to pinpoint the root causes that inflict trouble to IT assets accurately and effectively. IT management is keenly interested in determining the root causes rather than going on a troubleshooting spree. The information gleaned from SAM analyses provides a broad perspective on the situation at hand, but often with insufficient lead time to allow system administrators to predict and prevent the error from occurring in the first place. Thus, while the SAM toolkit provides great support to IT staff, they are still overwhelmed with issues while they are troubleshooting errors in complex IT environments.

To aid the effort for enhancing SAM's root casuse analysis ability, the *Logging, Monitoring and Diagnosis Systems for Enterprise Software Application (LMD)* research project aims to achieve a systematic solution that analyses error information generated by SAM in order to return the root cause(s) of these errors.

3.2 Root Cause Detecting Tools and a New Problem

Researchers from the Universities of Alberta, Toronto, Victoria, and Waterloo initiated a project to develop solutions to build on and enhance SAM's root cause detection capabilities. In the process, they explored different technologies to scrutinize and sift through SAM's log trace files and compute the possible root causes for detected errors. However, when the researchers designed, implemented, and tested their algorithms, they worked with sample trace files provided by CA Inc. without working directly with the SAM environment. Upon completion of the algorithms, the researches faced the issue of integrating their algorithms and tools into the core SAM application.

3.3 Goals of EAI SAM

By definition [12], *Enterprise Application Integration (EAI)* is an integration framework consisting of a collection of technologies and services. And this framework is built to let different applications in the enterprise share data and business process unrestrictedly through the network (cf. Figure 13). With current technologies, a resource management application, a Business Intelligent application, a chain-supply management application, and other different types of applications are not likely to communicate with each other to share data and rules. Thus these applications are normally referred to as islands of automation or technology silos.

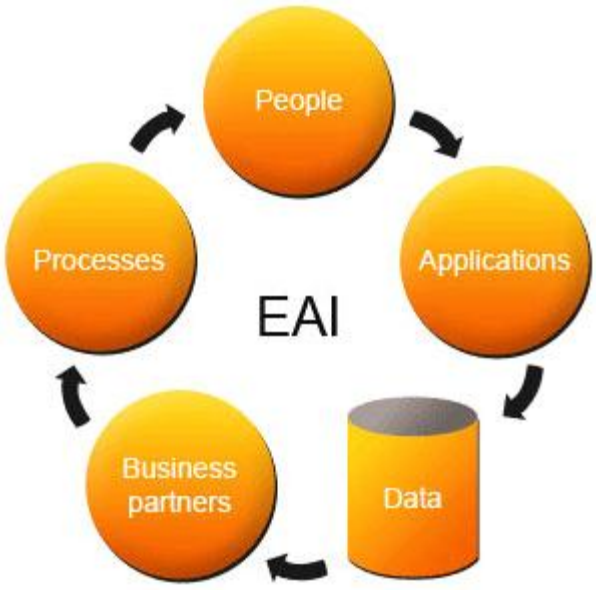


Figure 13: EAI and Applications [37]

IT cannot be managed in the form of islands of technology. A combination of different technology silos is required for a unified framework to improve IT management efficiency. The new integrated framework will afford more leverage and control to IT providers and will allow them to integrate resources developed by others.

The major objective of EAI is to integrate multiple technology silos to simplify and automate business processes, and improve the efficiency of IT management. At the same time, using EAI does not require sweeping changes to existing infrastructure to allow different applications to share data and business processes in the environment.

As mentioned in the first chapter, SAM and root cause detection tools are separate entities that operate in different locations (i.e., at different universities and CA Inc.). To solve this problem CA Inc. research staff proposed to apply EAI to build an integration middleware that not only

integrates the root-cause detection tools, but also provides an integration standard for all third party applications for SAM.

3.4 EAI Integration Patterns

EAI implements two major types of integration patterns. One is the *mediation pattern* in which EAI acts as a broker between multiple applications. Any interesting event that occurs in one application will be notified and propagated through all the applications via the EAI system. Another type is the *federation pattern* in which an EAI system could act as the overarching façade over different applications. An EAI system can act like a wrap over applications and define the relevant information and interfaces of every application to any interaction outside of the system. All the internal communications between applications appear as requesters in this pattern [12].

According to the relationship between SAM and the root-cause detection tools, both applications have already been developed and are running separately. CA researchers built an integration framework called Catalyst which applies the federation pattern to create integration wrappers outside of applications and define the communication protocols between applications. This approach allowed third party applications to be easily integrated with SAM and the root cause detection tools as well as to share business data and rules among the applications.

3.5 Quality EAI Integrations

The major goal of enterprise integration is to combine separate applications together to form a unified set of functionality. Key requirements for applications to be integrated include:

- Ability to be developed by the same company or third party vendors;

- Ability to run on multiple computers and under different operating systems; and
- Compatibility across different geographical dispersions or the ability to be operated outside of the enterprise.

The above requirements turn integration into a difficult task since it is always a challenge to come up a good and acceptable standard for different vendors out side of enterprise to implement their new solutions that will work with the existing framework without causing those developers too much extra work.

Like any other complex technological efforts, there are many important aspects that need to be considered to ensure excellent quality of software integration. Important aspects include application coupling, integration simplicity, format of data, data timelines, application asynchronicity, and data or functionality. These four criteria are discussed below.

3.5.1 Application Coupling

Before integration, it is important to calculate the proximity required between coupled applications. Integrated applications should reduce dependency on each other to avoid a service-wide failure if one of the applications experiences a breakdown. The purpose of integration is to maintain high availability, even if a fragment of the infrastructure becomes unresponsive. However, if the applications are coupled in a loose manner or if their functionality and scope is assumed, the entire integration is posed with a risk of a complete breakdown. Thusthe crucial factor, when implementing an application integration framework is to sort out the relevance among all applications in the framework to couple more relevant application while reducing the dependency between non-relevant applications.

3.5.2 Integration Simplicity

Integration simplicity refers to the level of detail in the code that is required for integrating applications. The common practice adhered to by developers is to minimize the code level impact on the existing applications, while also keeping the size of integration as small as possible. However, extra integration codes can always improve the quality of integration functions. Therefore, the approach with the least impact provides the best integration to the enterprise. Equilibrium must be maintained amid high and low impact factors according to different applications that are being integrated.

3.5.3 Format of Data

Integrated applications must recognize and adhere to a common data format or have some data translators translate different types of messages into an understandable one. By doing so, the message can be easily interpreted by other applications in the framework. Thus, for our integration framework, a commonly acceptable data format is required for data exchange among the integrated applications. While defining this standard format, it is important to consider how this data adaptation will evolve in the future.

3.5.4 Applications Asynchronicity

Application asynchronicity is different from the normal application synchronicity in which a procedure always waits while its sub-procedures are executing. In an asynchronized application, a procedure does not wait for its sub-procedures to complete; instead it logs a sub-procedure call in a log file if it is on hold and moves on to execute the remaining sub-procedures. In our integration framework, a procedure and its sub-procedures might not run on the same piece of application. The sub-procedure requires resources that the application or the service contains.

These resources might not even be available while the source procedure is waiting. Hence, in order to improve efficiency, the applications need be asynchronized, so that if a procedure has an unavailable sub-procedure, the source procedure need only log a request for the sub-procedure call and then move on to executing other related procedures. The logged sub-procedure call will eventually be executed when all required services are available [13].

3.6 Integration Styles

There are many possible approaches for integrating applications. Every approach can satisfy the integration criteria in different ways. To sum up, there are four major EAI integration styles which are considered to be popular in software integration.

3.6.1 File Transfer Approach

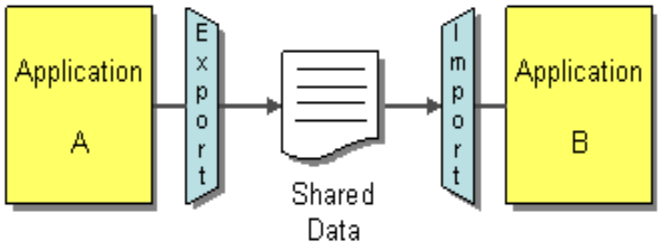


Figure 14: File Transfer Integration [13]

With the file transfer integration approach, each application in the integration framework stores shared data into a commonly shared file which is accessible by other applications inside the framework. Also, every single application inside the framework can retrieve information from other applications through the same shared data file (cf. Figure 14). Having each application store data that need be shared with other applications, the integrator becomes responsible for transforming data into different formats so it can be interpreted by applications running on

different platforms and languages. However, by choosing this approach, developers have to opt for a common file format to minimize the work load on the data interpreter. The selection need be made wisely since frequent interpretation of data can consume ample time and resources.

3.6.2 Data Sharing Approach

With the data shared integration approach, applications inside the integration framework share information by storing their business data into a single shared database (cf. Figure 15). With this approach, all integrated applications acquire data from the same database to eliminate data consistency problems.

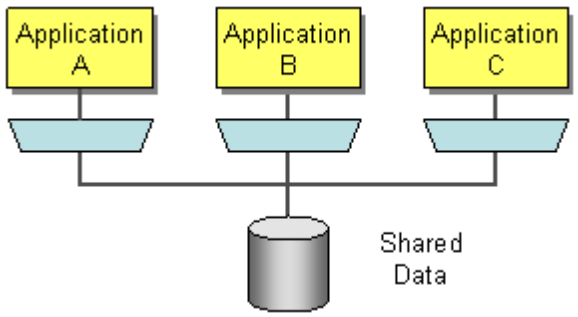


Figure 15: Shared Database Approach [13]

This approach is less efficient for integrating applications that undergo rapid changes and upgrades as that would require more work to modify the database and keep the edited database in-sync with the evolving applications.

3.6.3 Remote Procedure Invocation (RMI) Approach

The RMI approach allows every application, involved in the integration framework, to expose some of its procedures to be invoked remotely by other applications (cf.) in the same

framework. Therefore, applications can exchange data with each other by invoking these remote procedures.

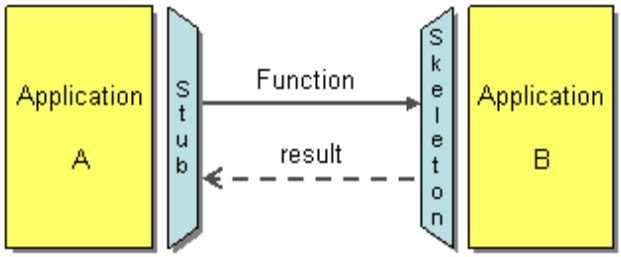


Figure 16: RMI Approach [13]

By encapsulating functions and procedures on every application, the RMI approach lets applications communicate with each other directly. The advantage to this procedure is that an upgrade or change in one of the applications does not affect others.

3.6.4 Messaging Approach

The Messaging approach connects applications with a message bus, a focal point for communication, to foster communication between applications by sending and receiving messages (cf. Figure 17). This approach can integrate as many applications as needed; also, it improves on the availability problem of the RMI approach by delivering messages even if the destination service if offline. If the application is not available during the time of message transfer, the message bus will store the message and dispatch it when the application or service becomes available. A standard message format is critical to make this approach more efficient. Although, this approach requires the development of a message bus, the cost of it is less compared to rebuilding a database.

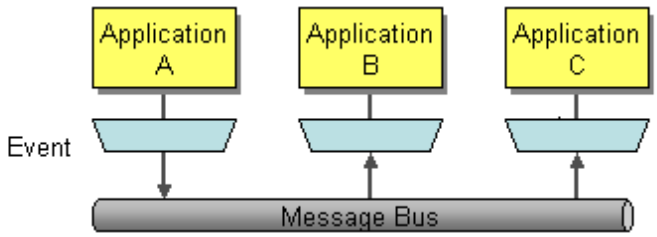


Figure 17: Messaging Approach [13]

3.6.5 Combine Different Approaches

Each of the aforementioned approaches has their own advantages and disadvantages. From our research, we decided to combine the shared data approach along with the messaging approach for our third-party application integration project. We choose this combination because with minimal new implementation (i.e., for the message bus), the message integration approach can provide us with a more effective integration framework compared to the RMI integration

approaches. Besides, using a shared database to store critical information is a more robust solution than using the shared file approach. The next section briefly discusses the development of a message system in an integration framework.

3.7 Integrating Applications with Messaging

The following major components are involved when developing a platform for messaging based integration (cf. Figure 18):

- The channel to send and receive messages;
- The pipe and filter pattern to organize and chain multiple process steps together;
- The message translator to transform the messages;
- The routing mechanism to navigate messages to the final receiver; and
- The endpoint that bridges the application functionalities and the message system together.

3.7.1 Message Channels

A message channel acts as a focal communication point by connecting two applications and allowing the connected applications to exchange data between each other. Likewise, message channels operate on a request basis. In other words, when an application expects certain data from other applications, it will only scan the available channels to complete that specific request rather than communicating with the sender application directly.

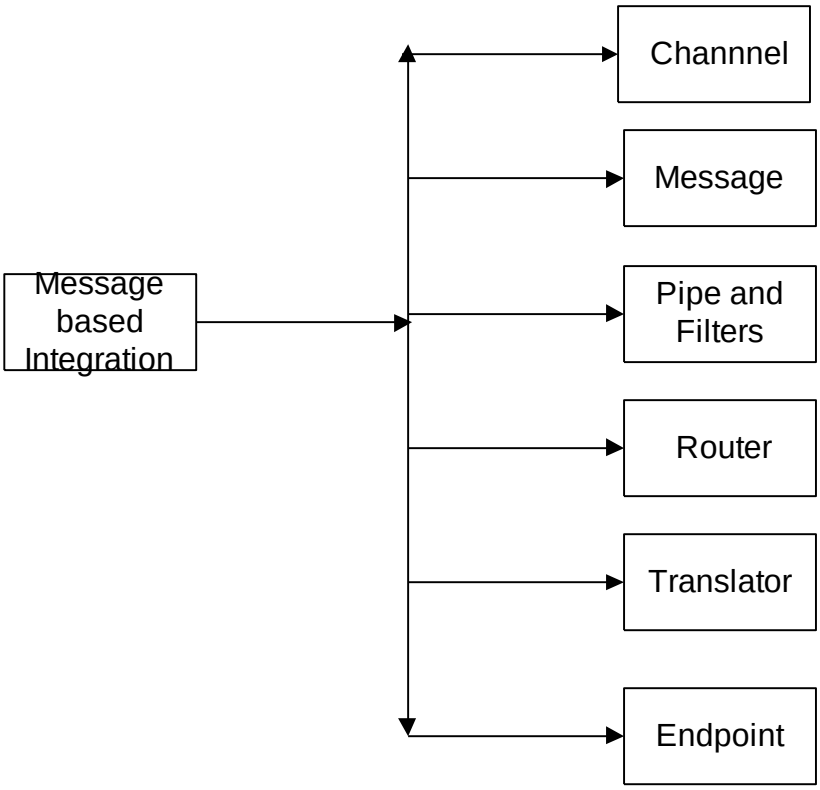


Figure 18: Architecture of Messaging Approach

The number of message channels is determined during the design phase of every integration project. In our case, every application is familiar with the application it will bond with to obtain data. Integration of additional third party solutions into the environment is also required. These new applications must first be integrated with existing message channels. When current channels fall short in providing necessary services to the new applications, the new channels will be supplemented into the integration framework to serve the newly integrated applications.

Every channel is unidirectional such that an application cannot send and receive data via the same channel. Through this approach, the case of applications receiving the data sent by them is avoided.

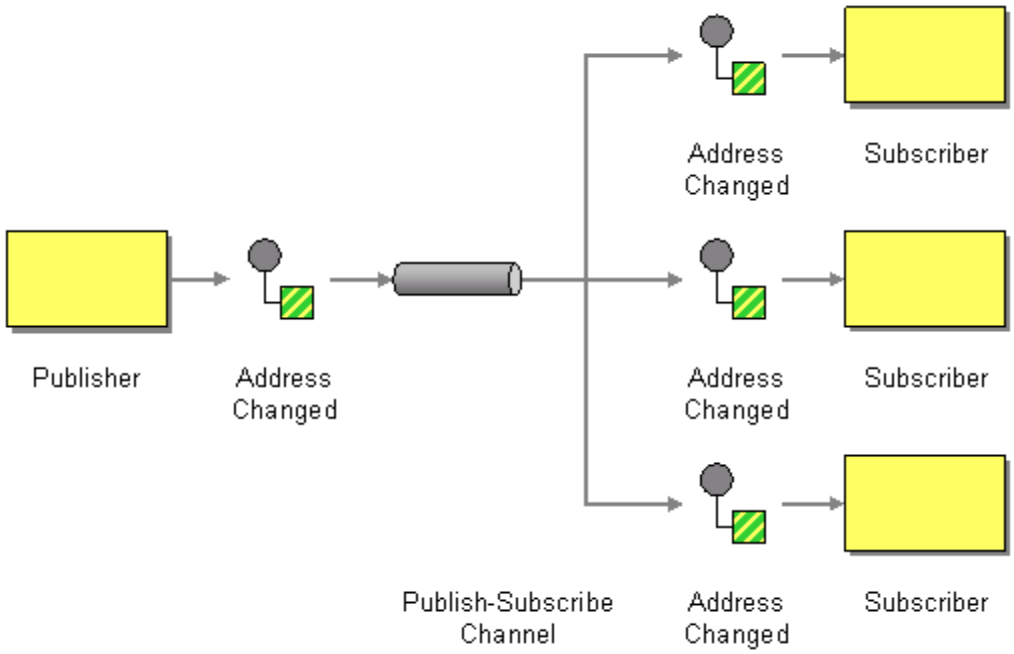


Figure 19: One to Many Channels [13]

As SAM will be integrated with many third-party applications, it is possible that certain data will be requested by numerous applications simultaneously. This is a scenario that will be evident in complex IT environments, where an application may be called upon many times. To resolve this issue, publish-subscribe channels can be used by implementing one-to-many relation between channels and receivers (cf. Figure 19). As a consequence, SAM would be able to allow many third party solutions to work on the same problem and select the best solution to satisfy user demand.

With integration of more applications, the framework gets populated with many message channels thereby turning the framework into a message bus (cf. Figure 19). The message bus stores all essential information about data access to every integrated application.

3.7.2 Message Construction

Although messages are just bundles of data, senders can differentiate them by allotting different intensions. For example, a command message can be used to invoke a function on the receiver; a document message can allow the receiver to handle data structure from the sender and an event message can notify the receiver of any state of change.

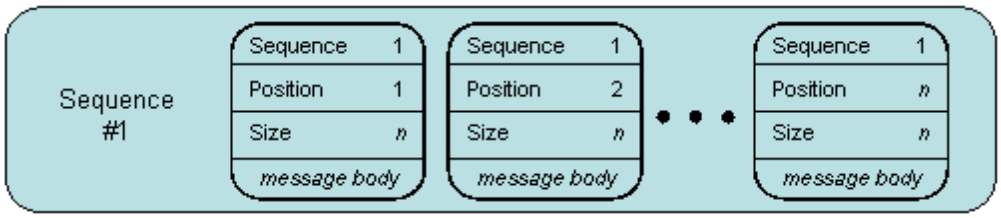


Figure 20: Message Sequence [13]

While dealing with large amounts of data, the sender should break down the message into small fragments, and deliver them as a message sequence so the receiver can re-unite the fragments for further processing (cf. Figure 20). A good message format is required for easy breakdown and re-uniting messages.

3.7.3 Pipe and Filter Architecture

For messages that require actions to be performed before they are navigated to the receiver, a pipe and filter mechanism need be deployed to connect the different actions together.

These connections are achieved through pipe and filter channels. To understand the idea of using a pipe and filter channel, consider the process of ordering a product through an online store. Once, a user places an order, the information is encrypted to prevent information theft. The user

information is then authenticated for correct information. Once this phase passes through, the information is delivered ahead for further processing.

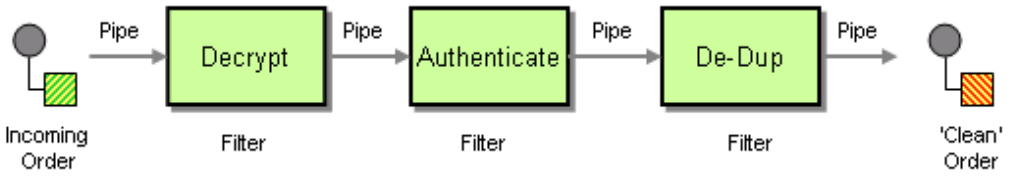


Figure 21: Sample Pipe and Filter Channel [13]

3.7.4Message Routing

In a large enterprise with multiple different integrated applications, a message might be required to pass through multiple channels before it could reach its final destination. A good routine methodology is mandatory to route messages accurately to the final destination.

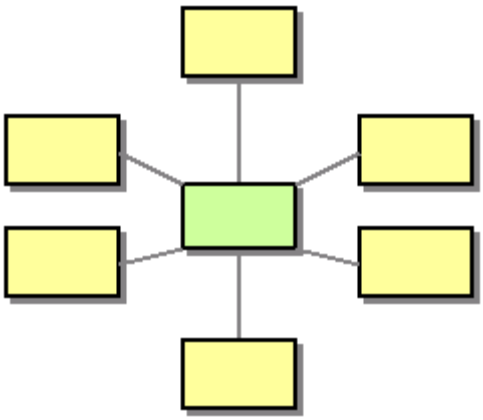


Figure 22: Message Broker [13]

There are two common routing technologies used for messaging. One is *basic routing* which represents variants of routing methods that deliver data from one sender to one or more receivers. A more complicated routing method is called *composed routing,* which takes a number of basic

routers together and forms a more complex message flow. Unlike the pipe and filter architecture, message routers allow a message to have multiple receivers.

Since, a message router will consist of many message channels leading to different receivers, it is not cost-efficient to implement such a message flow mechanism in our framework.

To reduce the amount of channels and improve efficiency of the integration framework, a more advanced routing technique known as message broker, needs to be implemented. A message broker always resides along the focal point of the application and receives messages from multiple senders. After receiving all messages, the message broker decides on the destination for every message.

3.7.5 Message Transformation

To allow new applications to be integrated into an existing integration framework, a common message format need be defined so that new applications can seamlessly integrate and access all services in the existing framework. However, applications might accept different types of messages by default, and to enhance the transformation process, the translation component intervenes at the midpoint of every message exchange. The translation component works as follows (cf. Figure 23): *Application A* attempts to send a message to *Application B*. *Application A* decides not to send the message directly to the receiver (i.e., *Application B*). Instead, it transmits the data to a translator which transforms the incoming data into the specific data format desired by *Application B.*

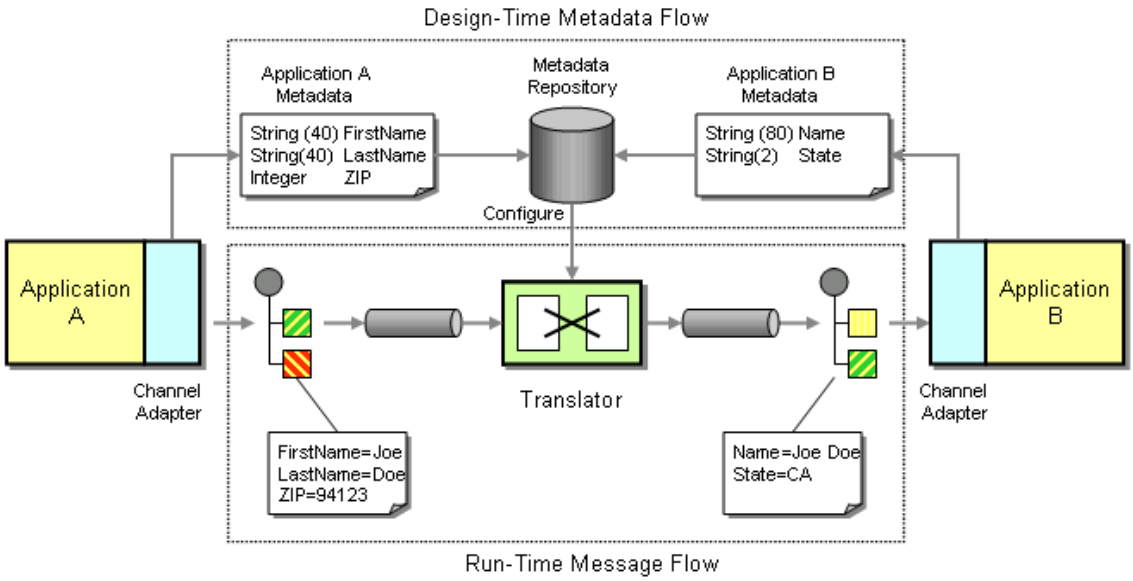


Figure 23: Messaging Transform [13]

3.7.6 Message Endpoints

The last important aspect of message based integration is the message endpoints. These endpoints are interfaces between integrated applications and the message system. They encapsulate applications and make their function accessible by the message system. Thus, endpoints function like APIs for integrated applications and these APIs compose functions of applications that are available via the message system.

3.7.7 Conclusion

This section introduced the most important components of developing a message-based integration framework. CA Inc. implemented these integration components and built Catalyst to allow integration of all third party solutions with SAM. The following section presents some important aspects of Catalyst and the shared database of the integration framework.

3.8 Catalyst

CA Inc. built the integration framework Catalyst to integrate third-party applications such as the root cause detection tools into SAM. The primary goal of Catalyst is to integrate CA applications to work in conjunction with third party applications to maximize efficiency. As discussed in the previous section, Catalyst applies a message based enterprise application integration standard to integrate different applications [14]. In other words, Catalyst achieves the goal of connecting different applications with channels and messages (cf. Figure 24).

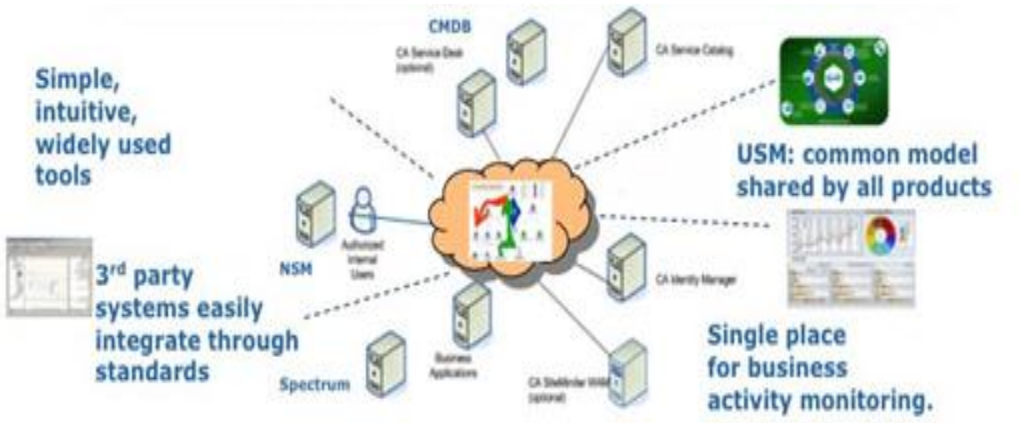


Figure 24: Catalyst and Integration [14]

Catalyst applies industry standards for web services including WS-Eventing and WS-BPEL. Through these standards, third party applications can easily query SAM's functions with the messaging system and vice versa. The web standards also make it easier for message systems to communicate with all applications involved in the integration framework. Catalyst can be

installed on top of any standard SOA platform and is capable of integrating and automating IT processes with various web services.

Furthermore, Unified Service Model (USM), a newly developed technology, acts as the key enabler of Catalyst and serves as an information model by recording all key elements that might impact to the integration framework. Details about USM are covered in the following section.

3.9 USM and CMDB Form a Shared Database

Catalyst is built with compliance to Enterprise IT Management (EITM) standard, and is a software integration approach offered by CA. EITM spans nearly every area of today's IT area including IT governance, business management, and service security.

Unified Service Model (USM) is one of the major pieces of CA's technologies for delivering EITM. It is a service-centric information model that collects information from different domain managers and provides businesses with a complete view of their IT environments. This view, not only contains technical information, but non-technical information that may provide valuable insight into technical operations. Service Definitions, Federation APIs, and Key Indicators are three major components of USM. Service Definitions are the Configuration Items (CIs) that support a given service; the inter-relationships between different CIs and service attributes are also included in the Service Definitions. Federation APIs are the interacting points for third party solutions that exchange service data with a database system known as CMDB. In this case, CMDB acts like a shared database for the entire integrated system. Key Indicators have the ability to extend the shared data model to accurately measure the performance of services.

One of the major differences between Unified Service Model and other typical data models includes the capability of not only providing the service infrastructure, but also providing insight into non-technical IT information such as the costs of delivering a service; the service level of a given service; and the people and projects that support a service (cf. Figure 25). USM successfully depicts powerful service measurement ability that reaches beyond regular service tracking and application hardware mapping.

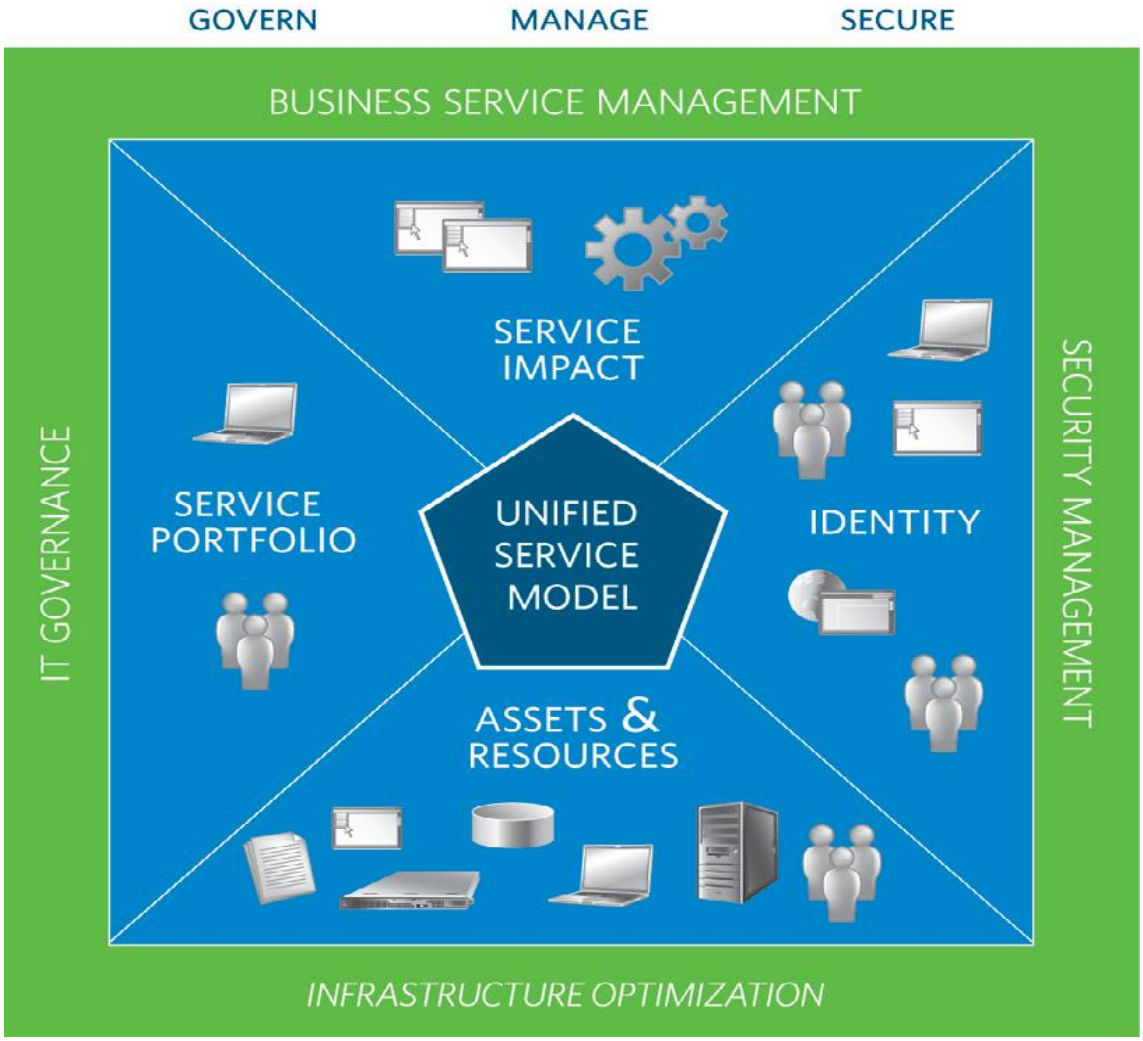


Figure 25: EITM & USM [11]

3.10 USM with Catalyst

USM provides the object model for Catalyst to achieve its objective. It also delivers a simpler, more cohesive and more effective management and control of the enterprise IT environment. Instantiated, monitored and managed by CA Catalyst, USM leverages federation services to form a logical schema referencing the SOA Registry, CMDB, and domain MDRs, for a single system of record.

Logically, Catalyst together with USM can be treated as a federated database, which implements rules and policies for mapping existing data representation to or from coherent USM representation (cf. Figure 26). It also reconciles information when IT management systems have conflicting values for shared information [14].

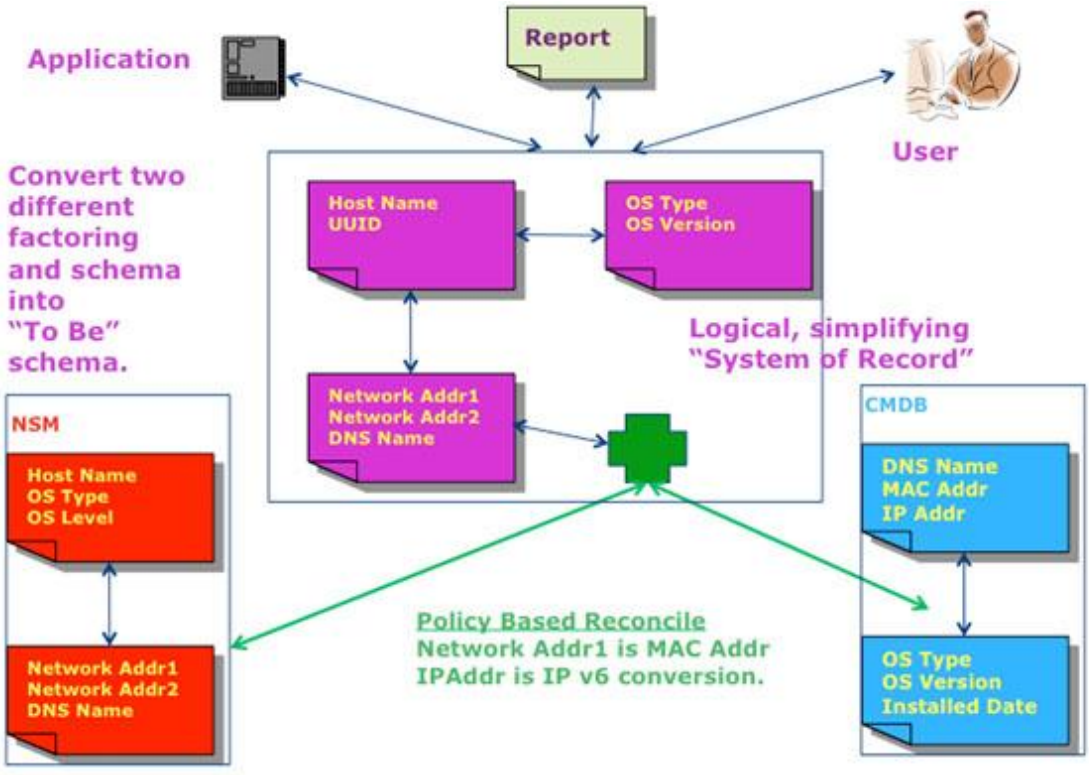


Figure 26: A Sample Mapping between USM and Existing Data Representation [14]

In addition, Catalyst/USM applies event-based integration to update information in the external IT management systems based on the rules and policies. IT administrators can see coherent information by looking at different IT management systems and they are not obliged to manually reconcile different data fragments to perform their jobs [14].

3.11 Managing the System

With functioning messaging and shared database integration systems, the remainder of the issues relates to managing and controlling the system. Developing such a complicated integration system is not a straightforward task, and it might be even more challenging to operate and maintain this coordination. Although loosely coupled applications reduce the workload to integrate different applications with each other, the approach could complicate efforts to test and perform system maintenance. Martin Fowler commented on loosely coupled applications with the following words: “Architect's dream, Developer's nightmare” [15].

Loosely coupled systems can increase the difficulty of testing a message if the sender is unaware of the message's destination and creation time.

It is essential to monitor and control message flows in the integration framework to avoid any mishaps that occur during the message delivery process. Figure 27 shows that the control bus mechanism can be implemented to monitor the input and output of applications. It can even control command messages to alter some operations if it detects any obstacle in the message delivery process.

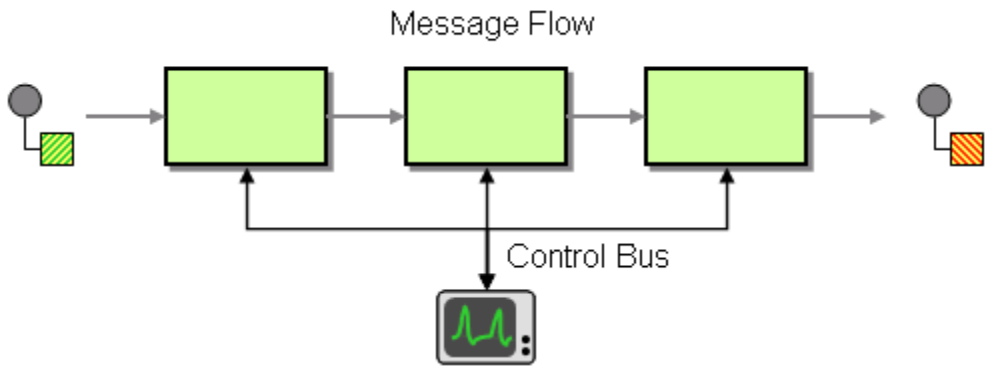


Figure 27: Message Bus [13]

3.12 Summary

This chapter presents an enterprise integration technology--Catalyst for SAM and Root Cause Diagnosis Tools. It starts from building a simple message channel for messages passing and receiving. Then it covers how to construct message format so that receivers and senders can understand each other. Later on, it shows how to build pipe and filters forward the message to the desired receiver. And it talks about setting up the routine for passing the message back and force. At the end, it introduces how to translate message into different formats and set up message endpoints so that messages can communicate with applications through these interface end points.

Chapter 4 Web-based Plug-in Solution to integrate Root Cause Tools with SAM

4.1 Message Based Point-to-Point Integration

As mentioned in Chapter 1 four Canadian universities in collaboration with CA Inc. undertook the *Logging, Monitoring and Diagnosis Systems for Enterprise Software Application (LMD)* research project to investigate root cause analysis and diagnosis methods, techniques and tools [29]. Every university pursued its own approach to develop root cause analysis and diagnosis techniques. This chapter discusses how we integrated these techniques into SAM given the different integration mechanisms outlined in the previous chapter.

We began with an initiative to formulate a standard data exchange format between SAM and other third party applications in order to run the entire integration process through web based services. The main idea was to allow third party application integration without affecting the core services of the framework. Since, we were also restricted with respect to time and person power, we did not pursue EAI standards for this project as outlined in Chapter 3. Compliance with EAI standards requires plenty of business design and infrastructure changes for each application that need be integrated.

Instead, we built an N-to-N integration framework to incorporate SAM and root-cause detection tools while also using message based integration concepts to obtain data to examine if the EAI

approach was feasible for software integration. The rest of this chapter discusses the message based N-to-N integration framework solution.

4.2 How does Web Based Integration Project Work with SAM

Our web based integration framework functions as a plug-in for SAM and runs as a web application that communicates with SAM and root cause detection tools.

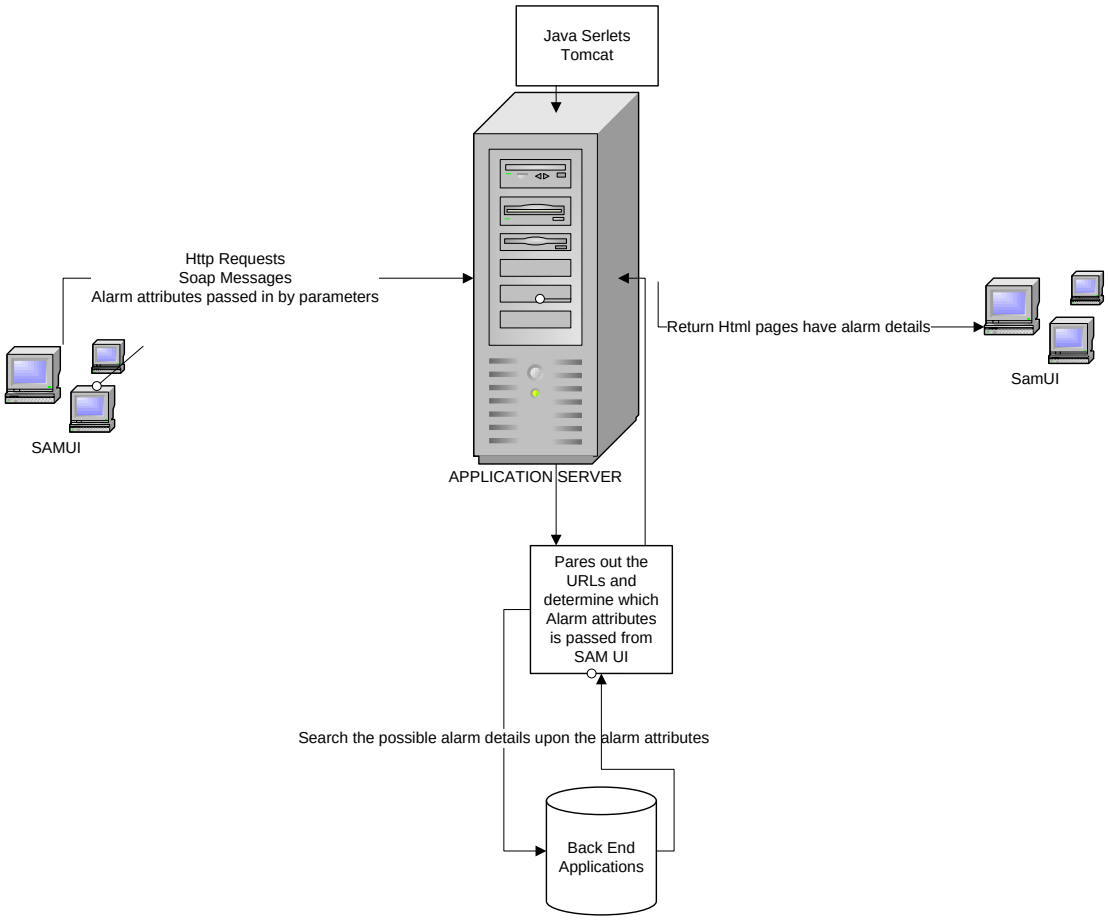


Figure 28: How Integrated Tools Communicates With SAM

The Figure 28 illustration shows a typical single phase interaction between SAM and our integration framework. Once, SAM detects an odd behaviour in the IT environment, it elevates a service alarm at its *Unified Alarm System console* (cf. Figure 29). Thus, the staff can intervene

and investigate the issue using the integration tools, which is running as a web application. The integration tool invokes the root-cause detection tools to examine the submitted details and provides a breakdown of possible root causes of the issue. The root causes will be displayed on the associated UI of our integration tool. All root-cause detection tools, developed by different universities, are deployed as web services and have the functionality to accept soap messages format querying.

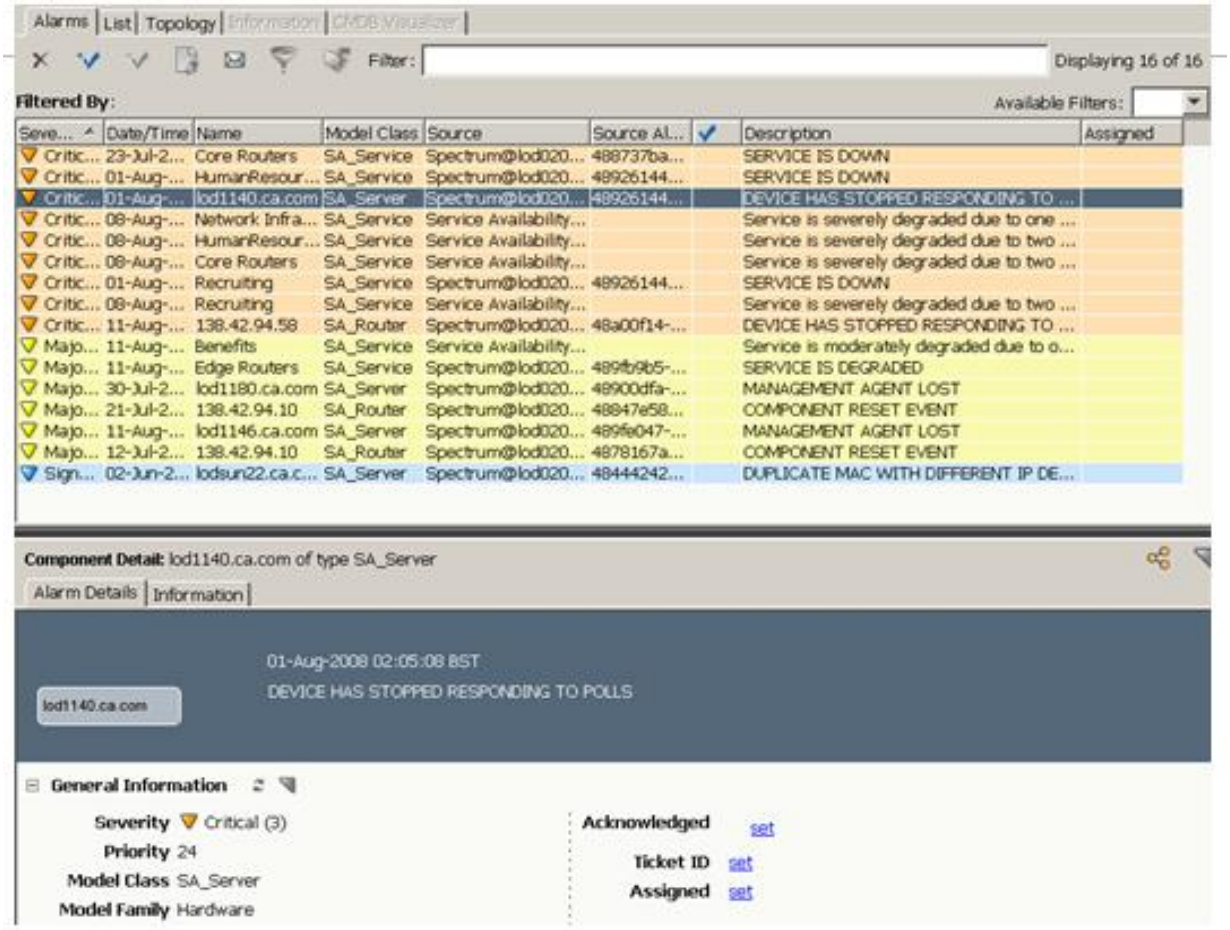


Figure 29: SAM Unified Alarm System

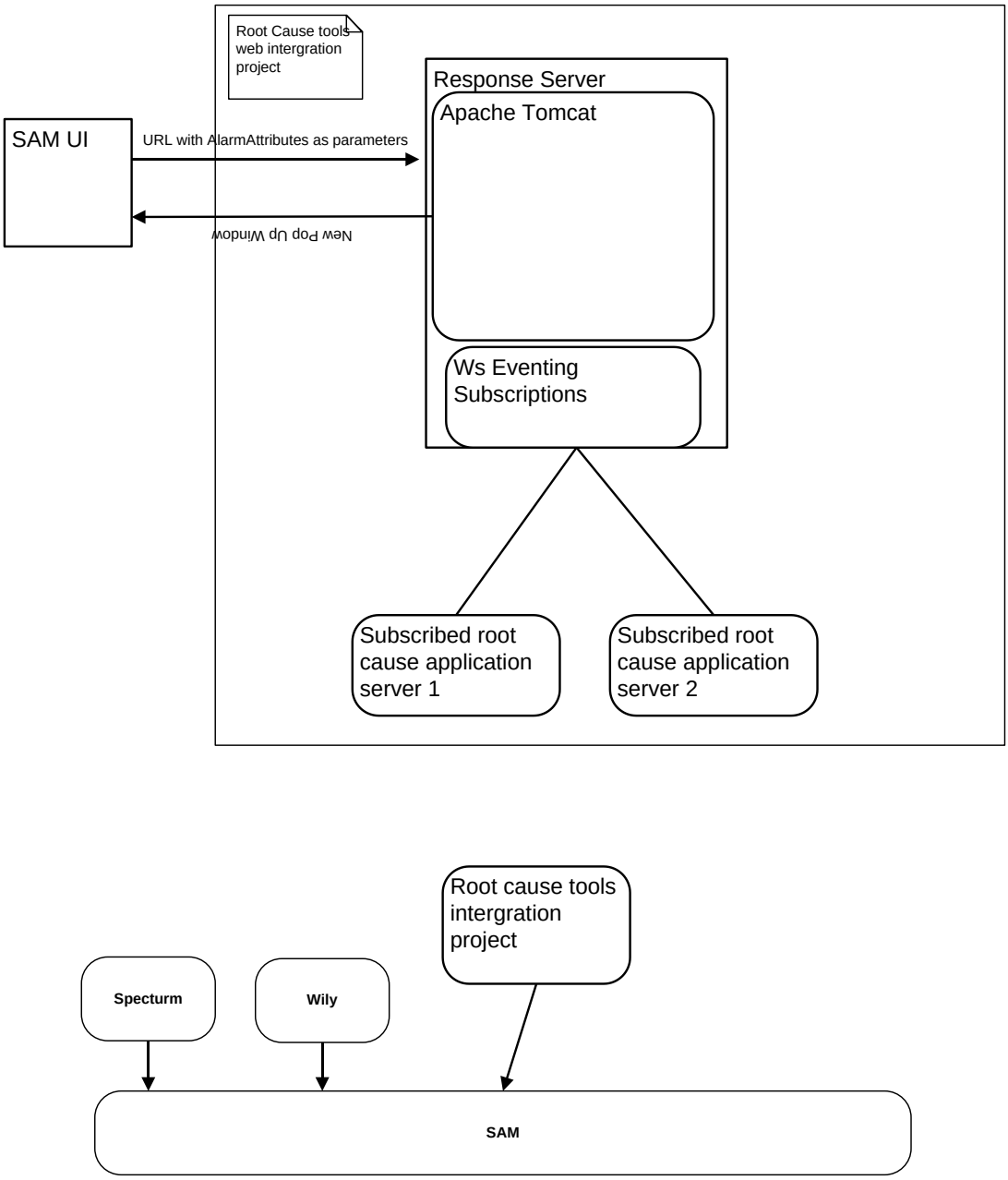


Figure 30: Architecture of SAM Integration Project

The above illustration displays the architecture of our web based integration framework. This framework functions as a message broker between SAM and the root cause detection tools. It defines a standard message format for all the communications occurring between the two

platforms so that all the IT alarm details and possible root-cause of these alarms can be communicated back and forth through the messaging system. With a unified way of data message data interpretation, data can easily be exchanged between the core SAM functionality and third party applications. In a large-scale scenario, this framework allows root-cause analysis tools to run parallel with other third party applications. The next few sections discuss our strategy to accomplish the integration project using our interface design and message system.

4.3 Customizing SAM's User Interface

In our interface, users will have the ability to investigate SAM's error alarms by passing alarm details to root-cause detection tools for a detailed and thorough analysis. The starting point for designing our interface was to extend SAM's current user interface so that users can investigate error alarms in *SAM's Alarm Dashboard*.

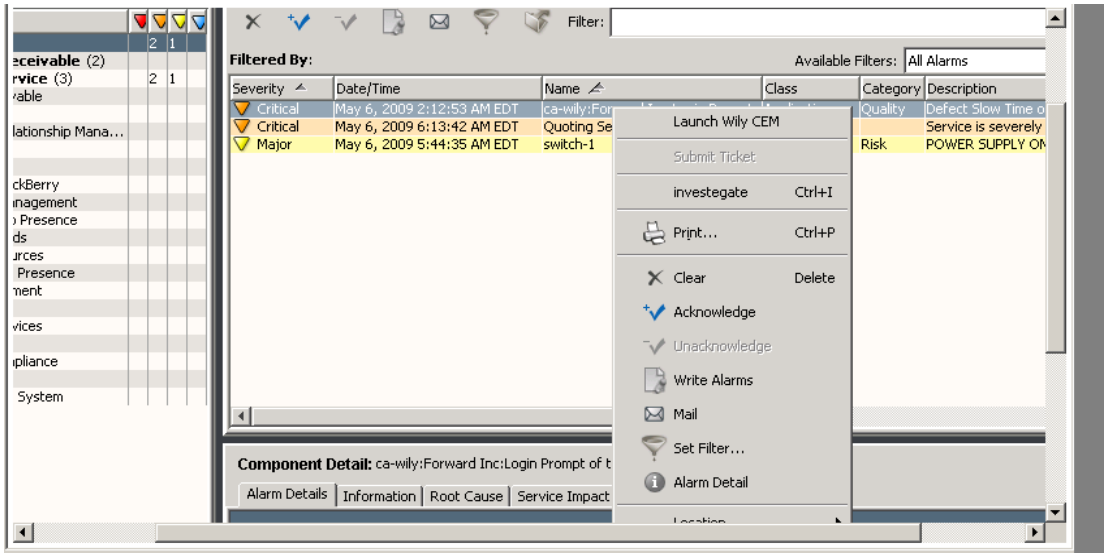


Figure 31: New Command in Menu

SAM's rich customizable user interface allowed us to achieve the user interface modification in two simple steps. The first step was to modify the pop up menu in SAM's Unified Alarm Panel to include a new operation called *investigation*. A *new investigation option* will be visible once a user right clicks on an error description from the application dashboard. The *investigation option* will trigger an action to transmit alarm details to our integration tool and redirect the user's attention to a list of all possible operations that the user can chose to execute on the particular alarm. This was accomplished by adding an extra configurable component into SAM's configuration file as follows:

```
<menu>
    <item>
        <name>investigate</name>
        <URL>http://localhost:8080/SAMUI</URL>
    </item>
</menu>
```

The second part of the solution, after obtaining possible root causes, is to choose an action to remediate the chosen error alarm. The SAM integration framework needs to work in concert with the root-cause analysis tools to perform operations listed inside its user interface. The next section presents details on the user interface of SAM's integration framework.

4.4 User Interface of the SAM Integration Tool

As mentioned in the previous chapter, once a user investigates an alarm under SAM's Alarm Dashboard, a pop-up from SAM's integration framework user interface will prompt the user with the details and available operations for the particular alarm. This user interface is to extract more

information about the IT environment monitored by SAM and identify factors such as service availability and service architecture. This will provide users with a comprehensive view and interpretation of the error along with a list of available options that can be performed for further analysis. Consequently, we need to develop a new web based user interface for the integration project and the user interface must satisfy all the aforementioned requirements.

We chose the Google GWT [16] technology to build our JavaScript based web graphical user interface (GUI). The illustration displayed in Figure 32 below shows the front page of the user interface which is displayed every time the investigation button is clicked. It parses the following pieces of information from SAM:

- the number of systems that are currently monitored and their health status;
- the service level of each system, and
- the actions that users are allowed to take according to each particular alarm.

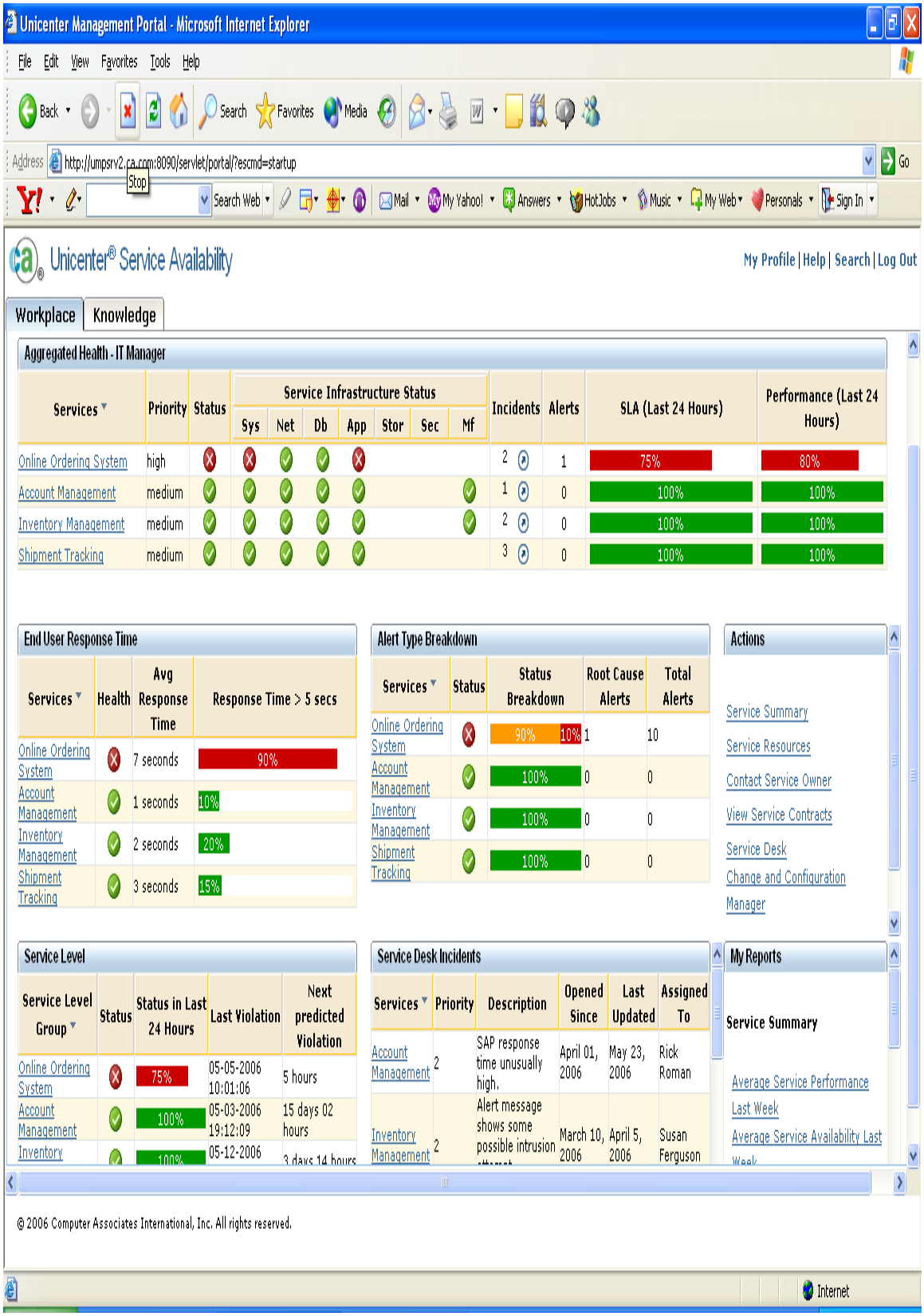


Figure 32: Front Page of the Web Integration User Interface

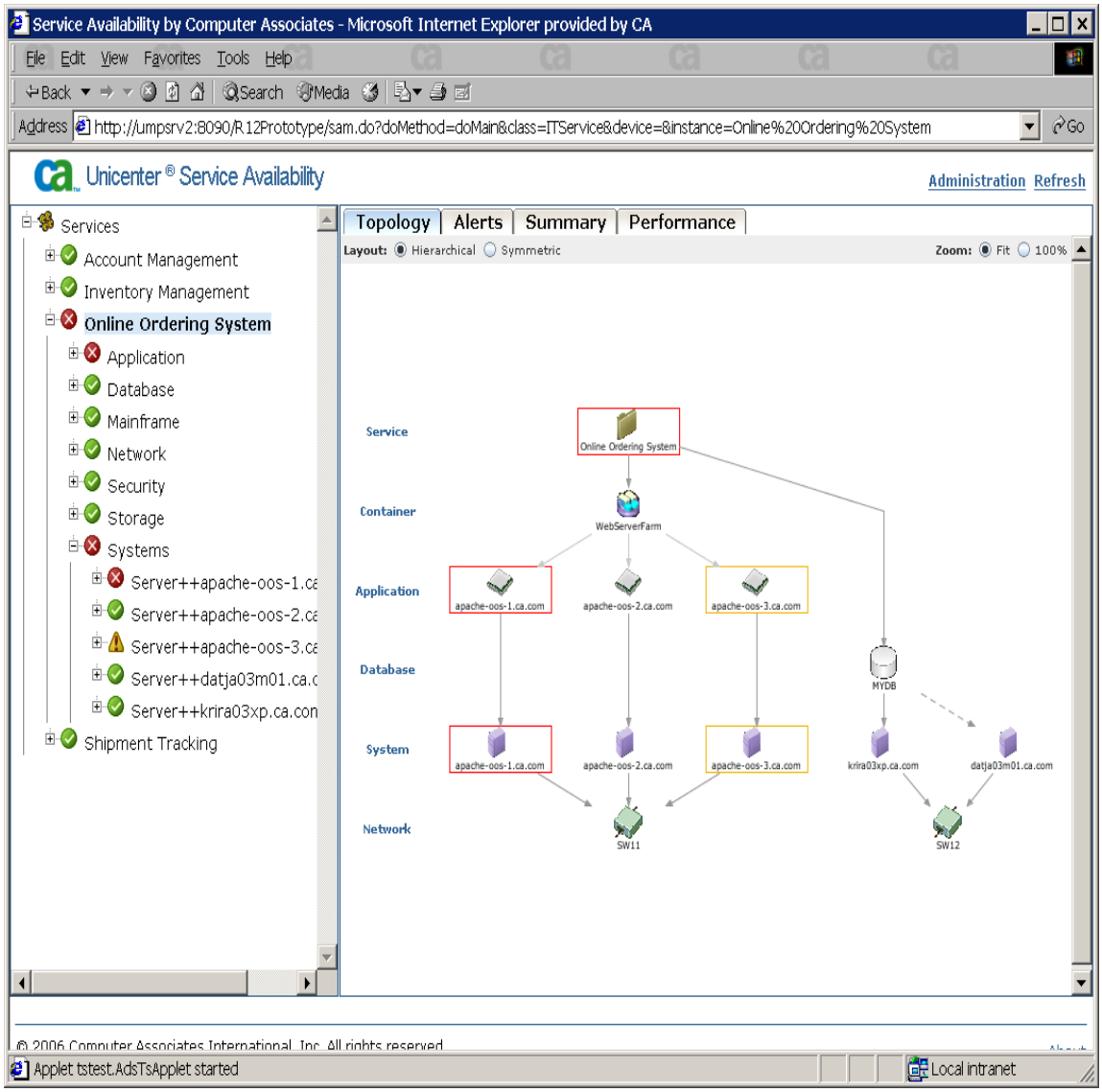


Figure 33: Service Breakdown

The user interface also requires a subpage that displays breakdown details of every monitored application in SAM along with more information about the IT environment. The Figure 34 below shows the break down structure of a sample online ordering system that includes basic system components such as *database*, *router*, and *storage*.

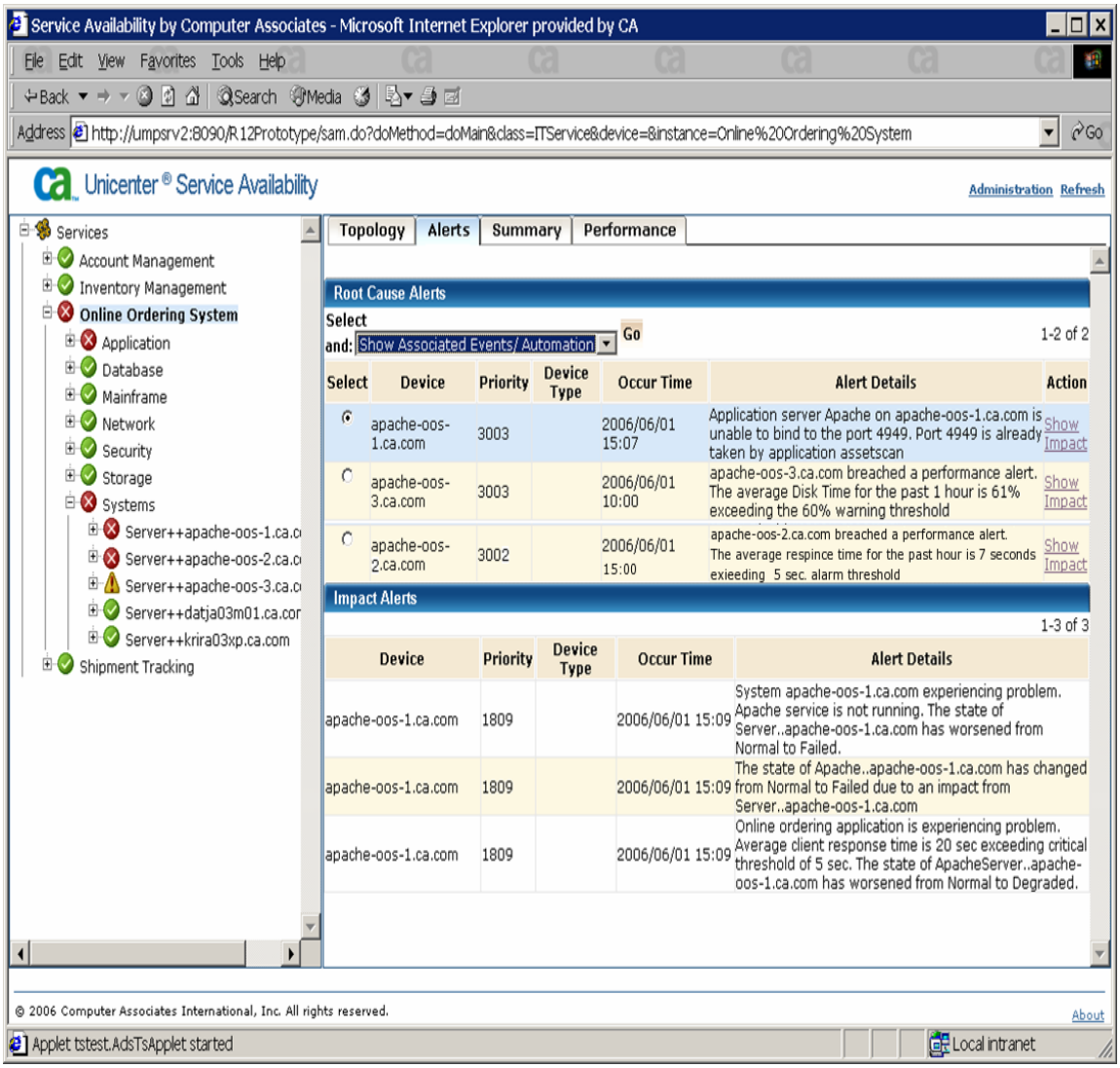


Figure 34: Alarm Diagnose Page

The user interface must have the ability to invoke the root-cause diagnosis tools to analyze SAM's alarms. As shown in the above illustration (cf. Figure 34), users have the ability to select an alarm, from the Alarm Dashboard, and trigger any of the possible listed actions, such as *show impact* and *diagnose root-cause*.

If a user decides to diagnose the root causes of a particular alarm as in the previous stage, the program will redirect the user to the alarm investigation page that will display the root causes in the form of a fishbone diagram (cf. Figure 35). As can be seen from the above example, servers that are not running and have an abnormally high load get categorized under the *High Response Time class.* Moreover, this diagram illustrates the association of one root cause to many different alarms, which concludes that a single alarm could be the major cause of other issues. A fragment of gwt code for generating fishbone diagram will be presented in Appendix C.

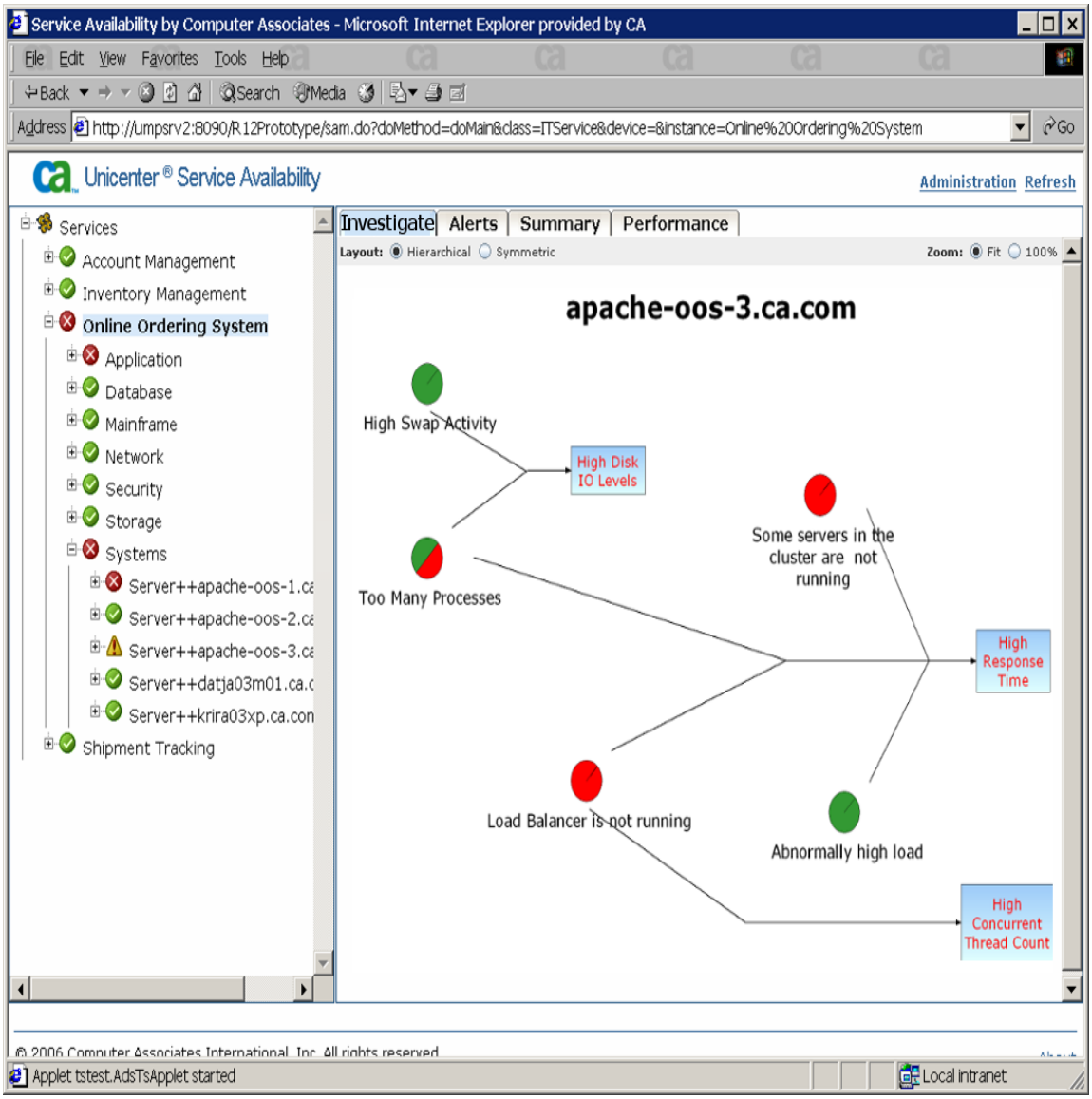


Figure 35: Fishbone Diagram for Root Causes

As the fishbone diagram need be drafted using feedback received from the root-cause diagnosis tools, the aforementioned code will need be configured to display the fishbone with respect to the data received from the diagnosis tools. We formulated a standard for diagnosis applications to relay feedback information to the main servlet and produce the following fishbone diagram:

```
<rootcause>
    <hypothesis1>
        <root cause 1></root cause 1>
        <root cause 2></root cause 2>
        <root cause 3></root cause 3>
        …
    </hypothesis1>

    <hypothesis2>
        <root cause 1></root cause 1>
        <root cause 2></root cause 2>
        <root cause 3></root cause 3>
        …
    </hypothesis2>
    …
</rootcause>
```

Appendix B contains three usage cases that demonstrate our user interface for SAM's Web Integration Project.

4.5 Design of the Service Part of the Integration Framework

According to the architecture of SAM's integration framework displayed in Figure 36, the service part of our integration framework is responsible for leveraging communication between SAM and the root-cause detection tools and thereby acts like a message broker. This section discusses the interactions among the integration framework, SAM, and the root-cause diagnosis tools.

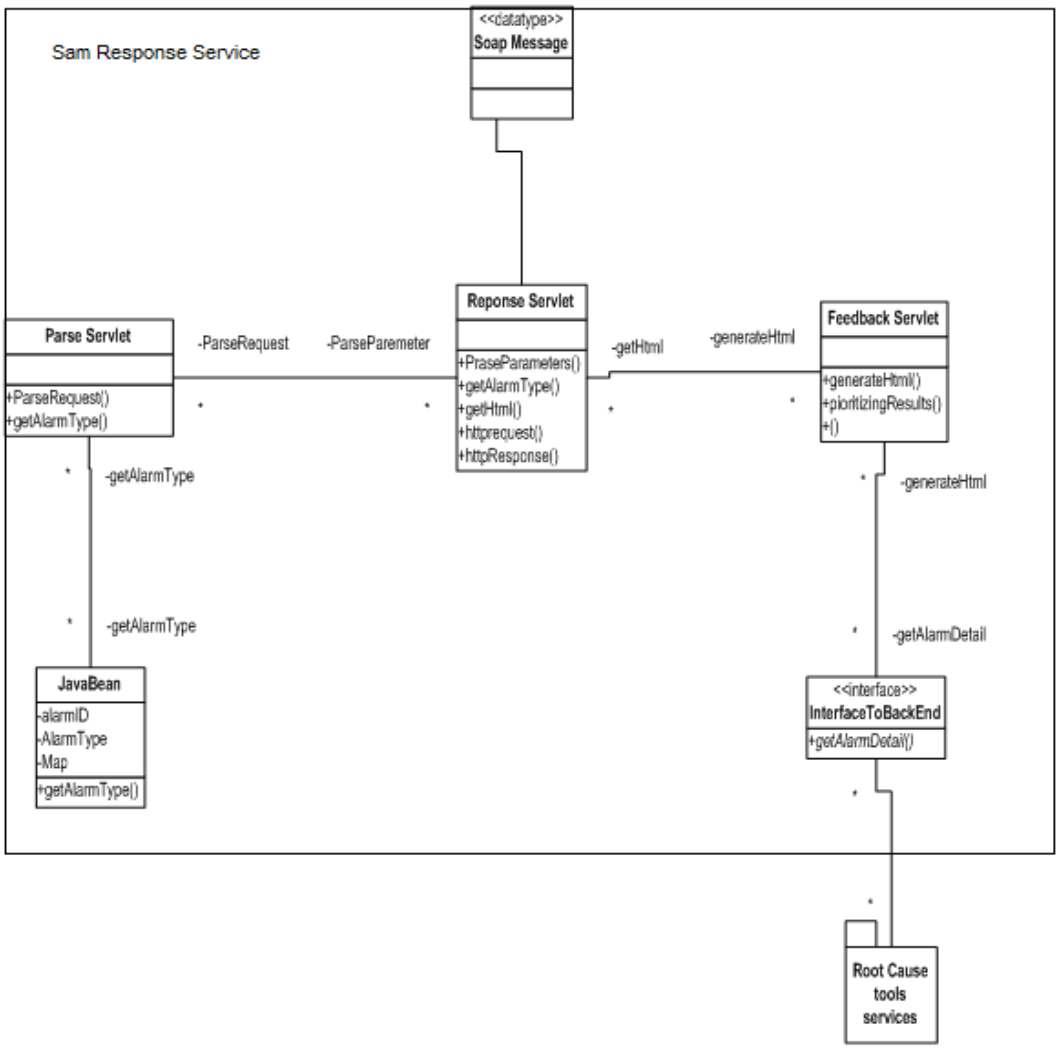


Figure 36: Class Diagram of the Response Service

We chose Java for developing our integration framework due to its compatibility and support for web applications. There are three Java servlets that interact with each other inside this service part of the integration framework. The *Response Servlet* acts in the middle of the process to deal with any incoming request from SAM in the SOAP format. Upon receiving such a request, the servlet invokes *Parse Servlet* and *Feedback Servlet* to perform different actions. It also contains the functionality to construct fishbone diagrams as per the obtained results from the root-cause diagnosis results. Furthermore, it handles service subscriptions from the root-cause detection

tools. The *Parse Servlet* contains a message parser that issues a prompt notification about alarm details received from the *Response Servlet.* The details received from the *Response Servlet* conform to the required data structure. The request message, consisting of alarm details, is also formatted before it is dispatched to the root-cause analysis tools.

The *Feedback Servlet* analyzes results from the diagnosis tools and returns the best possible diagnosis result to the *Response Servlet.* It also has the ability to perform RMI on root-cause diagnosis tools to commence the diagnosis.

The illustration below (cf. Figure 37) shows the interaction flow between the servlets, data structure, and root-cause diagnosis tools. As the initial step, SAM dispatches an investigation request to the *Response Servlet* which contains details about a specific alarm. The *Response Servlet* asks the *Parse Servlet* to analyze the request and transform it into a readable format for the root-cause diagnosis tools. The transformed message is sent back to the *Response Servlet,* which refers the new message to the *Feedback Servlet* for root-cause analysis. With the help of the root-cause diagnosis tools, the *Feedback Servlet* sends the analysis result back to the *Response Servlet* which in turn draws a fishbone diagram according to the result.

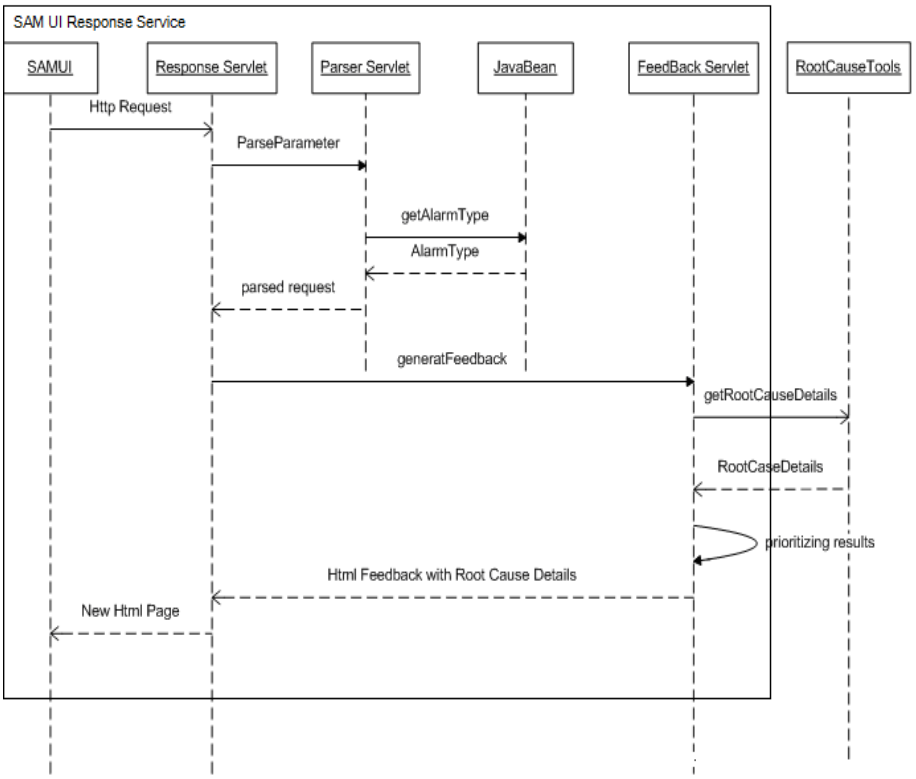


Figure 37: Interaction Diagram for the Response Service

4.6 Root-Cause Detection Tools and the Response Service

As discussed in previous sections, all root-cause diagnosis tools subscribe to the SAM integration framework. This section discusses the methodology through which the subscription architecture is achieved and its effectiveness with respect to the integration framework and the root-cause diagnosis tools. The top part of Figure 38 depicts the sequence of integration framework and the connection to root cause diagnosis tools.

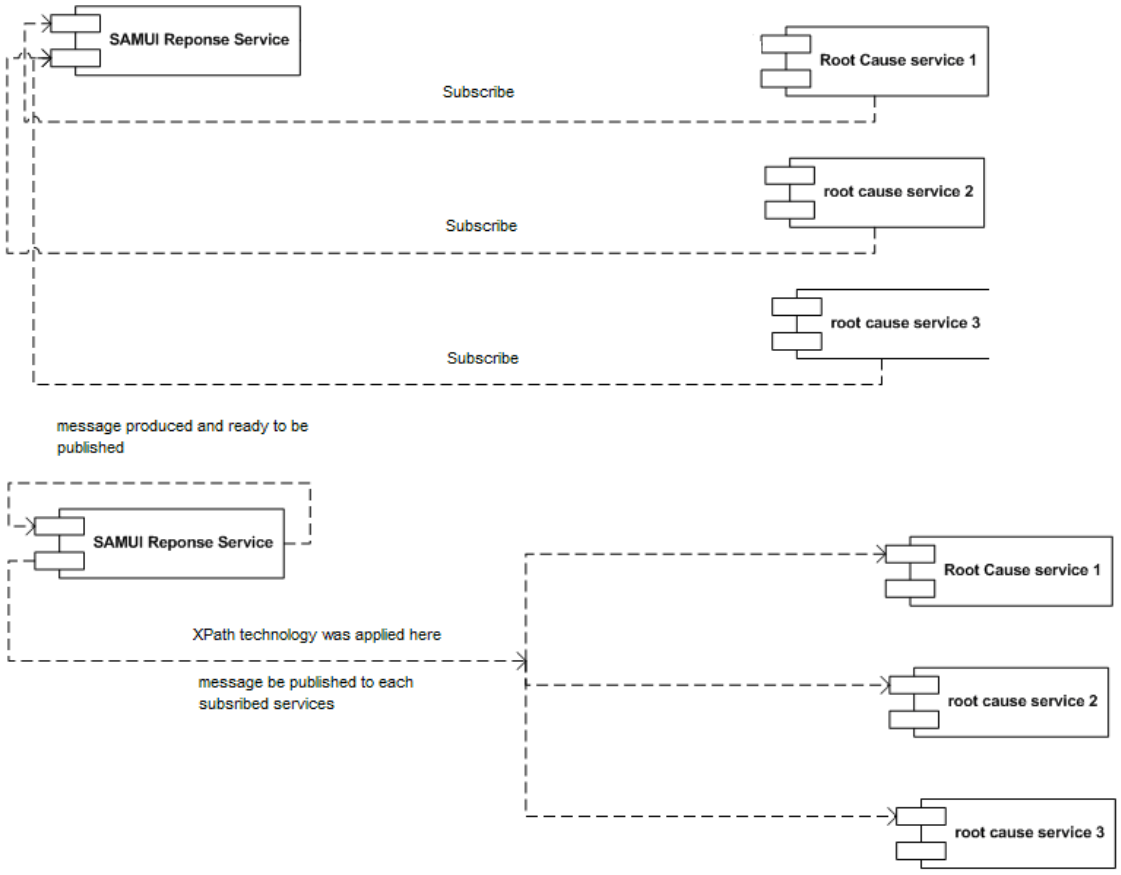


Figure 38: Interaction between Response Service and Root Cause Detection Tools

Since each error diagnosis tool runs as a web service, they are identified as root-cause services in this figure. Once the framework service receives an alarm diagnosis request from SAM, it transforms the message into the accepted system format and dispatches it to every subscribed root-cause service. Different alarms contain different attributes and these attributes decide which root-cause service will handle the diagnosis. This analysis procedure is achieved by distributing key attributes along with the diagnosis request to the respective tools. In this manner, subscribed

root-cause services can match their key required attributes with the attributes sent by the response service. The solution with most matches is selected to diagnose the particular alarm.

Since these messages are XML based, we can take advantage of the data format and perform a simple XPath query to find out the most suitable root cause service for a specific alarm. A sample XPth query is shown below:

> XPath
> <wse:Filter Dialect="xs:anyURI"? > xs:any </wse:Filter>
> Where anyURI can be retrieved with XPath /s: Envelope/s: Body/*/wse: Filter/@Dialect

4.7 Message Interactions

The root-cause detection tools are supposed to subscribe to the response service via the *Response servlet.* We decided to apply *WS-Eventing* standards to implement this subscription feature [17]. As mentioned earlier, the SAM integration project is a message based application. Thus, we need design the message interaction between SAM, response service, and root-cause detection tools.

4.7.1 A Sample Message between SAM and Response Service

This request message almost contains detail about a specific IT alarm from SAM including service serverity, creation time and alert details.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsd:schema xmlns:xsd"http://www.w3.org/2001/XMLSchema"
targetNamespace="taget server"
xmlns="http://localhost:8080/~xc/products"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
<xsd:element name=" Details">
<xsd:complexType name="Metadata">
<xsd:sequence>
<xsd:element name="alarmId" type="xsd:string"/>
<xsd:element name=" ACKNOWLEDGED " type="xsd:string"/>
```

```
<xsd:element name=" ALERTDETAIL " type="xsd:string"/>
<xsd:element name=" ASSIGNED " type="xsd:string"/>
<xsd:element name=" CONNECTOR_NAME " type="xsd:string"/>
<xsd:element name=" CREATION_DATE " type="xsd:date"/>
<xsd:element name=" EVENT_SOURCE " type="xsd:string"/>
<xsd:element name=" EVENT_SOURCE_ID " type="xsd:string"/>
<xsd:element name=" EVENT_OCCURRED " type="xsd:date"/>
<xsd:element name=" PRIORITY " type="xsd:string"/>
<xsd:element name=" SEVERITY " type="xsd:string"/>
<xsd:element name=" SITUATION_TYPE " type="xsd:string"/>
<xsd:element name=" TICKET_ID " type="xsd:string"/>
<xsd:element name=" USER_CLEARABLE " type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
 <xsd:element complexType name=”Context”>
      < xsd:complexType>
        < xsd:sequence>
          < xsd:any namespace="##any"/>
        </ xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

And the message that root cause application servers gonna return according to the request will contain a list of possible root causes and Hypothesis of each group.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
soapenv:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soapenv:Header/>
<soapenv:Body>
<element name = “Alarms”>
<element name=" SamAlarms">
<complexType name="Hypothesis">
  <sequence>
  <element name = “Name” type = “string”>
  <element name = “Symptom Database ID” type = “string”>
  <element name = “Call Back URL” type = “url”>
  <element name = “Pruning URL” type = “url”>
  </sequence>
</complexType>
<WilyAlarm>
</WilyAlarm>
<SpecturmAlarm>
```

```
</SpecturmAlarm>
</element>
</soapenv:Body>
</soapenv:Envelope>
```

4.7.2 Messages for Subscription Creation

To create a subscription:

```
  <wse:Subscribe …>
    <wse:EndTo>endpoint-reference</wse:EndTo> ?
    <wse:Delivery Mode="xs:anyURI"? >Xpath</wse:Delivery>
    <wse:Expires>[xs:dateTime | xs:duration]</wse:Expires> ?
    <wse:Filter Dialect="xs:anyURI"? > Xpath </wse:Filter> ?
    …
  </wse:Subscribe>
```

Response to subscription creation:

```
   <wse:SubscribeResponse>
   <wse:SubscriptionManager>
      <wsa:Address>
       http://www.example.org/oceanwatch/SubscriptionManager
      </wsa:Address>
      <wsa:ReferenceParameters>
       <ow:MyId>
         28
       </ow:MyId>
      </wsa:ReferenceParameters>
    </wse:SubscriptionManager>
    <wse:Expires>P0Y0M0DT30H0M0S</wse:Expires>
</wse:SubscribeResponse>
```

4.8 Conclusions

The SAM integration project does not apply a regular point-to-point integration approach since it utilizes messaging systems to communicate between the four different root cause diagnosis tools. This is as one-to-many integration solution that works with all types of root-cause diagnosis tools. Since this is not an EAI integration project, the approach only supports integration of the root-cause detection tools with SAM. It will not support other third party solutions without significant re-design of the user interface, messages, and data structures. This approach is

relatively effective as it costs less time and resource to build. The next chapter discusses the advantages and disadvantages of each integration approach and possible future work with respect to each approach.

Chapter 5 Comparison and Future Work

This chapter discusses both EAI and point-to-point integration approaches and discusses their advantages and disadvantages. Towards the end of the chapter, we outline an alternative strategy with regards to software integration.

5.1 Point-to-Point Integration

In a point-to-point integration framework, a unique connector component is implemented amid two applications that need be bundled. The connector handles all communications that occur between the two applications. In other words, such a connector is responsible for handling all the data transaction, integration, and other messaging flows between the specific pairs of applications [17]. As an example, the SAM integration framework acts like a connector between the core application and the root-cause detection tools. All data that is transmitted through SAM and the diagnosis tools has to bypass the SAM integration framework. The integration framework holds the key responsibility of defining all necessary interfaces for the two applications that will be joined for communicating.

5.1.1 Advantages of Point-to-Point Integration

The point-to-point integration works well with small infrastructures where only a few applications need to be connected together. Integrating an infrastructure with a small number of components, such as SAM and root-cause detection tools, does not entail time and resources for product design and development by applying point-to-point integration. Instead, several simple connectors among the applications are sufficient to integrate the entire infrastructure. Thus, a

compact point-to-point integration solution for small infrastructures could result in a lot of savings in person power on front design and development costs.

5.1.2 Disadvantages of Point-to-Point Integration

The point-to-point integration solution may not be suitable for large and more complex environments. For instance, in a three component infrastructure, only three connectors need be implemented to fully integrate the infrastructure (cf. Figure 39). However, if the number of components increases to five, a total of ten connectors would be required to fully integrate the infrastructure (cf. Figure 40). As the number keep growing, the number of connectors necessary will grow exponentially (i.e., $n(n\text{-}1)/2$). Moreover, in an integrated environment, connectors must be maintained and operated separately from each other. Thus, in a complex infrastructure with a large amount of components, point-to-point integration is no longer a viable integration option since it might require more resources than an EAI approach while delivering lower service quality compared to EAI.

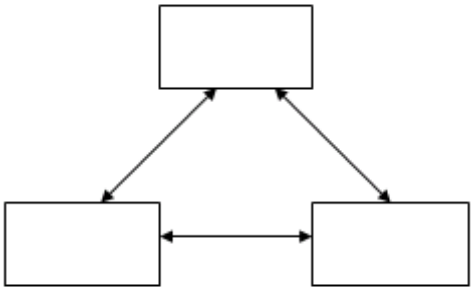


Figure 39: P-to-P Integration in a 3 Components Infrastructure

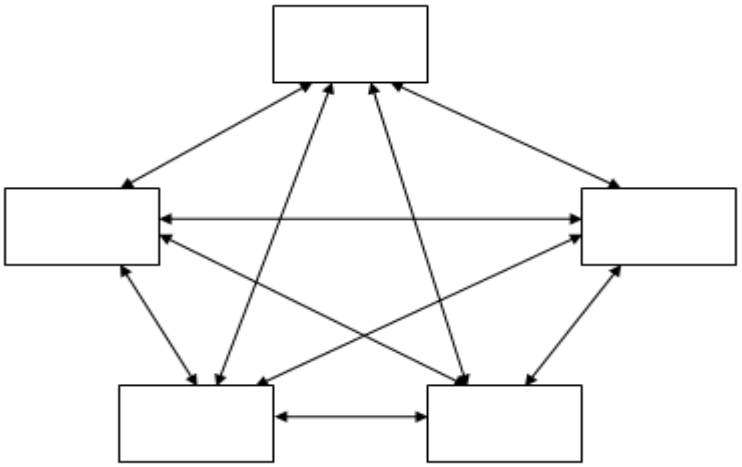


Figure 40: P-to-P Integration in a 5 Components Infrastructure

5.2 EAI

In the past, there have been two major problems with software integration. The first problem is related to the design quality of the software. In the old days, enterprise systems were designed poorly since many organizations only considered adapting modern technologies without considering how long the technology can last. Their focus did not include how the applications would communicate and share information with each other in the future. Due to this approach, a large number of legacy systems exist in the IT environment, however it is very difficult to integrate these legacy systems due to their monolithic design.

The second problem that exists, as mentioned in the previous section, is the old point-to-point integration technology that cannot sustain in business environments in the long term. Thus, any new solution should include capabilities to adapt to changes easily.

EAI was developed to integrate a large number of independent or legacy systems in the enterprise. It has a middleware layer and a message broker to provide standard data sharing mechanisms inside an organization so that any newly developed application can implement this standard for integration into the existing EAI framework.

5.2.1 Different Types of EAI

In order to establish a solid EAI implementation, it is important to know the kind of business and the type of data required for processing. This information is also important to determine the amount and cost of work required for the entire implementation. Typically, implementation using the EAI technology is divided into four major levels: *application interface level, data level, UI level* and *method level.*

Application interface level: This level of EAI refers to the mechanism that provides an ample number of interfaces to current applications. Developers can apply the package to access various functions of different applications. This aspect of EAI allows existing applications to send and retrieve data through the package thereby making the integrated system relatively easy to maintain while requiring few changes to the existing system. However, this approach requires that the initial system is well designed. It is also critical to provide interfaces to restricted resources to alleviate concerns regarding privacy or security breaches during the integration process.

Data Level: This level of EAI provides data integration for applications by manipulating the data flow passing through them. A data processor transforms and processes the same data on every application inside the integration framework. The biggest advantage of this approach is that no

code level changes are required for existing applications. Nevertheless, this approach only suits applications that mainly rely on data processes.

UI Level: This level of EAI refers to interface bundling of different applications to form a new interface for the integrated application. Since, it lacks usability and reliability, this integration approach is not preferred in most cases except for mainframe applications that require limited access of data and business processes.

Method Level: This level of EAI allows applications to share their business logics. This approach is mostly adopted for integrating applications within the same organization. [18]

5.2.2 Benefits of EAI

The most significant difference between EAI and other integration methodologies is EAI's recognition as a solid and elegant middleware that functions as a software tool at the central point of all the integrated applications. The major benefit brought by this middleware is to regulate the data exchange between all applications so that new applications can easily follow the rules to communicate with existing applications.

Moreover, the middleware provides levels of indirection that hide the complexities of the source code from integration. Therefore, developers only need to focus on their own portion of work rather than be concerned about code changes in different levels.

The middleware also allows applications to be loosely coupled by exchanging data and information asynchronously. Thereby, by knowing the message endpoint, one application can

send a message and continue to work on other tasks without waiting for a response from the receiver. This feature in the EAI framework is more effective than the point-to-point framework. [19]

EAI can always utilize applications' API and database to format a common data exchange set-up for exchange of messages inside the EAI framework. Thus, in many cases, EAI is a perfect solution to evolve existing legacy systems into new solutions since few changes are required. These changes are implemented inside the legacy applications and EAI often does not require extensive programming and interface changes to the legacy systems.

In addition, EAI has huge advantages as a web client. By using XML as a formal data exchange format, EAI has the capability to release their services online so that applications across the world can be integrated without any server or data center relocation.

5.2.3 Disadvantages of EAI

After discussing the benefits, this section covers the disadvantages if EAI. As mentioned in the previous section, EAI's middleware handles the entire data traffic of the integration framework. The key concern is the robustness of the middleware since a failure can disconnect every application in the EAI environment. Obviously, this outcome is worse than a solution where a single connector stops working in a point-to-point integration framework, affecting only two applications.

Besides, development of the middleware could accumulate a great deal of person power, time and resources during the design phase. The middleware development not only requires inclusion of all necessary functions for current applications, but also requires an expandable middleware approach for future applications, thereby making the EAI framework rather difficult to build. Compared to traditional point-to-point integration, EAI systems are not only more expensive but also a lot more difficult to build.

As a new technology, there is no clear standard to implement EAI solutions. Plenty of early developed EAI systems proved to be rather heavyweight, expensive, and only worked under certain system conditions. According to a recent study “70 percent of integration projects ultimately fail due to the flaws in early broker solutions” [19]. Thus, companies need to investigate further before allocating big investments towards building EAI systems due to the failure rate of past projects.

5.3 Summary

Table 1: Comparison between EAI and P-to-P integration

	EAI	Point-to-Point
Overhead	--	++
Data integration	++	-
Time to build from scartch	--	++
Code modification to add new tool	+	-
Service response speed	_	+
Integrate small amount of applications	--	++
Integrate large amount of application	++	--

This chapter discussed advantages and disadvantages of both traditional point-to-point integration and EAI. Table 1 shows the comparison results of EAI integration and Point-to-Point integration in a set of criteria after my research. As in the diagram, more "+" signs indicates the approach has more advantage in a certain criteria. Vice versa, more "-" signs indicate the disadvantages. From a technician's perspective, point-to-point integration is a superior choice when integrating a small number of applications. However, this solution does not have potential to grow in the future. EAI is a better approach for big organizations that want to integrate different and a large number of applications and services. EAI serves as a good approach for organizations that intend to continue using integration for potential future applications. EAI's

middleware broker architecture can effectively handle on-going integration by providing a standard mechanism for data communication. As mentioned earlier, the EAI framework costs more money and requires additional person power to develop. Organizations must thoroughly consider the benefits obtainable by an EAI framework before jumping onto the EAI integration bandwagon. In particular, the following questions should be considered before going the EAI route:

- How many applications need be integrated?
- Will there be a need to add additional applications in the future?
- How many communications protocols need be used?
- Do the integration requirements include routing, forking, or aggregation?
- How important is scalability to the organization in the long run?
- Does the integration situation require asynchronous messaging; publish/subscribe messaging models, or other multi-application messaging scenarios? [19]

Building an EAI framework consumes significant funds and person power than the traditional approach of point-to-point integration. However, an effective EAI solution can be cost-effective for organizations in the long run.

Chapter 6 Conclusion

6.1 Summary

As computing technology has become an integral part of society, managing complex IT environments in an efficient way, has become a challenge for today's businesses. CA Inc.'s SAM has emerged in the market place as an innovative solution to help organizations manage their IT environments with more control and leverage. It is a management tool that combines management functionalities for almost every aspect of an IT environment—from monitoring IT components to diagnosing root causes of errors. Moreover, it provides organizations with a thorough view of their IT environments with minimal human interaction.

To enhance SAM's capabilities for diagnosing errors and problems in IT environments, a research project entitled *Logging, Monitoring and Diagnosis Systems for Enterprise Software Application* was initiated to develop root-cause analysis and diagnosis tools [29]. To this effect data mining and learning algorithm tools were developed to augment the core SAM functionality in order to diagnose the problems that occur in the IT environments better and in particular trace them back to the root causes instead of just listing the errors.

After developing root cause analysis and diagnosis tools, they had to be integrated with SAM. As part of this research project, I developed a web-based point-to-point integration framework for this purpose. This framework utilizes Java technology to handle all layers of communication between SAM and the root cause analysis and diagnosis tools. To further improve the performance, the integration framework was built as a message broker so that the root cause diagnosis tools need to be subscribed to the framework to analyze and process the alarms and

errors detected by SAM. This message broker architecture provides the flexibility for new root cause analysis and diagnosis tools to be integrated into the existing framework and replace out-dated services.

In the meantime CA Inc. developed Catalyst, an EAI based integration framework, to integrate third party solutions into CA Inc.'s product including the root cause analysis and diagnosis tools. As a message based EAI framework, Catalyst uses message based channels to provide standard APIs to connect third party solutions to any existing CA Inc. framework. This integration approach allows Catalyst to not only integrate SAM and root cause analysis and diagnosis tools, but also present CA Inc.'s software products along with third party plug-ins.

6.2 Contributions

This thesis presented background on current IT management technologies, discussed their advantages and capabilities, outlined additional capabilities desired by IT organizations, and as a result the need to augment CA Inc.'s SAM. It is hoped that the innovative solution can assist organizations to better utilize their IT resources under a limited budget. We investigated different software integration approaches to integrate SAM with third party enhancement solutions to integrate root-case analysis and diagnosis tools.

The major contributions of this thesis include:

- In Chapter 2, I conducted background research into SAM and identified related work aspects.

- In Chapter 3, I proposed an EAI based integration approach to integrate third party solutions into SAM and conducted background research on Catalyst.

- In Chapter 4, I proposed and implemented a web based point-to-point integration framework to integrate root-cause analysis and diagnosis tools to SAM.

- In Chapter 5, I compared two integration approaches and defined advantages and disadvantages of each.

6.3 Future Work

There are many aspects of this research that can be further explored for a deeper analysis into integration in IT environments. This section discusses possible future work related to an EAI prototype proposed in Chapter 3. The disadvantages of traditional EAI, as discussed in Chapter 5, compel the industry to innovate a better and more stable EAI standard that can reduce the large failure percentage of EAI projects. A more advanced EAI model, known as Enterprise Service Bus (ESB), which is supposed to improve on the disadvantages of traditional EAI-based middleware.

6.3.1 Bus Architecture and ESB

Instead of gathering all functionalities for integration into a central middleware through traditional EAI based middleware, the bus-based EAI tries to reorganize these functionalities into separate components to reduce the risk of system failures in case of a malfunction. Additionally, these components can be grouped together in different ways to perform different integration tasks.

A more reliable EAI model, known as the ESB model, provides a common standard for applying EAI to software integration. This tactic is based on the bus architecture which divides EAI functionalities, such as transaction processing and error handling, into separate components.

6.3.2 ESB Services and Advantages

ESB-based EAIs configure endpoints for messages in a centralized position. They are also able to transform and convert messages into other formats that are commonly accepted by consumers. ESB is built to accept messages sent from most major protocols available today and can dynamically locate appropriate end users. If data fragments go missing in the middle of a few data transactions, ESB is capable of retrieving the missing data through existing message data. Above all, ESB provides better monitoring and security for the integration environment than traditional EAI [19].

ESB is a lightweight solution compared to the traditional EAI. Moreover, since it distributes work into smaller pieces, there is no need to build a centralized system for integration. Moreover, ESB based EAI can easily be expanded, since new applications are only required to coalesce their existing data transfer mechanism into the bus without being concerned about affecting the existing infrastructure [19].

6.4 Root Cause Analysis and Economics

A large portion of this thesis is talking about integrating root cause analysis tools with an existing software solution. This short section will present the value of doing root cause analysis for SAM.

“Root cause analysis is a problem solving method for identifying the root causes of problem ”[41]. There is a root cause behind every problem, and it is very important to find out the root cause to solve the problem forever. Economically, finding out the cause and solve it will be the best way to solve a problem. Since without taking out the root cause, the problem is very likely to happen again and again in future and eventually cost much more than the time and effort necessary to catch the root cause. Thus, CA’s decision to enhance root cause diagnosis capability of SAM is very critical to improve SAM’s overall performance and make SAM a solution that requires minimum attention while doing a solid job on managing IT environment.

Bibliography

[1] Wiki, “Enterprise IT Management,” http://en.wikipedia.org/wiki/Enterprise_IT_Management (Last accessed in July 2010).

[2] CA Inc., “CA Spectrum® Service Assurance,” http://www.ca.com/files/productbriefs/service-assurance-manager-pb-us_203765.pdf (Last accessed in July 2010).

[3] Forrest Research, “Market Overview: The IT Management Software Market in 2009,” http://www.forrester.com/rb/Research/market_overview_it_management_software_market_in/q/id/46930/t/2 (Last accessed in August 2010).

[4] CA Inc., “Innovations in managing IT service quality,” http://www.ca.com/files/WhitePapers/managing_it_service_quality_wp_215485.pdf (Last accessed in August 2010).

[5] ITIC, “ITIC Survey Indicates 35% of Companies Will Delay Network Upgrades for Lack of Money,” http://itic-corp.com/highlights/itic-survey-indicates-35-of-companies-will-delay-network-upgrades-for-lack-of-money/ (Last accessed in August 2010).

[6] Enterprise Management Associates, Inc., “CA’s Spectrum Service Assurance Manager Bridges Service Management Gap,” http://www.ca.com/files/industryanalystreports/ema-service-assurance-analysis_204682.pdf (Last accessed in Sept 2010).

[7] CA Inc., “Wily Introscope,” http://www.ca.com/Files/ProductBriefs/wily-introscope-product-family-brief_204415.pdf (Last accessed in Sept 2010).

[8] Zone, “Service Assurance-Aware Billing: Satisfaction Guaranteed,” http://www.billingworld.com/articles/2010/02/service-assurance-aware-billing-satisfaction-guar.aspx (Last accessed in Sept 2010).

[9] CA Inc., “Service Assurance,” http://www.ca.com/us/service-assurance.aspx (Last accessed in Sept 2010).

[10] J. LaFata and S. Hofmann, “Enterprise Application Integration a Primer in Integration Technologies,” http://www.liquidhub.com/docs/Horizons_EAI.pdf (Last accessed in Sept 2010).

[11] M. Waschke, and others, “CA’s Unified Service Model,” http://www.ca.com/files/whitepapers/unified_service_model_whitepaper_final.pdf (Last accessed in Sept 2010).

[12] Wikipedia, “Enterprise Application Integration,” http://en.wikipedia.org/wiki/Enterprise_application_integration (Last accessed in Sept 2010).

[13] B. Woolf, “Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions,” http://www.eaipatterns.com/toc.html (Last accessed in Sept 2010).

[14] D. Ferguson, “What Are CA Catalyst and the Unified Service Model?” ,http://community.ca.com/blogs/ctoblog/archive/2010/03/22/what-are-ca-catalyst-and-the-unified-service-model.aspx (Last accessed in Sept 2010).

[15] M. Fowler, “Errant Architectures,” http://www.drdobbs.com/184414966 (Last accessed in Sept 2010).

[16] Google Code, “GWT,” http://code.google.com/webtoolkit/overview.html (Last accessed in Sept 2010).

[17] D. Box et al., “Web Services Eventing,” http://www.w3.org/Submission/WS-Eventing/#Introduction (Last accessed in Sept 2010).

[18] S. Mota, “Enterprise Application Integration (EAI),” http://www.roseindia.net/eai/enterpriseapplicationintegration.shtml (Last accessed in Sept 2010).

[19] Mulesoft.org, “Understanding Enterprise Application Integration - The Benefits of ESB for EAI,” http://www.mulesoft.org/enterprise-application-integration-eai-and-esb#esb-next-step (Last accessed in Sept 2010).

[20] Wiki, “Enterprise Service Bus.” http://en.wikipedia.org/wiki/Enterprise_service_bus (Last accessed in Sept 2010).

[21] Apache Tomcat, http://tomcat.apache.org/ (Last accessed in Sept 2010).

[22] IBM developerWorks, "Web Service Eventing," http://www.ibm.com/developerworks/webservices/library/specification/ws-eventing/ (Last accessed in Sept 2010).

[23] Wiki, "SOAP," http://en.wikipedia.org/wiki/SOAP (Last accessed in Sept 2010).

[24] C. Gu et al., "As SOA based Enterprise Application Integration Approach," http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.jsp?tp=&arnumber=5557378&queryText%3DAn+SOA+Based+Enterprise+Application+Integration+Approach%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[25] Z. Chen et al., "Research on Enterprise Application Integration Categories and Strategies," http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.jsp?tp=&arnumber=5384638&queryText%3DEnterprise+Application+Integration%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[26] X. Ji, "A Web-Based Enterprise Application Integration Solution," http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.jsp?tp=&arnumber=5234730&queryText%3DEnterprise+Application+Integration%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[27] C. Mayerl et al., "SOA-based integration of IT service management applications," http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.jsp?tp=&arnumber=1530875&queryText%3DIT+management%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[28] A. Brown et al., "A Best Practice Approach for Automating IT Management Processes," http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.jsp?tp=&arnumber=1687536&queryText%3DIT+management%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[29] Q. Zhu, "Goal Trees and Fault Trees for Root Cause Analysis," In Proceedings IEEE Int. Conference on Software Maintenance (ICSM 2008), pp. 436-439, Beijing, China, 2008.

[30] Q. Zhu, "Adaptive Root Cause Analysis and Diagnosis using Complex Event Processing and Feedback Loops," Ph.D. Dissertation, Computer Science, University of Victoria, April 2011.

[31] H.A. Müller, M. Shaw, and M. Pezzè. "Visibility of Control in Adaptive System," Second International Workshop on Ultra-Large-Scale Software-Intensive Systems (ULSSIS 2008), Workshop at 30th ACM/IEEE International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 2008.

[32] Y. Huang, D. Gannon, “A comparative study of Web services-based event notification specifications,” http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.jsp?tp=&arnumber=1690679&queryText%3Dws+eventing%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[33] R. Jayasinghe, D. Gamage, and S.Perera, “Towards Improved Data Dissemination of Publish-Subscribe Systems,” http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.sp?tp=&arnumber=5552746&queryText%3Dws+eventing%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[34] Wiki, “Ishikawa Diagram,” http://en.wikipedia.org/wiki/Ishikawa_diagram (Last accessed Dec 2010).

[35] J.A. Dargham and others, “Computer-based messaging system,” http://ieeexplore.ieee.org.ezproxy.library.uvic.ca/search/srchabstract.jsp?tp=&arnumber=1033056&queryText%3Dmessaging+system%26openedRefinements%3D*%26searchField%3DSearch+All (Last accessed Dec 2010).

[36] J. Williams and P. Richardson, “CA CATALYST-Integration to simplify your complex IT Environment,” (Last accessed Dec 2010).

[37] BOB Tech Solutions, “Enterprises IT,” http://www.bob-technologies.com/application_developement.htm (Last accessed Dec 2010).

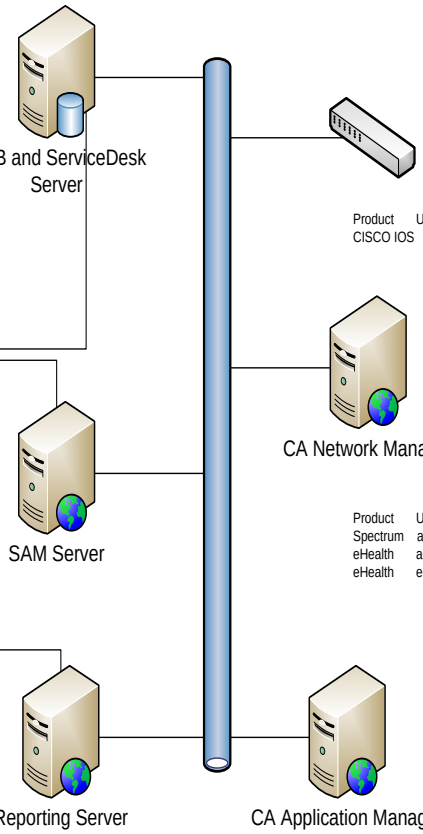
[38] R. Tuchyňa, “Service Assurance Management,” http://www.caforum.ro/presen/sam.pdf (Last accessed Aug 2011)

[39] nimsoft. A IT environment monitoring software compay. http://info.nimsoft.com/performance-monitoring.html?_kk=712a74f6-c0e6-41ab-a03d-1fd7f0390d0c&_kt=7239524219&gclid=CIaBtrTNmasCFcJrKgodj0ppfQ (Last accessed Aug 2011)

[40]ConnectWise. http://www.connectwise.com/landings/5-2/it-solutions/fivetwo.mvc?source=CW-PPC-Google-NAITS-3MQT-050211&utm_source=google&utm_medium=cpc&utm_term=information%20technology%20management%20software&gclid=CNz2rtLNmasCFUkCQAodGX8cmQ (Last accessed Aug 2011)

[41] Matt, “Root Cause Analysis”, http://www.sayeconomy.com/root-cause-analysis/ (Last accessed Aug 2011)

Appendix A: Installation Structure of SAM



CA-MDB-SDESK
192.168.38.75

MS SQL 2005 – default instance
CMDB r11.2 / Service Desk r11.2
NSM World View r11.2 / Master UKB (DIA)
EEM8.4

Product	Username	Password	Port
Servicedesk / CMDB	servicedesk	servicedesk	80 / 8080
	rhduser	rhduser	
EEM	eiamAdmin	oneclick	5250
SQL	sa	oneclick	1433
NSM	nsmadmin	unicenter	
Visualizer			9080

CA-SAM
192.168.38.76

CA SAM 1.0 Manager and UI
MS SQL 2005 Client
NSM r11.2 Manager with DSM
Portal and Dashboard services
Email Server

Product	Username	Password	Port
Portal	admin	oneclick	8090
SAM App Server			7090
SAM UI Server			7070
Active MQ			61616
Default SAM user	samuser	oneclick	

WebServer1
192.168.38.77

Business Objects XI
MS SQL 2005 Client
MySQL
Test Application for Scenario One

Product	Username	Password	Port
Boxi	administrator	Un1center	8080
	cms_user	oneclick	

Switch-1
192.168.38.96

Emulated Switch / Router running on image CA-Network-Mgr

Product	Username	Password	Port
CISCO IOS		oneclick	

CA-Network-Mgr
192.168.38.93

Spectrum 9.0
eHealth 6
Switch / Router Emulator

Product	Username	Password	Port
Spectrum	administrator	spectrum	80
eHealth	admin	ehealth	81
eHealth	ehealth	ehealth	

CA-Wily
192.168.38.94
192.168.38.95

Wily / CEM
Insight

Product	Username	Password	Port
CEM	cemadmin	quality	80
	admin	quality	|
Introscope	administrator		8080

Operating System Account	
Administrator	**unicenter**

Appendix B: Use Cases for SAM Integration Project's User Interface

System under consideration

The term *server* refers to a software component that receives HTTP requests and generates HTML documents in response to them. The requests can originate from SAM user interfaces or from the UI components generated by the server itself.

HTTP requests might contain:

- Alarm ID
- Alarm attributes
- Hypothesis ID
- Effect ID

Root cause detection tools outside the system boundary. The server communicates with root-cause detection tools using a web services interface.

Use Case 1

Title: Generation of initial *Investigation User Interface.*

Trigger: User selects "Investigate" menu option on an alarm.

Preconditions: SAM collected alarm related data, provided them as attributes, and provided access to the data with URL passed during menu selection.

Main Body

1. Server receives HTTP request with alarm and alarm attributes
2. Server parses the request
3. Server extracts alarm identifier from the request
4. Server extracts alarm attributes from the request
5. Server creates *generate hypothesis* request for the root cause analysis tools
6. Server invokes root cause detection tools
7. Server waits to receive root cause hypothesis information from the tools
8. Server parses root-cause analysis hypotheses returned by the root-cause detection tools in response to request issued in Step 6.
9. Server ranks the returned results
10. Server creates fishbone diagrams
11. Server provides color-coding of ranking information
12. Server generates HTML page containing fishbone diagrams and other UI elements of the investigation interface
13. Server returns the HTML page in response to the HTTP request received in Step 1

Post Condition: SAM displays fishbone diagrams of the possible root causes

Use Case 2 – extension for Use Case 1

Title: Generation of root cause refinement *Investigation User Interface*

Trigger: User clicks *find related events* button in the investigation interface

Preconditions: User selected a hypothesis that has to be refined.

Main Body

Step 5 is replaced by:

1. Server extracts hypothesis from the request
2. Server creates *find evidence* request to the root cause analysis tools

Post Condition: SAM displays fishbone diagrams of the possible root causes with refined evidence

Use Case 3 – extension for Use Case 1

Title: Generation of consecutive *Investigation User Interface*

Trigger: User clicks *find more hypotheses* button in the investigation interface

Preconditions: User selected a head of the fishbone for further generation of hypothesis

Main Body

Step 5 is replaced by:

1. Server extracts effect from corresponding to selected head of the fishbone from the request
2. Server creates *generate additional hypothesis* request to the root cause analysis tools

Post Condition: SAM displays fishbone diagrams of the possible root causes with refined evidence

Appendix C: Sample SWT Code to Generate Fishbone Diagrams

```
public MyJsGraphicsPanel buildFishBone (MyJsGraphicsPanel graphPanel,
                                        int anotherYDelta, boolean partial){

    // Based on how many rows in the previous table, this delta is added to all the Y's here
    int deltaX = 550;
    int deltaY = -400 + anotherYDelta;
    if (anotherYDelta == 0)        {
        graphPanel.setColor(Color.LIGHT_GRAY);
        graphPanel.fillRect(50 + deltaX, 580 + deltaY, 200, 50);
        graphPanel.setColor(Color.BLACK);
    }
    graphPanel.drawRect(50 + deltaX, 580 + deltaY, 200, 50);
    graphPanel.drawStringRect(alertMsg, 53 + deltaX, 583 + deltaY, 200, 50);

    // Starting positions
    int startX = /*350*/600 + deltaX;
    int startY = 540 + deltaY;
    int endX = /*250*/500 + deltaX;
    int endY = 590 + deltaY;
    boolean goDown = true;
    boolean evenTimes = true;
    hypothesisNumber = 0;
    boolean turnWhite = false;

    int i =0;
    while ((i < 3)) {
        int j =0;
        while ((j < 3)) {
            hypothesisNumber++;
            if ((rcDescriptions[i][j] != null) && ToolsSelected[i]) {
                if (turnWhite) graphPanel.setColor(Color.WHITE);
                graphPanel.drawLine(startX, startY, endX, endY);
                // Increment y when flip is true, and increment x when flip is false.
                if (goDown) { //if true means we go up and move to the right
                    if (turnWhite) graphPanel.setColor(Color.WHITE);
                    graphPanel.drawEllipse(startX-20, startY-20, 40, 40);
                    if (!turnWhite)
                        graphPanel.setColor(Color.GREEN);
                    else
                        graphPanel.setColor(Color.WHITE);
```

```
                                graphPanel.fillEllipse(startX-20, startY-20, 40, 40);
                                if (!turnWhite)
                                        graphPanel.setColor(Color.BLACK);
                                else
                                        graphPanel.setColor(Color.WHITE);
                                graphPanel.drawStringRect(rcDescriptions[i][j],
                                startX-10, startY-45,200, 30);

                                // prepare for next iteration
                                startY += 100;
                                goDown = false;
                                evenTimes = false;

                                // exit the loop
                                if (partial) turnWhite = true;
                        } else {
                                if (turnWhite) graphPanel.setColor(Color.WHITE);
                                graphPanel.drawEllipse(startX-20, startY-20, 40, 40);
                                if (!turnWhite)
                                        graphPanel.setColor(Color.GREEN);
                                else
                                        graphPanel.setColor(Color.WHITE);
                                graphPanel.fillEllipse(startX-20, startY-20, 40, 40);
                                if (!turnWhite)
                                        graphPanel.setColor(Color.BLACK);
                                else
                                        graphPanel.setColor(Color.WHITE);
                                graphPanel.drawStringRect(rcDescriptions[i][j],
                                startX-10, startY+30, 200, 30);

                                // prepare for next iteration
                                startX -= 250;//update here
                                endX -= 250;
                                startY -= 100;
                                goDown = true;
                                evenTimes = true;
                        }
                }
                j++;
        }
        i++;
}

if (evenTimes) {
        endX +=250;
        startX +=250;
```

```
            startY = +100;
        }


        // Skeleton: line extends as far as I have hypotheses, so it should be at the end of the loop
        int [] xPoints = new int [10];
        int [] yPoints = new int [10];
        xPoints[0] = startX+270;
        yPoints[0] = startY+50;

        if (anotherYDelta > 0) {
            for (int count =1; count < 10; count++) {
                if (count < 5) {
                    xPoints[count] = xPoints[count-1] + (10-count-3) * (10-count -4) ;
                    yPoints[count] = yPoints[count-1] + 25;
                } else {
                    xPoints[count] = xPoints[count-1] - ( count -3) * ( count -4) ;
                    yPoints[count] = yPoints[count-1] + 25;
                }
            }
            graphPanel.setColor(Color.RED);
            graphPanel.setStrokeDotted();
            graphPanel.setStrokeWidth(2);
            graphPanel.drawPolyline(xPoints, yPoints);//
        }

        // draw the main fish line
        graphPanel.setColor(Color.BLACK);
        graphPanel.setStrokeWidth(1);
        graphPanel.drawLine(endX, endY, 500 + deltaX, 590 + deltaY);

        // Finally paint
        graphPanel.paint();
        return graphPanel;
}
```