“Migration Through Value Added Services” - A Viable Approach to a Service Orientated Architecture for the medical domain?

Alan Kiernan

A dissertation submitted to the University of Dublin,
in partial fulfilment of the requirements for the degree of
Master of Science in Health Informatics

2007

Declaration

I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work, and has not been submitted as an exercise for a degree at this or any other university.

Signed:_____

Alan Kiernan
11/09/2007

Permission to lend and/or copy

I agree that the Trinity College Library may lend or copy this dissertation upon request.

Signed:_____

Alan Kiernan
11/09/2007

Acknowledgements

I wish to express my gratitude to the following people, without whom this dissertation would not have been possible.

- To Emma Mc Keon for her absolute and unwavering support throughout this process.
- To my supervisor, Dr. Declan O'Sullivan of Trinity College Dublin for his constant advice, input and guidance.
- To Lucy Hederman and Mary Sharp of Trinity College Dublin, for their advice on specialist matters though this project.
- To Paul Duff, Pat Power and Dave Creevy of Accenture, for their support and facilitation of the time required to deliver this dissertation.
- Finally, to all those medical, managerial and information technology professionals who participated in the workshops and provided additional insight into particular medical domains.

Summary

This dissertation aims to answer the question “Is 'Migration through Value Added Services' a viable approach in the upgrade of legacy medical information systems to an SOA model?”

This strategy may be described as the migration of a legacy application into a Service Orientated Architecture (SOA) through the addition of new value added service components and to analyse the model from the perspective of different stakeholders.

The approach taken was 'Demonstration through the development of a proof of concept system' followed by performance testing, cost/benefit analysis and stakeholder workshops. The development of a core legacy emulator, followed by the wrapping of the system as a SOA component was undertaken to replicate a real world scenario, as described by medical experts.

An examination of this development through the development lifecycle was performed to gain first hand experience and allow for a personal analysis of this migration approach. This development allowed for the design and execution of a series of performance test, to determine the processing capacity and allow for a performance analysis of the resulting SOA platform. A cost benefit analysis was performed to gauge the financial practicality of this migration approach. Workshops were performed with three groups of medical related stakeholders (Medical Managerial Personnel, Medical Professionals and allied IT personnel) were performed to gather and allow for the analysis of the sectors opinion to the suggested migration approach.

Combined, the results from these studies allow for a validated conclusion to be drawn on the suggested migration approach.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

The medical industry is becoming more aligned to a services based infrastructure, with services distributed to specialist treatment centres and the formation of regional health information organisations (RHIO) [1]. Irelands Health Service Executive (HSE) is currently in the process of implementing shared services, with the eastern region implementing services across five sites, including one designated "multi functional" shared services centre.[2] In parallel, new IT architectures are being created to provide a "services orientated" application methodology, creating large-scale applications as a "sum of parts" of smaller specific services.[6] Given the suitability of a Service Orientated Architecture (SOA) to the services based medical paradigm, there is an increased effort in the migration of in-house, legacy systems to an SOA model. [1]

1.1 Purpose of Study

This project aims to answer the question "Is 'Migration through Value Added Services' a viable approach in the upgrade of legacy medical information systems to an SOA model?" This question is answered through the application and evaluation of literature reviews, a development exercise, performance testing, a series of workshops and a cost benefit analysis.

1.2 Motivation

While a significant amount of research has investigated the issue of legacy system migration, it is reasonable to say that some organisations may not have the time or resources to perform a full legacy system migration, yet require additional functionality from legacy system, preferable as part of a SOA. Little work has been done to date on a hybrid approach of "migration through value added services". Through this strategy, the core legacy system is retained as a "service" and newer value added functions are implemented as service components, interacting with the legacy system service. Eventually, the core legacy system may be rewritten as a "pure" service. Such a methodology may improve migration quality, delivery time and overall performance in the move from a disparate application environment to a services centric IT environment. The necessity to migrate from singular, legacy systems to a service centric architecture is examined in detail in chapter 2.

1.3 Methodology

The approach taken was 'Demonstration through the development of a proof of concept system' followed by performance testing, cost/benefit analysis and stakeholder workshops. Initially, an examination of the current migration methodologies, backed up with real world case studies was performed. The study then moved to examine factors influencing the decision to wrap or replace software and the process of user migration between systems was performed. The development of a core legacy emulator, followed by the wrapping of the system as a SOA component, referred to as the "proof of concept application" was undertaken to replicate a real world scenario. The system chosen was a disease laboratory processing system; ward staff submits tests and results are issued electronically. In addition, the ward staff may recall previous test results. Through discussion with medical personnel using such a system, the possible new requirement to such a system was defined as follows:

> *'Upon return of the results to the ward staff, the results are to be verified against the patients history to alert on both the issue of medication and risks (Based on previous history and current medications) and the recommendation of medication based on prior patient history'*

An examination of this development was performed to highlight the pros and cons of migration in this manner. This system was in turned used to perform a series of performance tests, used to examine the scalability of an SOA solution and to perform a series of workshops with 27 stakeholders from medical management, information technology and clinician backgrounds. Finally, a cost / benefit analysis on the development and deployment of an SOA solutions Vs a legacy system upgrade was performed to examine the rollout of an SOA solution in term of longer term costs and benefits.

Finally, the combination of analysis generated from these examinations was evaluated in the overall context of "Would this approach to legacy migration be viable?"

1.4 Outline of Dissertation

The following is an overview of the layout of this dissertation.

- Chapter 1: Introduces the scope of this dissertation, its motivation, methodology and outline.
- Chapter 2: Presents a literature review, examining the need to migration from legacy systems.
- Chapter 3: Presents a further literature review to examine the current migration methodologies, factors dictating the wrapping or replacement of a system and user migration approaches. Furthermore real world case studies are provided to highlight literature findings.

- Chapter 4 focuses on the development of a core legacy emulator, followed by the wrapping of the system as a SOA component, referred to as the "proof of concept application", undertaken to replicate a real world scenario. The chapter details the design of an emulated legacy system, a detailed definition of the new functional requirements and the design and development of the SOA proof of concept application, a view of the alternate legacy upgrade, customisation for medical environments and finally, an analysis of the development.
- Chapter 5 focuses on performance testing, in which the developed proof of concept application was used to perform a series of performance tests in both a simple and clustered execution environment. Tests were designed to test the throughput capacity of both the application and its execution environment, allowing for an analysis of the capacity and scalability of the new system to grow, in line with organisational expectations.
- Chapter 6 focuses on a series of workshops, through which 27 stakeholders from medical management, information technology and clinician backgrounds were presented with an overview of the migration approach, the application and a questionnaire. The chapter outlines the design of each workshop, its target audience, goals, composition and an analysis of each groups' results. Finally, a comparative analysis is provides across all workshop groups.
- Chapter 7 details a cost / benefit analysis on the development and deployment of an SOA solutions Vs a legacy system upgrade was performed to examine the rollout of an SOA solution in term of longer term costs and benefits. The chapter states the SOA and legacy systems used in the comparison, an investigation on how the proposed SOA system meets the organisations requirements, an examination of hard cash costs, soft cash saving, cost avoidance, benefits and finally, a comparison of these findings.
- Finally, chapter 8 provides an evaluation of the approach and a conclusion. An evaluation of the state of the art comparison, application development exercise, performance test results, workshop results and cost benefit analysis are detailed, followed by the dissertations conclusion. The chapter and dissertation then close on an analysis of the conclusion and dissertation critiques.

2 The Need for Migration

A legacy system may be defined as "a mission critical software system developed sometime in the past that has been around and has not changed for a long time without undergoing systematic remedial actions"[3] or "Any information system that significantly resists modification and evolution" [4]. This chapter argues the need for migration from legacy systems and the adoption of SOA.

2.1 Legacy System Issues

Given the restructuring of core business function over the past 30 years, legacy systems continue to play a vital role at the heart of major businesses. In viewing the health service as a business from the perspective of management, legacy systems drive long term and short term health functions.

"There are only a couple of problems. The first is to keep the IS's running. The business depends on them. The second is to modify them to meet current business needs." [4]

Globally, there are estimated to be 150 - 200 billion lines of legacy code running of proprietary mainframes across business domains representing an investment of $5,000 billion and accounting for four fifths of global IT resources. [5] Taking COBOL as the principle language of business applications built over the past 30 years, today, more application development and maintenance tools are available for COBOL than for any other programming language. As of 2000, Gartner research revealed that 70 percent of mission-critical applications were written in COBOL, and that through 2005, approximately 15 percent of all new application functionality would be in COBOL. [7]

Approximately 90% of the total cost of ownership of these systems is spent maintaining this software and managing its evolution. [8] On average over 60% of the typical IT application budget is allocated to maintaining and enhancing legacy environments. [5] It is reasonable to assume that this drains IT department investment in new technologies. Constant maintenance of legacy code can define the departments' goals as fire fighting current systems and budgets, not providing support in the work practices within the organization.

Legacy systems are proven to be inflexible, difficult to manage and expensive to run. Such inflexibility inhibits the introduction of new work practices. The cost associated with the introduction of new systems and new work practices can prove justified. In the case of the Cincinnati children's hospital the introduction of new technology from Siemens costing $15.8 million, including $1.8 million in hardware costs. However return on this investment was realized though reduction in both patient risk from human error and staff time. Times taken for specialist processes are reported to be reduced by as much as 75% through the use of new technology and

work practices. [9] Comparatively, other business sectors, such as insurance, expect to see a return in excess of 200% over costs. [10]

Globally, the health service is undergoing significant changes in service structure and definition. The concept of "Regional Health Organisations", specialist health centres and community based care are delivering more efficient care through advances in technology and information sharing. As legacy systems are presented to be ill-equipped to evolve at the pace of business practice change and easily share information with other organizations, alternative architectures and solutions must be examined to meet new requirements.

Furthermore, given the risk associated with the migration of any large computer system "cold turkey", surveys indicate that approximately 51% of businesses will replace components piece by piece and 31% will employ wrap and replace strategies. [10] This is not surprising considering most full-system replacements will take longer than the 6 to 18 months in which management would like to see a return in investment. [11] Bisbal, Lawless et al. describe the most serious problems associated with legacy systems as:

- The systems generally run on obsolete hardware which is slow and expensive to maintain.
- Maintenance of the software is generally expensive; tracing faults is costly and time consuming due to lack of documentation and a general lack of understanding of the internal workings of the system
- Integration efforts are greatly hampered by the absence of clean interfaces.
- Legacy systems are very difficult, if not impossible to expand.

[12]

Given these issues with legacy systems, management are often reluctant to migrate legacy systems, with concerns over migration costs and the impact to core business services.

2.2 Arguments for SOA Adoption

The use of a SOA provides the means to interact with the core business process provided by the legacy system, provide new business functionality and allow for the secure sharing of data to other systems in a platform and language independent manner. Through the use of SOA, "Organizations can shift their efforts from maintaining a complex data interface strategy to creating service-oriented applications that support interoperability while more closely aligning with healthcare processes." [13] Surveys indicate that 74% of CIO's and technology executives state that their goal is to have fully integrated systems based upon SOA within two years time. [14]

SOA is a flexible, industry supported, open standard. Standards such as Hypertext Transfer Protocol (HTTP), Simple Object Access Protocol (SOAP) and XML are used in web-service

communication, providing a mechanism for applications to interact without the need for a specific object model, such as those used by CORBA, DCOM and J2EE. [15]

Accenture [14] lists the advantages of SOA as:

- Enable organization to become more agile, respond quickly to new business imperatives, develop new capabilities and leverage existing systems.
- Promotes reuse of existing assets and reduction of application development costs.
- Create composite solution, supporting next generation development consolidating multiple legacy systems, custom business solutions, package applications into a single user interface or workspace.

Through the physical separation of components acting as a single composite application, services are easily located on disparate platforms over any series of networks. This lends to a robust system, where systems may be grown, shrunk, upgraded or replaced, as their use as part of single or series of composite applications changes. In terms of healthcare, this enhances the provider's ability to keep their services running 24*7 or as their service license agreement (SLA) defines. Plans for upgrades, release or reconfiguration may be carried out without the knowledge of other composite parties, with allowances for application redundancy in the technical architecture used to support the SOA application components.

With the migration of health services to a shared services, Regional Health Information Organisation (RHIO) dictates that patient information be shared at the point of care and as appropriate, with payers and employers. Through the use of SOA to provide composite applications, rapid implementation of new solutions which leverage current systems is possible. This results in a rapid return of investment, lower total cost of ownership, reduced administrative costs, improved physician and staff workflow and the ability to meet evolving legislative, regulatory and accreditation compliance. [1] From the patients perspective the use of SOA to support shared services and data integration amounts to potentially better standard of care and shorter stays in regional hospitals due to shorter waiting times for specialist care.

SOA has gained the support of industry, in a move that migrates from proprietary closed communications and message standards to open architectures. Major solution providers and innovators in the healthcare domain are supporting SOA as "the future of networked services". Such groups include Accenture, committing to 3 year initiatives to accelerate the development of SOA architectures customized for different domains, beginning with healthcare. Such initiatives cost in the region of $450 million. [14] Other global technology firms committed to SOA include Hewlett Packard [16], IBM[17], Capgemini [18], Sun Microsystems [19], Tibco [20], [21] and BEA [22].

Collaboration between such organizations is taking place to bring new solutions to the market. For example, SAP and Accenture are collaborating to produce a new Collaborative Health Network suite, using SOA to interact between SAP's CHN solution and Accenture's Electronic Health Record solution.[23]

With regards to the life expectancy of an SOA application, the nature of SOA allows for hardware and software components to be upgraded when required. With that, there is no "life expectancy" for the composite application, but the standard industrial life expectancy for hardware and software still applies.(Example: The life expectancy of an application server until the vendor decides it has reached end of life.)

Confidential information is secured over SOA web-services through the use of a set of well defined security standards. These include Security Assertion Markup Language (SAM) for identity federation and auditing, Extensible Access Control Markup Language (XACML) for the expression of access control policies and Web Service Security (WSS) to protect soap messages as they pass over the network. In combination SAM, XACML and WSS provide access control, message encryption and digital signing. Prior to WSS, the most common approach to protecting messages was to use the SSL or TLS protocols. While adequate, these do not have the specialized flexibility and capabilities designed explicitly for web-services. [24]

2.3 Summary

This chapter has examined the need for migration from legacy systems to a services centric SOA. The next chapter presents migration strategies currently in use by industry.

3 State Of The Art

This chapter examines the current technological approach applied to the issue of migration from legacy systems. A study of the current migration practices is detailed, followed by an examination of environmental factors which may influence the approach. Furthermore, four case studies are presented to highlight the current practise employed.

3.1 Current Migration Practices

There are several methodologies available to migrate a legacy system; these may be divided into those focused on extending the life of a legacy system through integrating the system with other architectures and those focused on the complete migration of the legacy system to a new architecture. This dissertation focuses on a combination of both methodologies. The legacy system is wrapped to extend its life through the capability to provide "modernized" services, thus allowing for new value added services to be added through the use of SOA. Following this stage, full migration can occur as and when required.

3.1.1 Partial Migration / Legacy System Integration:

Integration is focused on making the system accessible to an external system through wrapping the system and providing access through a defined interface. The interface may be provided through a number of mechanisms including screen scraping, data level wrapping, interface gateways, common gateway interface (CGI), object and component wrapping.

3.1.1.1 Screen Scraping

Screen scraping involves wrapping old, text based interfaces with new graphical interfaces. Typically, the old interface is a set of terminal screens. The new interface can potentially be in any format, including a PC based GUI, HMTL based web page or can consist of an intermediate application/gateway API. The technology can be extended easily, enabling one new UI to wrap a number of legacy systems. [25]

The old interface and new interface interact via commercial screen reading and writing tools – such an example is Winrunner. The vendor WRQ, describes this approach as "By far the simplest and least risky approach. It requires no changes to the host application and preserves the business rules, removing the need for replication." [26] While implementing this solution may prove a "quick solution", the system is inflexible and difficult to maintain, having earned the slang title "Whipped cream on road kill".

The issues associated with screen scraping are stated by ASNA Incorporated as "It vastly limits an organization's options for migration down the road. The model presupposes that the underlying

legacy system will always be available, that the platform on which it lies will always be available, will always be supported, will always be maintained and that the organization will never want to migrate the mission critical business applications off of it."[27]
However, the approach is useful where the application is stable, unlikely to require changes and the principle objective is to improve usability. [25]

3.1.1.2 CGI Wrapping

Legacy integration using the Common Gateway Interface (CGI) is often used to provide access to existing legacy assets through web based systems. The interface interacts directly with the legacy system to retrieve the required information, formats it and redisplays to the user. While easy to implement, this mechanism is difficult to maintain in a system which may require expansion. [25]

3.1.1.3 Data Wrapping & Data Gateways

"Data wrapping enables accessing legacy data using a different interface or protocol than those for which the data was designed initially."[25] This may be achieved through the use of proprietary interfaces provided by the legacy vendor. A data gateway is a software set that maps and translates between different data access protocols. To maintain flexibility, it is preferable to avail of an industry standard gateway when wrapping legacy data. This decouples the data access layer of the application from the legacy system, allowing reuse on other data sources and types.
There are two de-facto standards for gateway:

- Open Database Connectivity (ODBC) Provides access to relational and non-relational databases through a vendor-neutral mechanism.
- Java Database Connectivity (JDBC) Provides open access to databases which provide an implementation of the JDBC specification for their database. This may be thought of as the "Java Version" of JDBC.

Data wrapping is a simple mechanism for legacy system integration and is often at the core of the component wrapping approach. Integration is achieved by wrapping existing systems using web services and industry standards to provide a uniform and adaptable interface with functionality for interoperability as well as secured and controlled access amount the individual systems. [28]

3.1.1.4 Component Wrapping

The principal concept behind object wrapping is that applications, services and business data are represented as objects. Component wrapping may be seen as an extension of object wrapping, in that it components are objects which conform to a component model. As such, a component framework may be devised to provide quality services.

Zou and Kontogiannis describe a component as "a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties." [29]

There are currently three dominant component frameworks:

- CORBA 3 Component Model by OMG
- Enterprise Java Beans (EJB) by Sun Microsystems
- Distributed interNet Architecture (DNA) / .NET by Microsoft.

Taking EJB as an example, the legacy system may be wrapped at the level of business logic or data interface. [25] describe an EJB as "a transactional and secure remote method invocation (RMI) or CORBA object, some of whose runtime properties are specified at deployment using special files called 'deployment descriptors'". The EJB is executed in an environment which implements all quality services defined by the framework, known as an application container.

The EJB specification defines several types of beans:

- Entity beans: Represent data entities, of which there are two subversions – container managed, through which the application container is responsible for implementation of data access methods and table / object mapping through user defined bean descriptors and bean managed, through which the developer is responsible for the same data interface functionality.
- Session Beans: Implement business logic functionality.

The component wrapping methodology has several advantages:

- The process is efficient, with relatively little effort; the advantages of component based systems are supported. The components are flexible; the wrapped beans may be used in the future in unanticipated ways.
- The wrapped beans adhere to the EJB component framework and can be integrated fully with the management facilities and services defined by the framework and implemented in the application container.
- The component model provides a roadmap to substitute the legacy system incrementally.

[25]

3.1.2 Complete System Migration Methodologies

There are several methodologies employed for full system migration. The goal of this methodology is to move the entire application and data to the target platform.

3.1.2.1 Forward Migration

The data is initially migrated to the new system, followed incrementally by the business logic and interfaces. During the migration process, the legacy system remains running in parallel. [4]

3.1.2.2 Reverse Migration

The legacy application is migrated to the target platform while the data remains on the legacy platform. [4] Once the legacy application is migrated and testing is complete, the data is migrated as a final step.

3.1.2.3 Composite Database Approach

Using this approach, the legacy applications are gradually rebuilt on the target platform. During migration, the legacy data and applications form a "composite application" between the legacy and target systems with both a forward and reverse gateway in use to allow the legacy system to access data in the target platform and target system to access data in the legacy platform. As data will be duplicated on both systems and require synchronization, an intermediate data coordinator is used to intercept all data update requests and update data in both systems as required. [30] One refined version of this approach, the "Chicken Little" or Stepwise approach, proposed by Brodie and Stonebraker defines an 11 step generic process for composite migration, ending once all legacy system functionality has been executed and the target system is essentially a fully working clone of the legacy data.[4]

3.1.2.4 Butterfly Method

The butterfly method, devised as part of the MILESTONE project [31] defines a gateway free approach to migration, in which users will not access both the legacy and target systems simultaneously. The target system would remain in production until migration is complete. The process of migration is defined as a generic six phase process. The target system is configured with sample data and all non-data components are migrated. Following testing, the data is incrementally migrated and users are trained. Once complete, cut off may take place without the need of a final "cold turkey" data migration.

3.1.3 Composite Migration Strategy (Migration through Value Added Services)

This dissertation suggests the use of a composite migration strategy. Through the initial use of wrapping with new application interfaces based on web services, a Service Orientated Architecture can be introduced to the data access elements of the application while allowing the application to evolve to a composite model. The legacy system has essentially been wrapped to join a composite application at this point. Given that the purpose of most medical systems which are now considered

legacy are that of information storage and retrieval, leaving business and medical decision to clinicians and supporting administrative and medical staff, it is not unreasonable to view the system as data driven. As such, application business logic will prove minimal and can be reengineered as part of the legacy wrapper.

With the legacy system wrapped and the composite application deployed, users have what appear to be two systems running in parallel; when in-fact there are only two views of the same legacy system. Users can be "weaned" onto the composite application as required. The system may stay in this state as long as is required, with any core changes in the legacy system requiring a matching change on the new wrappers.

Once the decision is made to finally migrate the legacy system, the use of the composite database migration approach can incrementally migrate the legacy data to a target system. As the target system is also based on web services and SOA, integration with the new target system and the composite application which the legacy system has joined will be transparent.

Stevenson and Pols describe their experience in such a migration approach following the failure of a cold-turkey attempt. Success was achieved efficiently through a mixture of wrapping and adding new business functionality. The team moved from a scenario where reengineering was required of 250 legacy table to 10 for completion of core business functionality. Enormous inefficiencies were identified in the legacy system due to business management and IT practices such as poor documentation and the creation of redundant tables and code bases with each previous legacy release. [32]

3.2 Application Environment / Impact Study

In reviewing the current state of the art practices, the approach of rewriting and replacing the current system in its entirety and the approach of wrapping the software is examined. This section examines current practices and arguments for each. In addition, the issue of migrating users between a current legacy system and a new system is also of concern, as the end use and uptake of the system ultimately defines its success or failure, regardless of the technical foundations on which the system is based.

3.2.1 Application Environment: Software Rewriting / Replacing or Wrapping

There are several factors to consider when making the decision to wrap or replace the legacy system. These are strongly driven by the business case requiring new services of the legacy system. For the purpose of this dissertation, the legacy system is defined as a pathology lab order and result system, with new functionality required to check medication recommendations against patient data,

to reduce risk to the patient. The system is primarily data centric, with few business rules surrounding the result retrieval calls. In-fact, the primary function of the legacy upgrade is to introduce new business rules and integrate with previous unaffiliated data.

In view of system replacement, Heuvel states that this process "has proven to be a highly complicated, time-consuming, expensive and risky endeavour, as these systems are not only tightly coupled to the business workflow, but also suffer from a rigid and contrived architecture because of many years of ad-hoc patching and fixing, offering very limited openness to other systems."[33]

The replacement of a legacy system requires in-depth knowledge of both the system and the application domain. It is more than likely that the original and subsequent developers involved in creating and maintaining the system are unavailable and that detailed functional and technical design documentation are non-existent. With that, the requirements of the new system may be based on management's perception of the legacy systems function and not the legacy systems true function. This scenario is realized by Stevenson and Pols on a financial legacy system supporting 100 traders with 250+ database tables. [32]

Good [34] defines a simple checklist for the argument of rewriting and replacing as:

- Business rules satisfactory but needs extensive functionality added
- No off-the-shelf solution comes close to meeting needs
- Poor quality code in existing, with high maintenance costs
- Application significantly out of line with business needs
- Willing to make changes to business model to fit off-the-shelf solution
- Can afford time, cost and disruption involved

Given the nature of this projects upgrade proof of concept application, it is assumed that the application is stable (as it has managed pathology test requests and results) and that the business domain cannot afford the time or disruption involved in rewriting the system.

With legacy system wrapping, the system may be extended quickly through a number of technologies and frameworks. The form of wrapping selected, as discussed in the next section will dictate the systems extendibility in the future. One the surface wrapping appears to be a more efficient solution given that "On the one hand, (CIO's and IT departments) are being asked to improve efficiency, reduce operating and system maintenance costs and implement new technologies to meet new legislation and business needs. At the same time, budgets are being cut." Kimberly Harris, Gartner Group [11]

Good [34] defines a simple checklist for the argument of legacy system reuse as:

- Business rules satisfactory

- Low operational costs of existing application
- Difficult to separate logic from persistent data and presentation
- Simple Web access required, allowing a wrapping solution
- Have resources to keep core legacy maintained

In the migration of the projects proof of concept application, data level wrapping appears to be a better solution than rewriting the legacy system. This will allow for a quick access to legacy data as part of a composite application at low cost, application expansion in the future and as the application is data centric, poses little issues in the context of writing or wrapping business rules.

3.2.2 User Migration

Legacy Systems, developed over a period of time, tested and tweaked to satisfy provider requirements become trusted by the user. Any replacement system that provides inferior functionality or disrupts the care providers' workflow is likely to be rejected. [35] The use of parallel migration, in that the user is required to operate both systems at once disrupts workflow, similarly, a cold turkey cutover to a new system forces new work practices on the user, assuming that the new system is not identical to the legacy system. "In reality, users prefer familiar interfaces even at the price of lesser functionality. This is especially true for new, additional functionality." [35]

To avoid this scenario and allow users to continue to work seamlessly throughout migration, alternative approaches may be taken to user migration and system development.

One such approach, suggested by Schoenbery is the use of intermediately software. The MBRIDGE application allows the users to continue to use the legacy system while mimicking their actions on a new web based system. This is accomplished through a mixture of application triggers via emulator (legacy wrapper) activity and URL based redirection. "The service allows clinicians to work on the legacy platform while context-sensitive clinician content is streamed to the browser without their intervention." [35]

Another approach is the use of socio-technical design principles. While this does not dictate the manner in which user migration is performed, it ensures a stakeholder interest is created within the end user group, involving them throughout the project lifecycle from requirements gathering to testing. Using this principle, the end users obtain a sense of ownership of the final system and better alignment between both the user and work practice may be achieved.

"The implementation process of an IS (Information System) starts after the development or acquisition of an IS" and that the implementation process may be divided into two parts, firstly, the technical implementation of the IS and initial training of the users and secondly, activities to integrate the system into work practices. [36]

Given the theme of this project, it would appear that a socio-technical approach to development would enhance the prospects of user migration. Indeed, in migrating one legacy system to a common composite application platform, a consistent look and feel to user interfaces may be achieved and as such, following the migration of an initial system, subsequent system migrations will be accepted by the users more freely on the basis of familiarity. In addition, the user would have a vested interest in the new functionality added to the system, aligning the use of this functionality with their work practices where applicable.

3.3 Case Studies

This section examines the adoption of SOA by medical providers for the integration and migration from legacy systems to SOA. The case studies display the advantages gained through the adoption of SOA, namely, communication, applicability to outsource and cost reduction.

3.3.1 Adoption Example: Norwegian National Insurance Scheme

The Norwegian National Insurance Scheme supports health payments by the National Insurance Administration (NIA) to the value of over 5% of the Norwegian government budget. The Administration undertook a project to upgrade the systems architecture to a SOA model. [37]

The legacy system supporting the application was based upon EDI communications and EDIFACT messages running on the X.400 messaging protocol. Message security was built upon proprietary Public Key Infrastructure.

The major disadvantages of this architecture were:

- Messaging was performed through a central system, with no scope for interaction between client units. (Pharmacy to Pharmacy)
- The system ran on a proprietary mainframe, which was expensive to upgrade and maintain.
- The X.400 network was expensive to maintain, given the availability of cheaper TCP/IP networks within the organization.

Other key aims of the project were to support modern technology and open standards, support a larger number of connected applications and enable new services, such as electronic transmission of prescriptions. These requirements were met through the use of a SOA based system, running over TCP/IP on cheaper and more maintainable UNIX systems, implemented in Java. The use of XML messaging allows for clearer message specification and automatic validation. [38]

As a result of this adoption, doctors are now connected to other organizations in the Norwegian healthcare network in addition to the NIA. Messages are signed with personal private keys and sensitive messages are encrypted with the doctor's private keys. [38] Through use of open

standards, additional applications may be built and deployed in a rapid manner in a platform neutral architecture. Since launch, the new system has transported several million messages and payments to the value of €1.2 billion. [37]

3.3.2 Adoption Example: Los Angeles County Public Health (LAC-PH)

LA County covers an area of 4000 square miles, with over 10 million people, 114 acute care hospitals and a highly diverse population. These factors limit the county's ability to respond to public health emergencies such as BIO Terrorism.

Issues identified with LA County Public Health were:

- Limited collaboration and information sharing among public health programs.
- Unacceptable burden on reporting sources and public health partners.
- Vast duplication of effort and incomplete, untimely data.
- Limited access to critical community health information.
- Protracted policy-making process and inefficient public health response.

[39]

An action plan was devised to enable LAC-PH to enhance surveillance capabilities (both disease and terrorism based), provide web based applications to related data and systems, enhance communications between systems and develop training tools to support the new application suite.

Development Objectives of the action plan were:

- Public Health Incident Management System (IMS): Collection of web enabled tools integrating information across numerous systems.
- Healthcare Electronic Data Exchange (HEDEX): An electronic interface to collect, analyse and provide feedback on healthcare data between different healthcare providers.
- Los Angeles Immunization Network (LINK): Develop additional response based modules on existing system.
- Health Alert System Training and Education Network (HASTEN): Provide communications between health partners and response agencies.
- Public Health Data Mart: Provide large scale integration and health data analysis

[39]

As can be seen from the objectives, the theme of the requirements is communication between related agencies. In migrating to a SOA architecture, the selected vendor (CAL2CAL) was able to provide a cost effective strategy for the development of integrated components using web based components and moving from legacy systems. In addition, the ROI has been realized quickly through leveraging existing legacy systems. [39]

3.3.3 Adoption Example: Blue Cross Blue Shield of Massachusetts (BCBSMA)

In 2004, BCBSMA was required to upgrade its systems to meet business demands to integrate with a variety of network services. Issues faced included integrating with its existing infrastructure, maintaining rigid, inflexible legacy systems, and controlling extremely high development and maintenance costs.

BCBSMA introduced SOA, based upon Sun Microsystems and SeeBeyond technologies. Through the use of platform independent SOA, BCBSMA were able to outsource code development and systems integration tasks. Existing investments were realized though leveraging legacy system components. This flexibility also provides BCBSMA the ability to easily implement additional business initiatives or comply with future regulations. [40]

3.3.4 Adoption Example: UK NHS

The National Health Service (NHS) provides healthcare for 59.6 million citizens of the United Kingdom. As healthcare migrates to a shared service paradigm, the mechanism through which information is shared between various entities of the NHS has grown. [41]

To meet these changes, the world's largest data integration project was undertaken. Spanning numerous previous disparate systems, the NHS has adopted SOA to provide shared services and single patient view. At time of writing, Sun Microsystems SOA based Java Composite Application Platform (CAPS) was being delivered. In terms of scale, this SOA solution will handle 50+ million patients, 250 hospitals, 1.4m health care providers (doctors, nurses, scientists), 10,000 systems, 40,000 sites, 6 billion transactions per year by 2010, 420 messages per second with a response time of < 0.2 seconds and an available uptime of 99.9% (44 minutes downtime per month). With development underway, other providers such as ISoft Oracle and Accenture are also developing components to "plug in" to this system. [42]

In 2005, 2 billion messages were passed through the system. Improvements in patient care and saving throughout the NHS are anticipated, with some departments predicted to save 20% initially and 2% per annum thereafter. [42]

3.4 Summary

This chapter has presented the current state of the art practices for migration from legacy systems, factors considered when deciding to either rewrite a legacy system or wrap the systems components, the issue of user migration and four case studies of legacy migration / SOA adoption in the medical domain. The next chapter presents the design, development and analysis of a proof of concept application used to present the suggested value added migration strategy.

4 Proof of Concept Application

This chapter examines the design and implementation of a legacy application emulator and a composite, SOA based proof of concept application. This provides an insight from the perspective of planning and implementing an SOA solution to extend the life of a legacy system. The proof of concept application is in turn used as part of a series of workshops, performed by managers, IT professionals and medical professionals. The design of an emulated legacy system is presented, followed by a definition of the new functional requirement and the design and development of the SOA proof of concept application, including the data model, the application design, web-services definitions, a view of the alternate legacy upgrade, customisation for medical environments and finally, an analysis of the development exercise.

4.1 Application Selection

The application was selected to emulate a plausible real-world scenario, encompassing an emulated legacy system and new services running as SOA components in a distributed environment. Based on input from both ward staff and laboratory staff, an application for the retrieval of laboratory results was selected.

This application provides the following benefits:

- The application consists of multiple interfaces, only one interface set is to be updated (ward staff interface), existing interfaces (laboratory staff interface) must continue to operate as exists.
- The application is primarily data-driven. There is little business logic functionality on the legacy system.
- Ward staff use the system to retrieve data, there is no input on the test result or the patient performed through the user interface.

4.2 Emulated Legacy Application

The legacy application contains a single database table storing the test results. Medical staff interact with a simple interface for lab result retrieval.

4.2.1.1 Data Model

The data model for the legacy system consists of a single table / data source as follows:

Element	Type	Description
test_id	Number	Primary Key

test_barcode	String	Barcode of the test sample
test_status	String	Current Status of this test PENDING|INPROCESS|COMPLETE
result_id	Number	Result Identifier
result_details	String	Free text comment on test and result
result_medication_id	Number	Medication Identifier
test_request_date	Date	Date test was requested
test_result_date	Date	Date result was produced
test_requested_by	String	Name of person requesting the test

Table 4.1: Data Object for Legacy Lab Result

4.2.1.2 Application Execution

The interface used by ward staff to retrieve a test result operates as follows:

- Ward staff select the option "View Test Result".
- On the loaded screen, the test identifier is entered.
- The result details are then displayed.

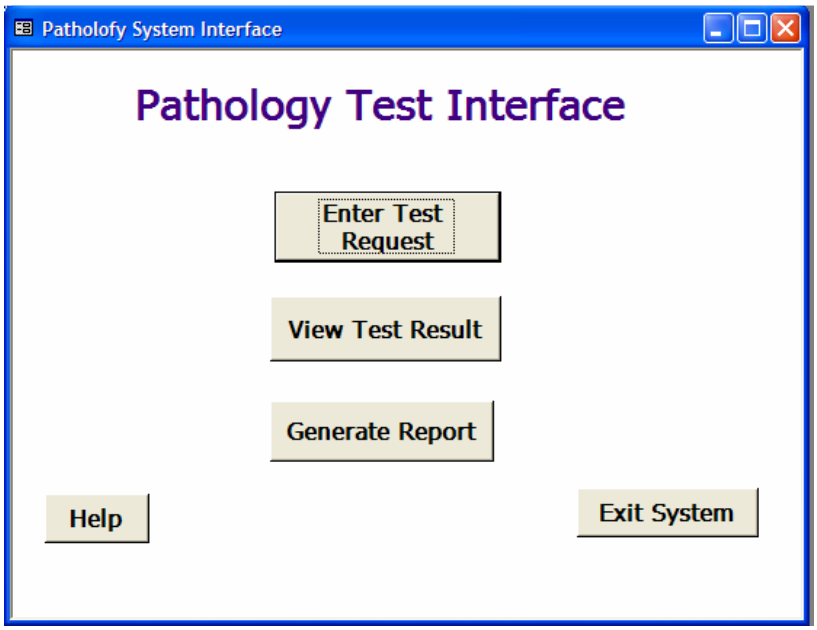


Figure 4.1: Lab Result Interface – Menu

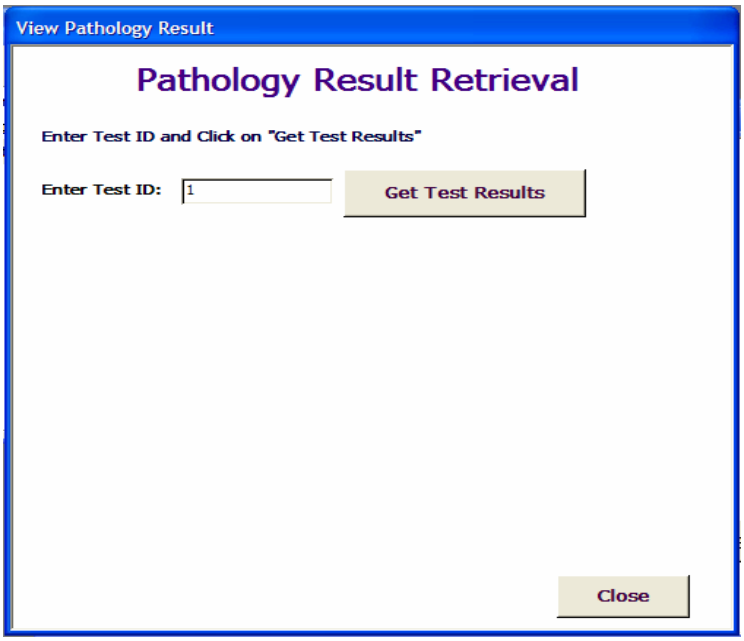


Figure 4.2: Lab Result Interface – Enter Result ID

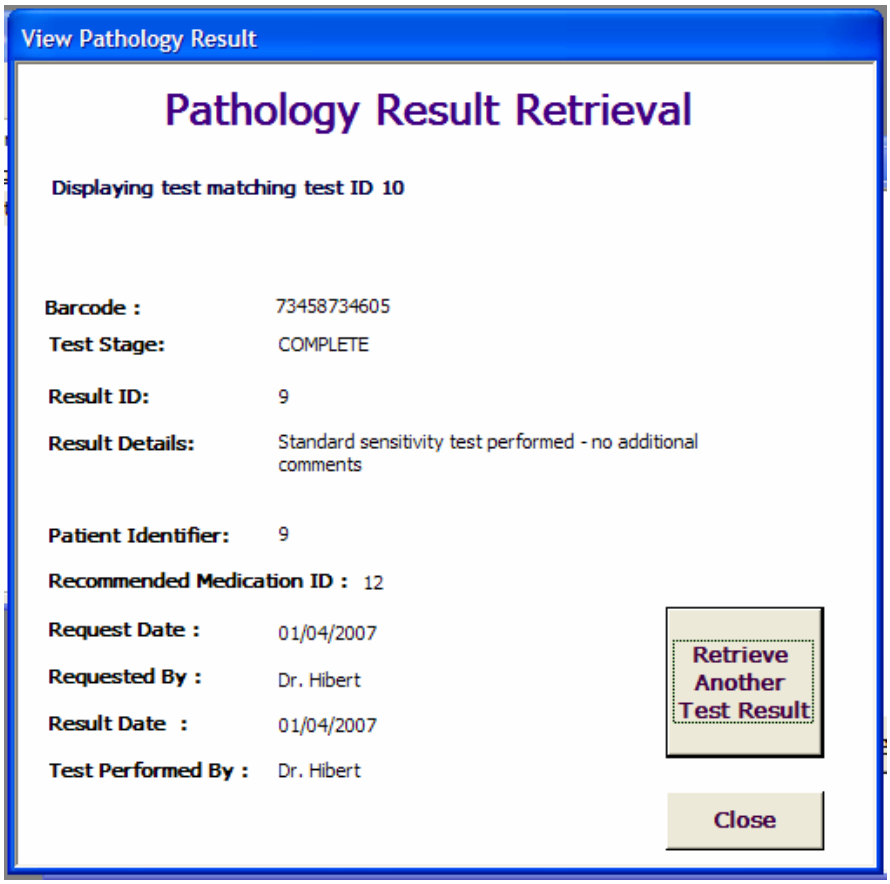


Figure 4.3: Lab Result Interface – Enter Result ID

4.3 New Function Requirements

The organisations requirement for a system upgrade is driven by both patient care quality and ward efficiency initiatives. Through discussion with ward, laboratory and management staff, the following new requirements were devised:

"Upon the retrieval of a pathology result, the recommended medication shall be checked against the patients' medication history, biological profile and current medication listing for contra-indicators."

Upon review, this requirement may be refined as:

- Check the recommended medication against the patient's recent medication history.
- Check the recommended medication against the patient's recent current medications.
- Check the recommended medication against the patient's biological profile.

Such checking will reduce "file pull" times and reduce risk of inappropriate medication prescriptions. Given the technology and current infrastructure, the legacy system does not have access to the required data. Upgrading the legacy system or replacing the system may not be desirable options and the organisation may be reviewing other technologies to assist in data integration. Against this backdrop, the application of new business rules as composite applications would meet the overall strategic plan. As the SOA solution involved wrapping the database, the lab staff can continue to use the legacy application until it is appropriate to update, inline with organisational plans and budgets.

4.4 SOA Composite Application

As defined, there are three services required which are external to the legacy system, these services require both access to the patients data and other support systems, such as decision support system to validate medication mixes. Through use of SOA, a composite application can meet the requirements of live system integration, with current systems wrapped in a web-services layer.

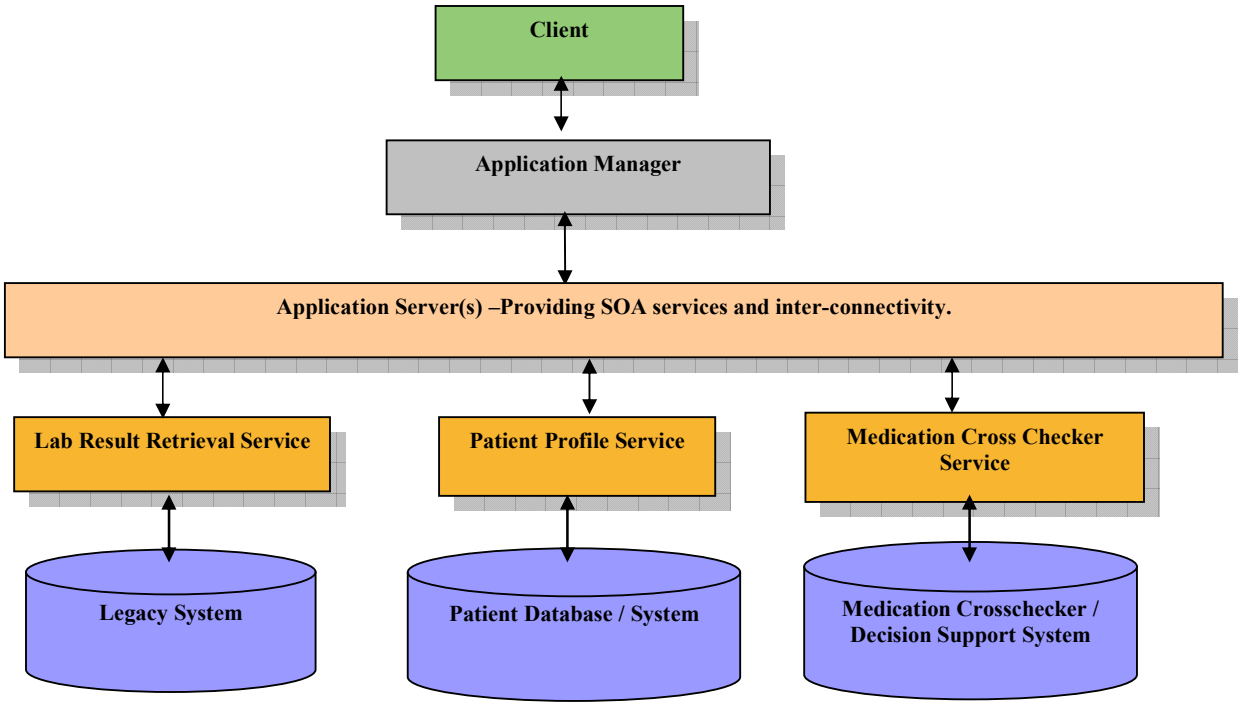


Figure 4.4: Composite Application Components

4.4.1 Data Model

The following diagram represent the entity relation of the data used in the sample application.

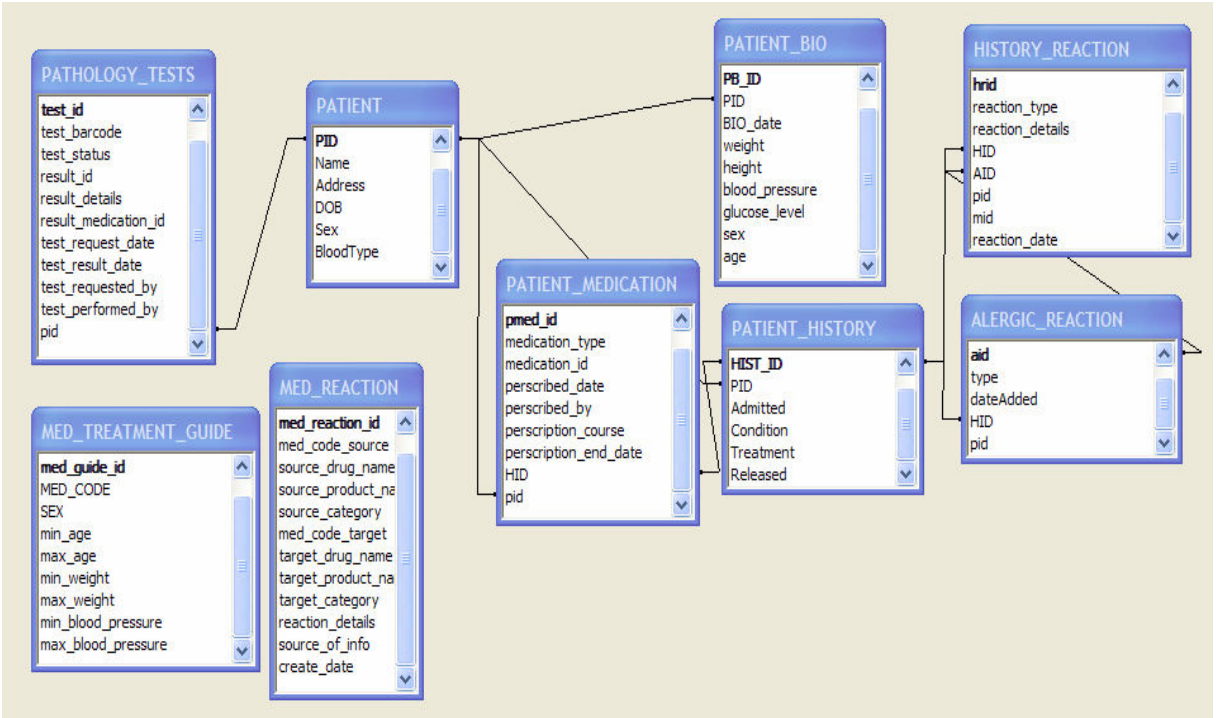


Figure 4.5: Composite Application Data Model

The tables listed may be physically divided as follows:

- PATHOLOGY_TESTS: Pathology system table, located on legacy system.
- MED_REACTION and MED_TREATMENT_GUIDE: Medical decision support table, located on a decision support system or database.
- All other tables: Patient data tables, located on another system.

4.4.2 Application Design

The SOA application is based on an N-Tier model, implemented using EJB version 3 and industry standard patterns such as the business delegate and session façade patterns, with the presentation layer developed using a Model, View, Controller (MVC) pattern.

There are two types of services identified in the design of the application:

- Data Wrapping Service
- Business Logic Service

As the application is primarily data driven, the core of the application are the data wrapping services. Each table on the legacy database is represented by a container managed entity bean. The bean performs mapping between the java objects and defines SQL interfacing.

A value object is also created to represent the entity data to be used throughout the application. This is a light weight serializable object. The Entity Bean contains a method to create and return the value object representing its data. Similarly, methods may be added to the Entity Bean to reset the data in the database based upon the contents of the value object. The data is exposed on a domain level via the use of business delegate (data delegate). The delegate, a session bean acts as a single point of contact for the lower data layers, retrieving and updating the legacy data through the use of the appropriate value objects. At the top of the access stack sits a web service interaction class. This class performs marshalling and un-marshalling of XML data, interacting with the data through the delegate bean.

Figure 4.6 outlines the application stack in use for the retrieval of a medication bio mix match, as used by the application manager.

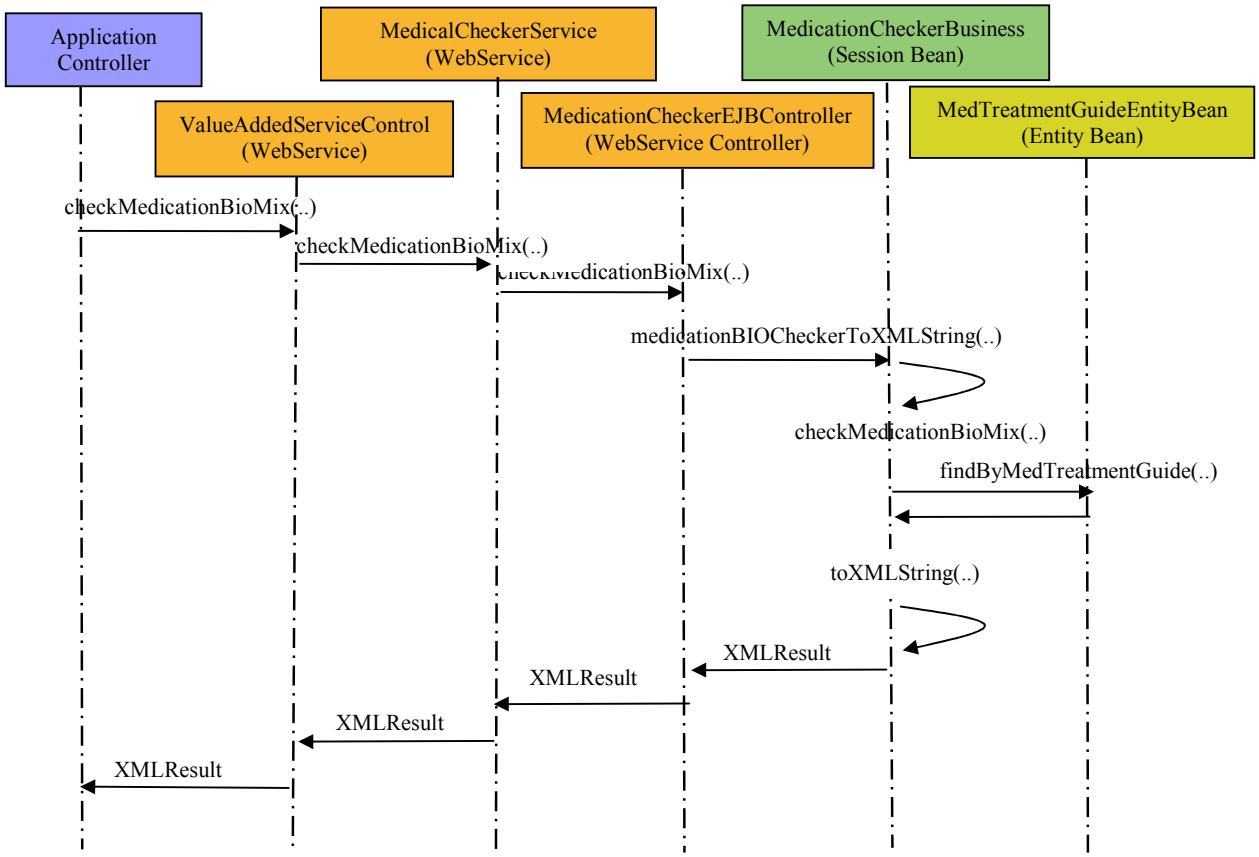


Figure 4.6: Sequence of calls invoked in Medication Biological Mix Check.

For details on sample classes and methods (Medication Checker Servlet and Med Treatment Guide Entity Bean), refer to appendix C: Javadoc.

Figure 4.7 outlines the application stack in use for the retrieval of data from the emulated legacy system and wrapping of this data as the result of a web-service call.

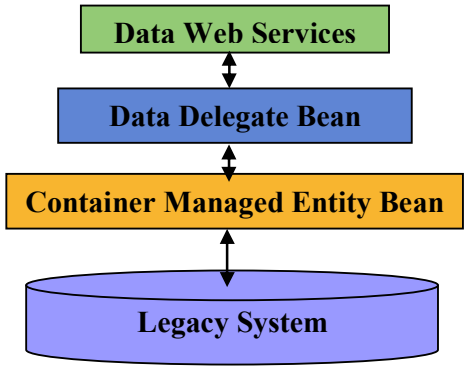


Figure 4.7: Data Service Application Stack

The business process of retrieving a lab result, complete with value added services is defined as follows:

- The application manager makes a call on the Value Added Service.
- The Value Added Service calls the Lab System Wrapper Service, the Patient Profile Service and the Medication Cross Checker Service to retrieve their specific information sets. These web-service calls include the un-marshalling of request data, interaction with the legacy system via the value object and entity bean and the marshalling of the result as an xml response.
- These results are combined in a single xml document. The composite result is now a combination of the lab result data returned from the legacy system and “advice” on the prescription of the suggested medication based up factors such as the current or recent medication prescribed to the patient, the patients history and biometric data.

The following sequence diagram displays the interaction of services in the retrieval of a laboratory result, including the new value added services:

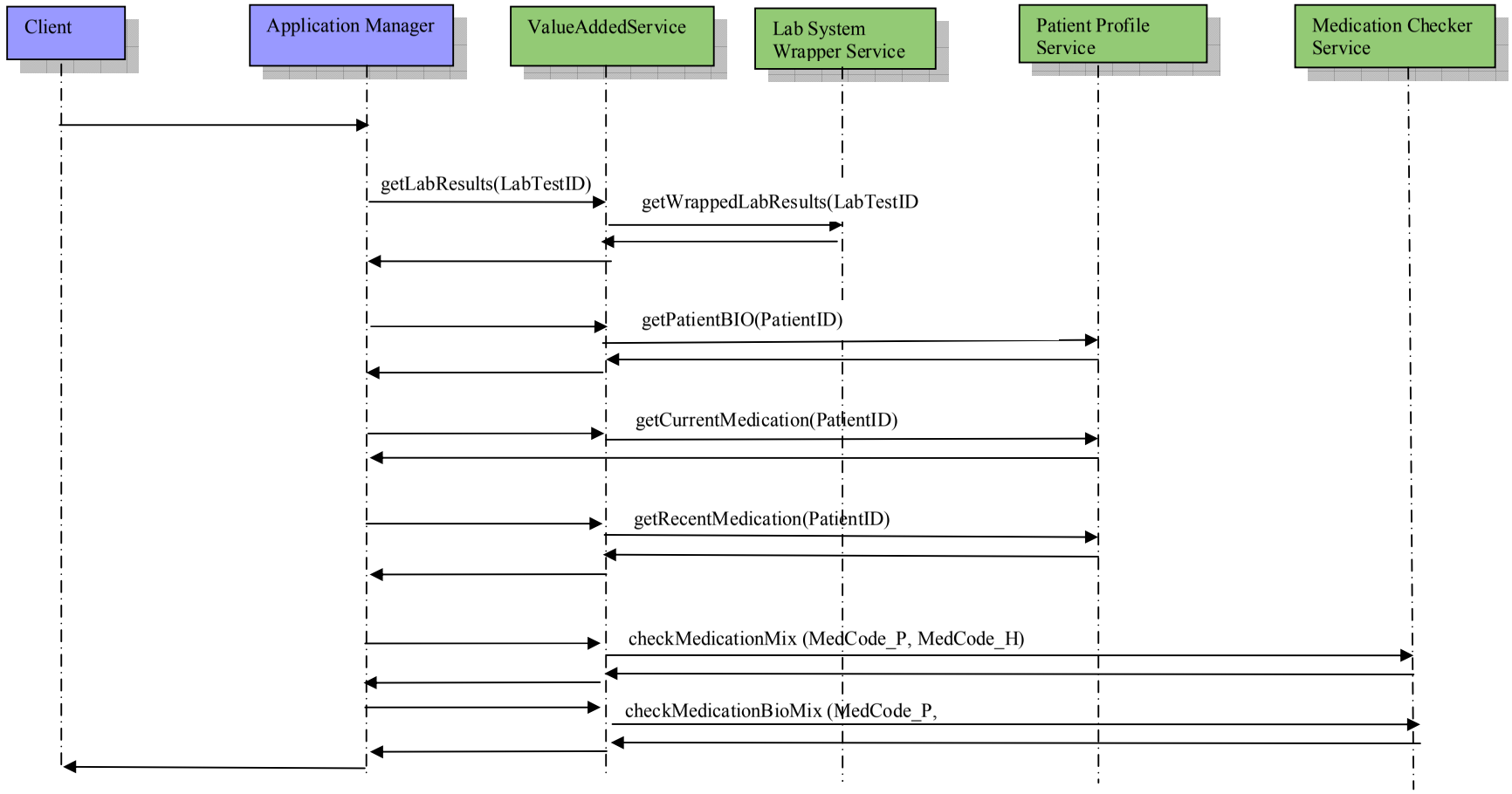


Figure 4.8: Service Interaction to Retrieve Value Lab Result and Value Added Data.

4.4.2.1 Interacting with the Legacy System

It should be noted that while there are many methods for interacting with legacy systems, the mechanism chosen for this project is via industry standard database connectivity. JDBC connectivity is suited to the SOA model in that support is guaranteed across application servers meeting J2EE and .net standards and highly configurable in terms of reliability and performance.

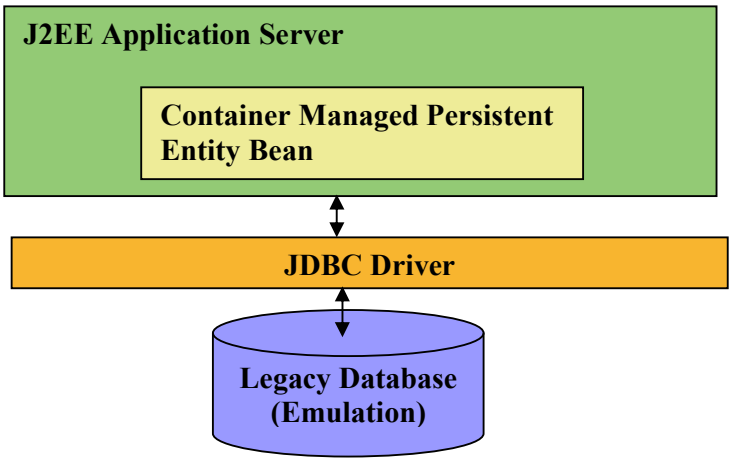


Figure 4.9: Interaction with a Legacy System (Database)

4.4.3 Web-Service Definitions

Each disparate system taking part in the composite application is interfaced via web-services.

These web services are defined as follows:

4.4.3.1 Lab Results Retrieval Service

This service is called by the application manager to retrieve the lab result and value added service data. The total result is returned to the application manager as an XML document. Refer to appendix B for the full Web Service Description Language (WSDL).

- **Web Service Name**: getLabResults

  Inputs: Test ID

  Returns: Lab Result with value added data.

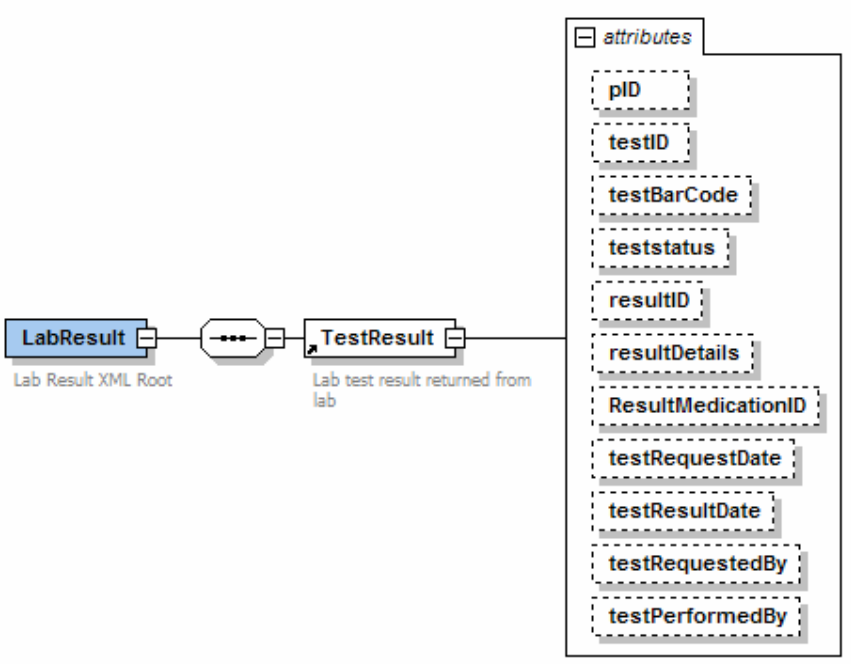


Figure 4.10: Lab Results XSD

4.4.3.2 Patient Profile Service

This service provides capability to retrieve information on the patient, their history and particular patient orientated functions applicable to this application. Refer to appendix C for the full Web Service Description Language (WSDL).

- **Web Service Name:** getPatientBio

  Inputs: Patient identifier.

  Returns: Patients biological information.

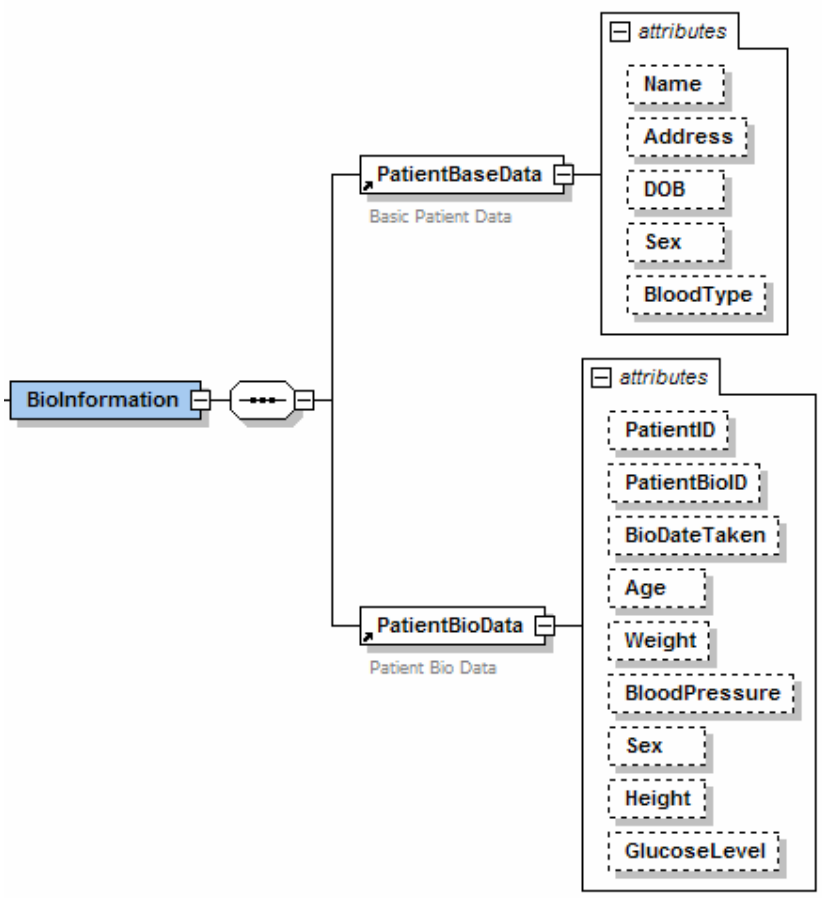


Figure 4.11: Patient Profile XSD

- **Web Service Name:** getCurrentMedication

  Inputs: Patient identifier.

  Returns: List of medication patient is currently taking.

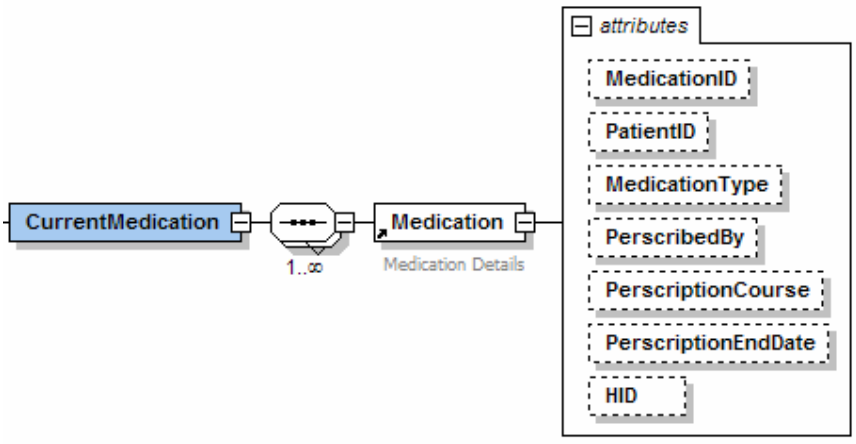


Figure 4.12: Lab Results XSD

- **Web Service Name:** getRecentMedication

  Inputs: Patient identifier, Medication Code.

  Returns: List of reactions in the patients' history to the supplied medication code.

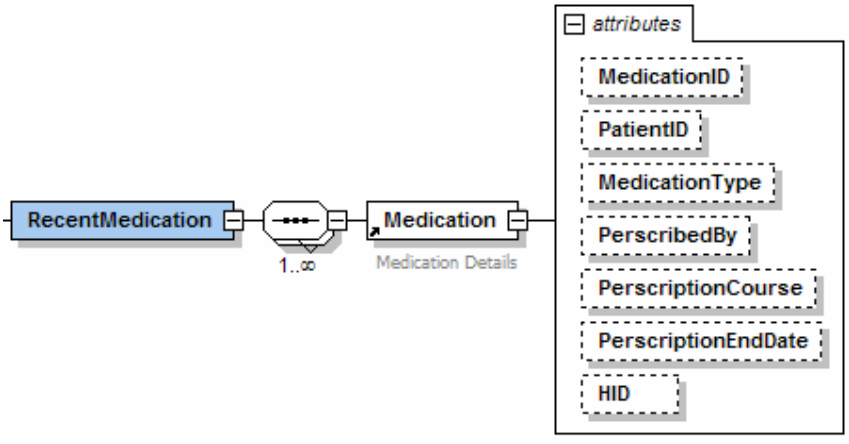


Figure 4.13: Recent Medication XSD

4.4.3.3 Medication Cross Checker Service

This service provides services to analyse combinations of medication for know reactions. Refer to appendix C for the full Web Service Description Language (WSDL).

- Web Service Name: checkMedicationMix

  Inputs: Suggested Medication Code, Currently or Recently Prescribed Medication Code

  Returns: Reaction details to mixing the medications.

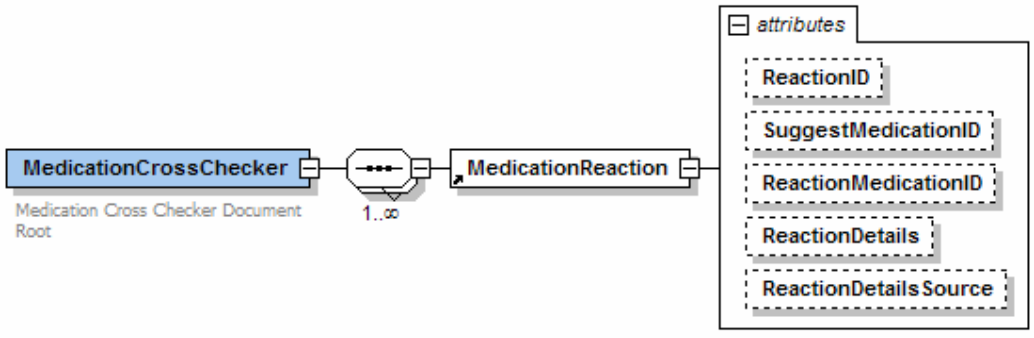


Figure 4.14: Medication Mix XSD

- Web Service Name: checkMedicationBIOMix

  Inputs: Suggested Medication Code, Patient Bio Data

  Returns: Biological profile details matching the patients' data and medication recommendations.

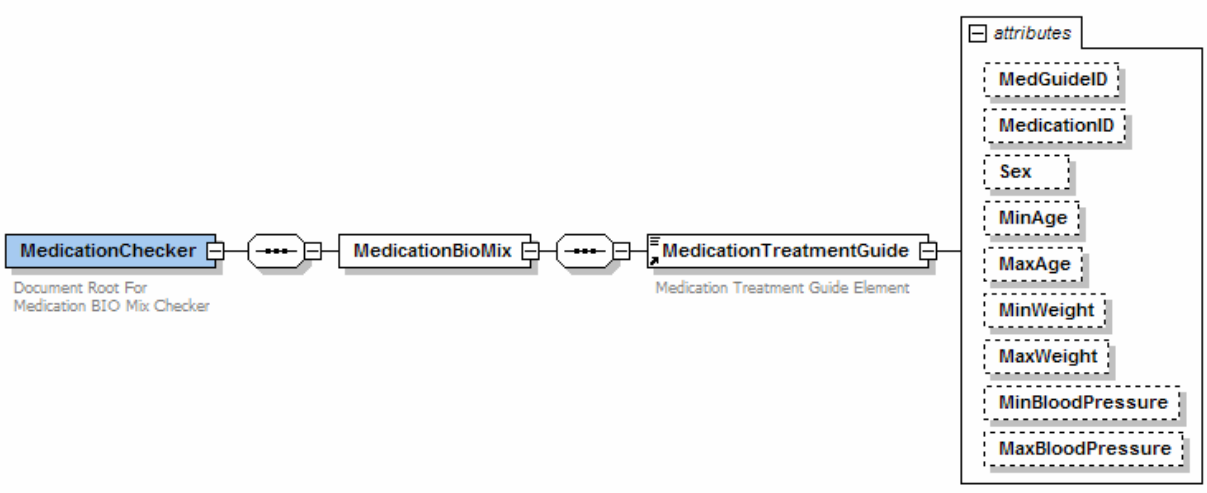


Figure 4.15: Medication Biological Mix XSD

4.4.4 Alternative Legacy System Upgrade Design

In contrast to an SOA solution, the development of a solution based on upgrading the legacy system may involve the interfacing of the legacy system with other systems, such a medication cross checking system or the patient database. Similarly, the business rules defined in the SOA application manager would be implemented in the legacy system.

Such a development may require the following:

- Development of interface specification and business rules and between the legacy and external system.
- Extension of legacy application to integrate new business rules.
- Updating of legacy data to meet new data requirements.
- Depending on the legacy architecture – primarily data driven or data driven in addition to business rules, an addition layer of business logic may be required to integrate legacy data, external data and new business rules.

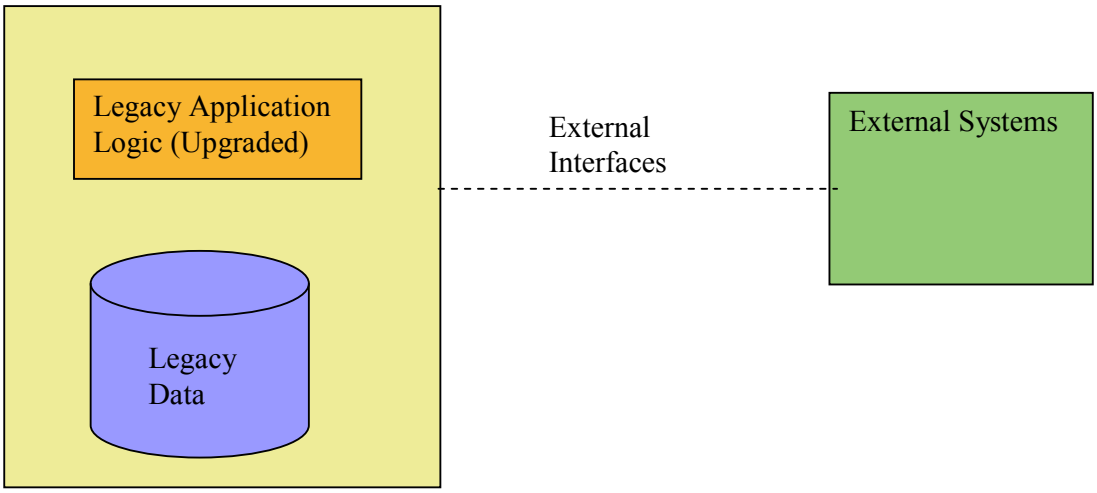


Figure 4.16: Legacy Application Development

Based on the definition of the legacy system, interfacing with other systems may prove difficult, in-particular if the legacy system is treated as a black box. It is also unlikely that real-time interfacing with other systems is possible without either significant reworking of legacy components, or the integration of other middleware technologies, for example TIBCO's information bus. [43]

4.4.5 Generic Model, Customised for Medical Environments

The proof of concept application serves provides web-services with a custom set of XML responses. The schemas for these, as documented in section 4.3.3 has been designed to service a specific function, meeting the requirements outlined by the organisation. In view of organisational growth and interaction between different medical bodies, it is possible that the web-services developed may be required to serve other organisations and other systems.

Given the business case selected for the proof of concept application, an extension may be in the cooperation of laboratories or the outsourcing of a specific laboratory function to a third party provided. In either case, the external group may require access to the laboratory information system while operating its own proprietary technology. As such, the XML conversation held between organisations is likely to operate on different schemas and different vocabularies, in the case of medication codes and condition codes. (Examples HL7, ICD9, SNOWMED)

To facilitate this scenario, it is possible to utilise EXtensible Stylesheet Language Transformations (XSLT). This XML/XSL based technology acts as a translator, translating a supplied XML string based on user defined criteria. Thus, instead of the redevelopment of an entire web-service to accommodate integration with another organisation, the in-house web-service response may be

parsed and translated before returning to the client, minimising the integration tasks to the development of an XSLT translator and a form of proxy service to redirect the request in order to pass through the XSLT engine.(Figure 4.17) In the absence of both the organisation and the new client system developing their schemas in cooperation, this solution minimises effort for both parties.

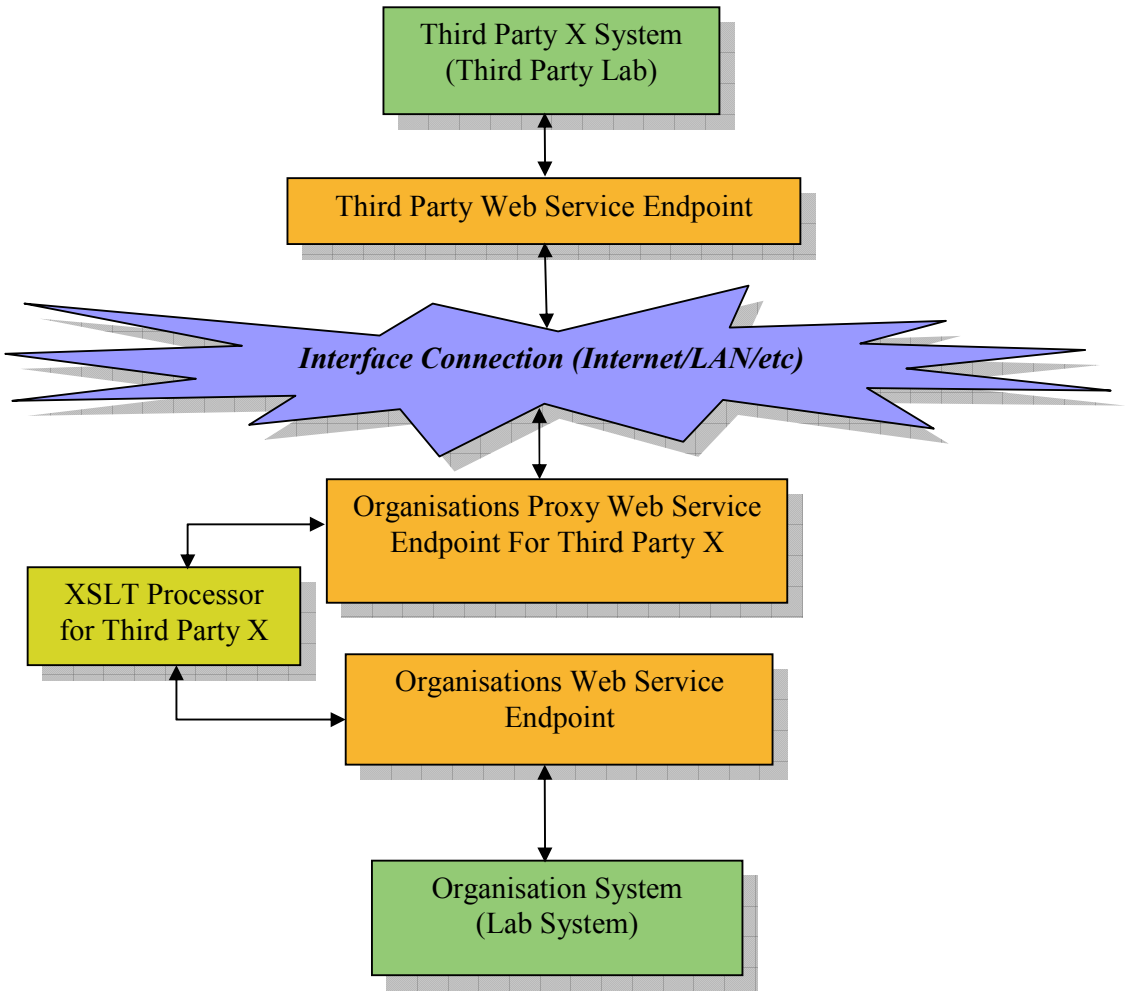


Figure 4.17: Cooperating with new Third Party Systems

It should be noted that the XSLT processor could be integrated by either the third party client or the organisation. Indeed, in this case it is arguable that the translator should be developed by the client as the client receives the benefit from interconnecting with the organisations system. However, such decisions may be based on the managerial view of the relationship between the organisation and third party, the operational scenario and other contractual factors.

4.4.6 Development Analysis

4.4.6.1 Development Experience

There were no major issues encountered during the development of the proof of concept application. The use of industry standard patterns and tools such as eclipse, and the deployment into a popular application container (BEA Weblogic) lent to a smooth development process as there were numerous tuorials and high quality documentation available. The functionality of each layer of the application stack was tested through he development of specific test applications. In using this approach, coding issues were resolved quickly and few unexpected exceptions resulted when performing the final web-service and user interface integration. The implemenation of web-services was aided by a graphical web-services tool within the eclipse/weblogic workshop environment. The development of the user interface layer was performed using the Apache Beehive framework[44] and toolkit which is based on the Apache Struts framework[45]. Based upon the development process experienced, it may be stated that the development of a web-services tier is a simple extension on any J2EE application, as code compoents and business logic is simply wrapped with a web-service controller and interface.

The majority of issues encountered in this development were environment orientated. Specifically, the use of JDBC drivers to wrap the proof of concept legacy system proved a problem. To aid in rapid development, Microsoft Access was chosen to act as the emulated legacy system. Initial attempts were made to use the JDBC-ODBC bridge driver; however this driver is not supported under weblogic for the development on container manager entity beans. The use of supported drivers led to licensing issues, no non-commercial drivers were available and preview drivers held restrictions on the number of queries which may be run with the driver or an expiry timeout set on the driver. To resolve these issues, the legacy data model was ported to a pointbase database, which was shipped with Weblogic. Following the migration of the data, and the reconfiguration of the data sources within weblogic, the proof of concept application worked first time, with no further complications.

4.4.6.2 Phasing out the Legacy System over time

In lending to the long term migration to an SOA application, the development of the proof of concept application as an SOA solution provides several core components for the further migration towards SOA. Upon development, each disparate system was wrapped in a web-service layer for the exposure of business methods and each table required for these business services was individually wrapped as an entity bean. Ignoring the development of the specific business rules for the calling of individual services and the merging of the returned data as a composite result to the application manager, the majority of the design and development work (80%) was invested in the

development of entity beans, session beans and web-service end points. Thus the development of new composite applications or enhancement of current services would require significantly less effort than the proof of concept application, as the core wrappers and services are already in place. This initial investment therefore saves effort in the longer term, as the organisation migrates each legacy system to SOA, the effort in each migration decreases, eventually reducing the effort to the development of the core business rules and minimal addition of table wrappers which have not been encountered in the previous SOA developments.

4.5 Summary

This chapter has presented the design, development and analysis of a proof of concept application, used to present the suggested value added migration strategy. The design of the SOA application has provided details on the core application design, data model supporting the application and web-service definitions. The analysis has provided an account of the development experience and its relevance to the dissertations migration strategy.

The next chapter, performance testing makes use of the proof of concept application to perform and analyse a series of tests investigating the application throughput capacity and scalability in both a simple and clustered environment configuration. The chapter opens with a definition of the required performance, followed by the design of the performance tests, testing parameters, details on the test environment, the execution and results of the tests and concludes with an analysis of the results.

5 Performance Testing

This chapter details the requirement, creation, execution and analysis of a series of perfomance tests. The use of performance testing provides an insight into the performance capacity of the proof of concept application and through comparison of a single instance and cluster performance test, the scalability of the SOA application. The chapter is structured to outline the quantification of performace test parameters, the design of the tests applied, the environment in which the tests were executed, the results of the tests and finally, an analysis of the test results.

5.1 Required Performance

In designing a performance test for the SOA application, factors such as the hardware, software, application server and network connectivity must be taken into account. From the organisations perspective, performance testing will assist in the evaluation of SOA and the migration approach through providing accurate statistic's of the systems capacity. This in-turn will dictate the hardware requirements, licensing costs, support costs and availability of the system.

Given the nature of the sample application, the following load factors were identified:

Input from pathology lab staff, ward staff, IT management and medical management indicates:

- Maximum Normal Usage: 5 requests every 10 seconds.
- Usage should other services be introduced, such as report generation: 10 requests every 5 seconds.

To establish maximum usage, the system should be tested with load until its point of failure.

Placing this into a test plan, the following tests should be performed:

Test Plan 1: System Requirements Testing

- 5 requests every 10 seconds.
- 10 requests every 5 seconds.

Test Plan 2: System Stress Testing

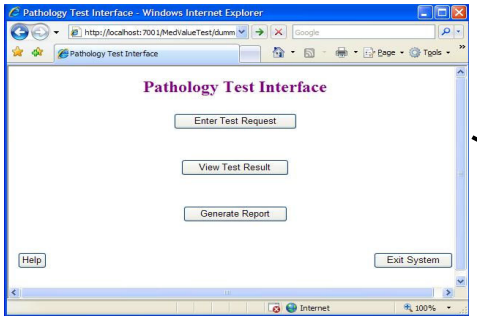
- Test 1: 10 User Requests on a 5 second period, repeated 10 times.
- Test 2: 25 User Requests on a 5 second period, repeated 10 times.
- Test 3: 50 User Requests on a 5 second period, repeated 10 times.
- Test 4: 75 User Requests on a 5 second period, repeated 10 times.

- Test 5: 100 User Requests on a 5 second period, repeated 10 times.

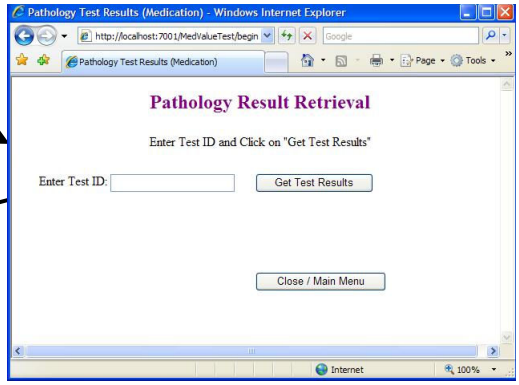
5.2 Performance Test Design

Performance testing was carried out using industry standard tools such as weblogic application servers as the application container and Jmeter (www.jmeter.org) as the load test emulator. Jmeter is an open source tool originally developed to test apache's webserver; however its success led to adoption to test other applications as it provides features on parallel with commercial products. [46] [47-49] The base Jmeter test plan executed is listed as Appendix J. Each test consisted of the following set of activities, replicating a user interaction with the system:

1. User launches the Menu Screen.



2. User accesses the Pathology Lab Result Retrieval Page



3. User Retrieves the Result for a given Test ID

4. User Returns to the Menu Screen

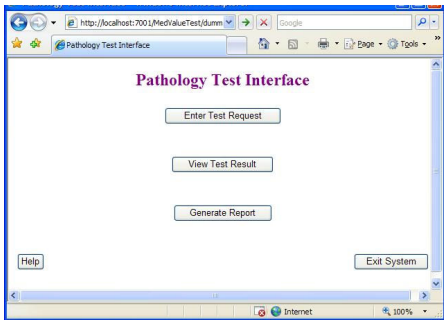


Figure 5.1: Jmeter Test Process

5.3 Test Parameters

A sample of 2500 unique cases are used to perform the required testing, with a maximum case use of 12000 samples. Each sample consists of a unique record on each table for each data-source interconnected to fulfil the test result request.

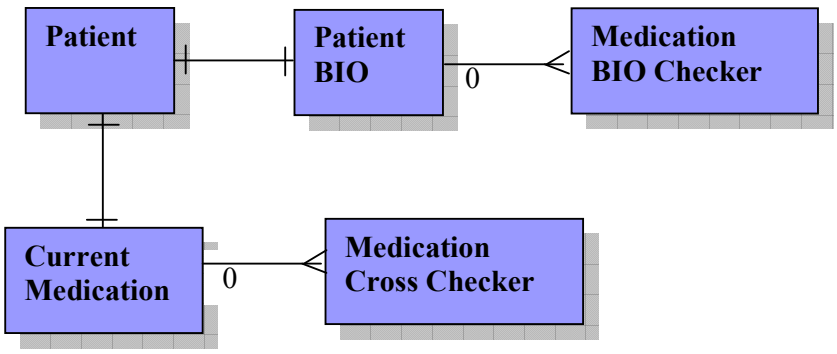


Figure 5.2: Performance Data Sample Data

5.4 Execution Environment

To perform both functional load and distributed load testing of the sample application and taking the outlined factors into account, two unique configurations are to be defined:

5.4.1 Functional / Proof of Concept Testing On Single Machine

A proof of concept environment was created to replicate a plausible development / system testing environment. In addition, this environment may represent a production environment on a small scale. It is feasible that the legacy application would run on the same physical environment as the data source, indeed, it is feasible that the legacy application and the data source are inseparable. “Mainframe systems tend to be monolithic and provide no immediate or easy way of identifying appropriate services.”[50]

Two environments were configured and tested, the first with calls from the test harness (Jmeter) directly to the application server, the second, with calls from the test harness to the application server via a load balancer. As a load balancer is used in the clustered environment, the use of a load balancer in this context was deemed necessary for a more accurate comparison of results.

The proof of concept environment(s) were configured as follows:

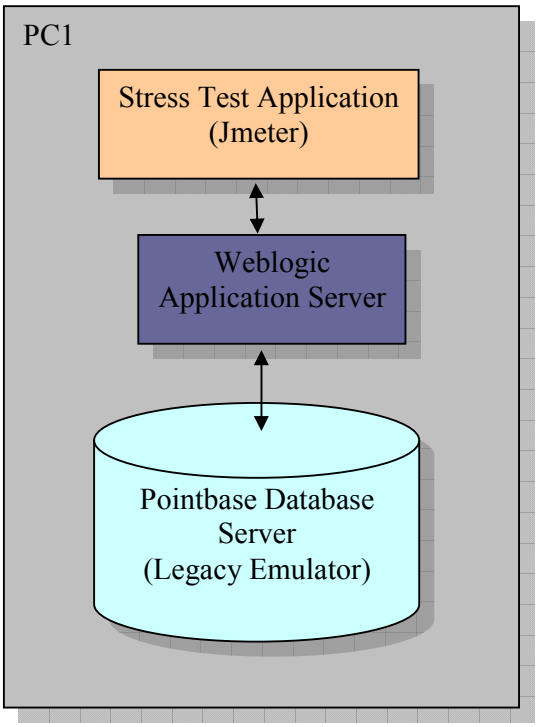


Figure 5.3: Simple Configuration

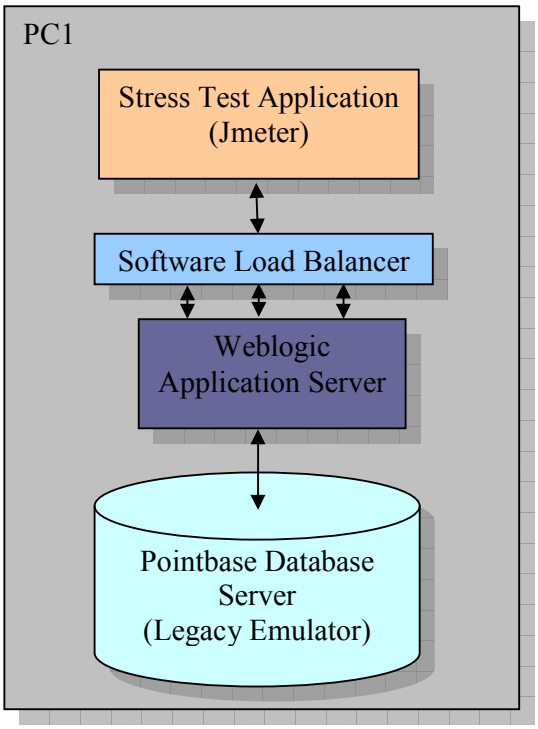


Figure 5.4: Load Balanced Configuration

5.4.1.1 Environment Details

The following hardware and software was used to carry out this stress test:

Hardware:

- Dual Core AMD Turion 64 bit Processor
- 512 megabytes of Ram
- 30 Gigabyte hard drive

Software:

- Weblogic 9.2 Application Server
- Point-base Database Server (Acting as a legacy database system)
- Point-base Database Server (Acting as a new database system storing distributed systems data patient data, medical decision support system etc)
- Jmeter – Stress testing application.
- Radiux Central Director – Open source software load balancer. (www.sourceforge.net)

Network:

The network configuration is changed on the mode of testing to be performed:

- All components on local machine, no network overhead.

5.4.2 Functional / Proof of Concept Testing in a Clustered Environment

An application cluster environment was created to replicate a real world scenario involving the distribution of application load and the capacity of the SOA solution to be seamlessly upgraded. In terms of capacity, the processing power of the application should be increased by n%, where n=100/total number of application servers. It should be noted that a separate exercise would be carried out to determine the number of application servers which can be run on any one physical machine, without degrading performance. For the purpose of this project, the simple configuration of one application server per physical machine was used.

Another consideration in stress testing the application server on a clustered environment is the legacy data source. With the continual expansion of the application server layer, bottlenecks may appear on the wrapper around the legacy system or indeed on the throughput which can be provided on the legacy system itself. Load testing with a cluster configuration will allow for the determination of such bottlenecks, providing an indication of the level of throughput which can be tolerated before the legacy layer wrapper or legacy system required a resource upgrade or final migration to a more powerful platform.

The clustered environment used in this test consisted of three computers. Each computer ran a copy of the application server. Computer no. 2 also ran the emulation database server, containing the legacy system data and, through a separate data source, the other data resources (patient data, medication cross checking data etc). In addition, computer no. 2 ran both Jmeter, to instigate client requests and a software load balancer, which directed requests from Jmeter to any one computer in the group on a round-robin basis.

The proof of concept cluster environment was configured as follows:

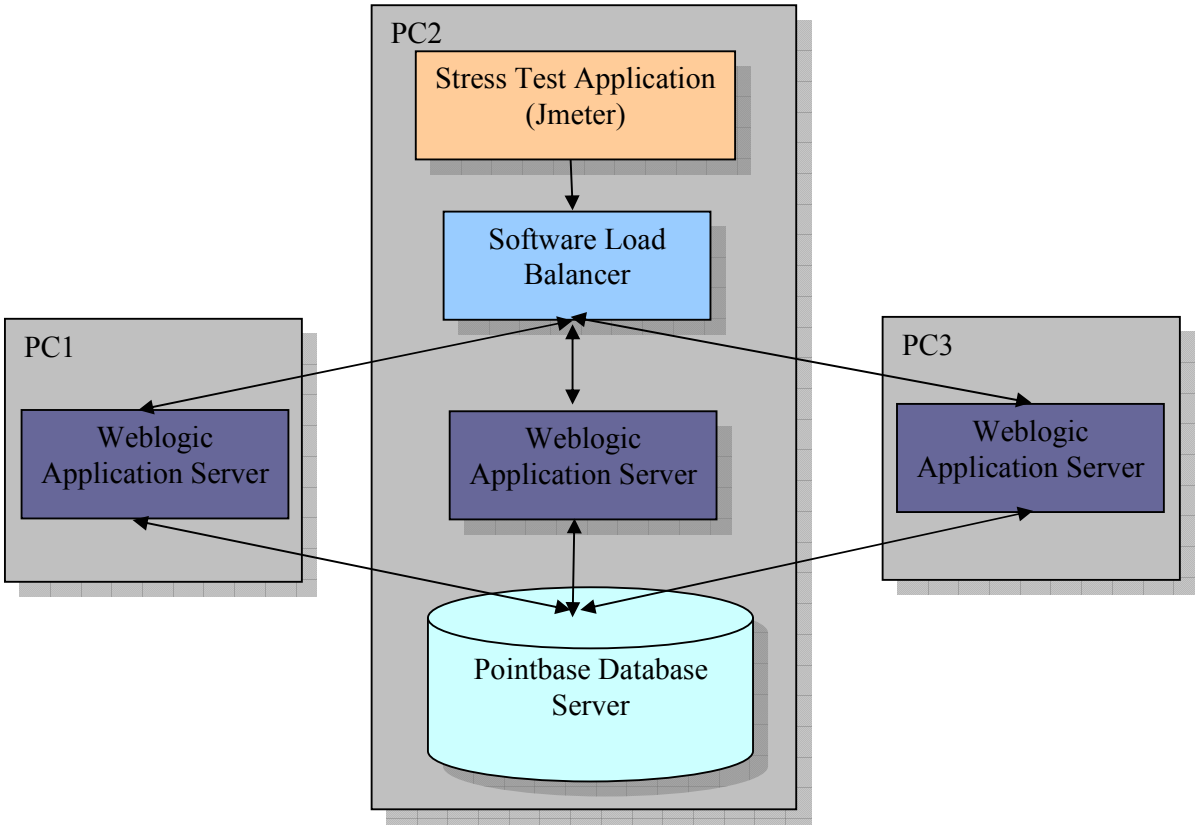


Figure 5.5: Cluster Environment Load Testing Configuration

5.4.2.1 Environment Details

Hardware: Cluster of 3 machines, each specified:

- Pentium 4 Processors
- 512 megabytes of Ram
- 30 Gigabyte hard drive

Software:

- Weblogic 9.2 Application Server
- Point-base Database Server (Acting as a legacy database system)
- Point-base Database Server (Acting as a new database system storing distributed systems data patient data, medical decision support system etc)
- Jmeter – Stress testing application.
- Radiux Central Director – Open source software load balancer. (www.sourceforge.net)

Network:

The network configuration is changed on the mode of testing to be performed:

- All components are accessible on a LAN, with interfaces running at 100MBps.

5.5 Performance Test Execution & Results

5.5.1 Proof of Concept / Functional Testing on a Single Machine

5.5.1.1 Test Series 1: System Requirements Testing

In terms of system performance requirements, the following performance was measured:

5.5.1.1.1 Test 1: Maximum System Usage Expectation Testing:

5 individual users every 10 second, this cycle was repeated 100 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	500	168	125	281	15	1344	0.00%	2.3/sec	4.85
Enter Result ID	500	168	125	266	16	1391	0.00%	2.3/sec	6.04
View Result	500	1635	1422	2265	250	12438	0.00%	2.3/sec	9.38
Return to Menu	500	171	125	281	15	1453	0.00%	2.3/sec	4.88
TOTAL	2000	535	156	1578	15	12438	0.00%	9.0/sec	25.07

Table 5.1: Simple Environment, Maximum Expected Users

5.5.1.1.2 Test 2 Double Maximum System Usage Expectation Testing:

10 User Requests on a 5 second period, repeated 100 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	1000	299	156	922	15	3578	0.00%	1.8/sec	3.90
Enter Result ID	1000	296	157	813	15	24500	0.00%	1.8/sec	4.86
View Result	1000	4449	3922	5688	235	36765	0.00%	1.8/sec	7.56
Return to Menu	1000	338	188	1000	15	2844	0.00%	1.9/sec	4.17
TOTAL	4000	1346	250	4156	15	36765	0.00%	7.3/sec	20.20

Table 5.2: Simple Environment, Double Maximum Expected Users

5.5.1.1.3 Test Series 2: System Stress Testing

In terms of system stress testing, tests were carried out through connecting the test harness (Jmeter) directly to the application server and through the use of an intermediate load balancer. In this

configuration, the load balancer diverts all requests to the local application server. The use of a load balancer was deemed necessary to provide a balanced set of results for comparison with the clustered test results.

5.5.1.1.4 Test 1: 10 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	100	390	235	1219	32	2422	0.00%	1.8/sec	3.95
Enter Result ID	100	270	203	469	31	2156	0.00%	1.9/sec	5.11
View Result	100	3663	3188	7251	516	11767	0.00%	1.9/sec	7.87
Return to Menu	100	459	235	1562	31	2125	0.00%	2.3/sec	4.81
TOTAL	400	1196	297	3516	31	11767	0.00%	7.2/sec	20.01

Table 5.3: Simple Environment, 10 Requests

* Results achieved via use of load balanced configuration

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	100	119	63	218	15	906	0.00%	59.0/min	2.10
Enter Result ID	100	198	93	656	15	1485	0.00%	59.9/min	2.66
View Result	100	2059	891	10031	265	13750	0.00%	59.2/min	4.08
Return to Menu	100	206	78	890	15	1328	0.00%	1.1/sec	2.43
TOTAL	400	645	109	1250	15	13750	0.00%	3.7/sec	10.16

Table 5.4: Simple Environment with Cluster, 10 Requests

5.5.1.1.5 Test 3: 25 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	250	1689	1031	3766	31	15204	0.00%	1.5/sec	3.16
Enter Result ID	250	1554	969	2906	15	15141	0.00%	1.5/sec	3.94
View Result	250	7948	7720	13907	391	17938	0.00%	1.5/sec	6.11
Return to Menu	250	1608	1062	3438	31	13766	0.00%	1.5/sec	3.18
TOTAL	1000	3200	1875	8610	15	17938	0.00%	5.9/sec	16.32

Table 5.5: Simple Environment, 25 Requests

* Results achieved via use of load balanced configuration

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	250	711	500	1563	16	3172	0.00%	2.3/sec	4.90
Enter Result ID	250	889	656	1906	31	8531	0.00%	2.3/sec	6.09
View Result	250	7831	7453	12016	312	17937	0.00%	2.3/sec	9.43
Return to Menu	250	846	547	1672	32	9875	0.00%	2.3/sec	4.89
TOTAL	1000	2569	891	7922	16	17937	0.00%	9.1/sec	25.20

Table 5.5: Simple Environment with Cluster, 25 Requests

5.5.1.1.6 Test 3: 50 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	500	2379	2297	4594	15	14579	0.00%	1.9/sec	4.10
Enter Result ID	500	2480	2063	5187	16	17735	0.00%	1.9/sec	5.09
View Result	500	15592	15547	22173	969	30220	0.00%	1.9/sec	7.89
Return to Menu	500	2468	2203	4656	15	13439	0.00%	2.0/sec	4.28
TOTAL	2000	5730	2984	16688	15	30220	0.00%	7.6/sec	21.02

Table 5.6: Simple Environment, 50 Requests

* Results achieved via use of load balanced configuration

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	500	4196	4031	6579	31	16109	0.00%	2.2/sec	4.74
Enter Result ID	500	3988	3907	6328	31	9375	0.00%	2.2/sec	5.77
View Result	500	12103	9594	17266	2735	61782	0.00%	1.8/sec	7.66
Return to Menu	500	4067	4125	6328	32	14422	0.00%	1.9/sec	3.99
TOTAL	2000	6088	4797	10719	31	61782	0.00%	7.4/sec	20.46

Table 5.7: Simple Environment with Cluster, 50 Requests

5.5.1.1.7 Test 4: 75 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	750	4462	4547	7516	31	29236	0.00%	1.4/sec	2.94
Enter Result ID	750	4463	4438	7376	31	26814	0.00%	1.4/sec	3.67
View Result	750	34036	31596	75865	2672	97459	0.00%	1.4/sec	5.68
Return to Menu	750	5459	4563	9501	15	29174	0.00%	1.6/sec	3.50

TOTAL	3000	12105	5344	35158	15	97459	0.00%	5.4/sec	15.04

Table 5.8: Simple Environment, 75 Requests

* Results achieved via use of load balanced configuration

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	295	744	94	984	15	25500	12.20%	2.3/sec	4.64
Enter Result ID	277	2222	109	1000	15	30000	10.47%	2.1/sec	5.41
View Result	264	4189	938	24328	250	31203	17.05%	2.0/sec	7.47
Return to Menu	257	810	94	3016	15	7766	12.45%	2.4/sec	4.97
TOTAL	1093	1966	204	3219	15	31203	12.99%	8.0/sec	20.51

Table 5.9: Simple Environment with Cluster, 75 Requests

5.5.1.1.8 Test 5: 100 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	1000	9082	6391	23033	78	38910	0.00%	1.1/sec	2.36
Enter Result ID	1000	10500	7798	23673	32	40691	0.00%	1.1/sec	2.95
View Result	1000	48022	37690	89101	1032	213408	4.60%	1.1/sec	4.61
Return to Menu	1000	9809	8344	20423	47	36816	0.00%	1.2/sec	2.52
TOTAL	4000	19353	9719	44519	32	213408	1.15%	4.4/sec	12.20

Table 5.10: Simple Environment, 100 Requests

Note: At this level of throughput, errors result at the rate of 1.15% of total requests or 4.6% of all view result requests. The source of error was identified as a lack of resources on the database connection pool. Further throughput may be achieved through tuning the connection pool.

* Results achieved via use of load balanced configuration

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	540	645	906	1046	15	2875	45.56%	4.6/sec	8.14
Enter Result ID	514	712	907	1078	15	2281	49.42%	4.4/sec	8.84
View Result	490	1633	1000	2172	265	13859	48.37%	4.2/sec	11.70
Return to Menu	490	645	906	1015	15	2344	54.29%	4.4/sec	7.36
TOTAL	2034	900	922	1266	15	13859	49.31%	16.5/sec	33.72

Table 5.11: Simple Environment with Cluster, 100 Requests

5.5.2 Proof of Concept / Functional Testing on a Clustered Environment

5.5.2.1 Test Series 1: System Requirements Testing

In terms of system performance requirements, the following performance was measured:

5.5.2.1.1 Test 1: Maximum System Usage Expectation Testing:

5 individual users every 10 second, this cycle was repeated 100 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	500	55	31	78	0	1344	0.00%	4.7/sec	9.98
Enter Result ID	500	53	31	79	0	1266	0.00%	4.7/sec	12.43
View Result	500	818	625	1500	266	4797	0.00%	4.7/sec	19.28
Return to Menu	500	51	31	78	0	1172	0.00%	4.7/sec	9.99
TOTAL	2000	244	47	688	0	4797	0.00%	18.6/sec	51.52

Table 5.12: Clustered Environment, Maximum Expected Usage

5.5.2.1.2 Test 2 Double Maximum System Usage Expectation Testing:

10 User Requests on a 5 second period, repeated 100 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	1000	96	47	172	0	1813	0.00%	5.9/sec	12.66
Enter Result ID	1000	101	47	188	15	1656	0.00%	5.9/sec	15.83
View Result	1000	1232	1110	2032	234	3406	0.00%	5.9/sec	24.60
Return to Menu	1000	93	62	172	0	1532	0.00%	6.0/sec	12.74
TOTAL	4000	381	78	1391	0	3406	0.00%	23.7/sec	65.51

Table 5.13: Clustered Environment, Double Maximum Expected Usage

5.5.2.1.3 Test Series 2: System Stress Testing

In terms of system stress testing, the system was tested as follows:

5.5.2.1.4 Test 1: 10 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	10	37	31	109	15	109	0.00%	2.2/sec	4.73
Enter Result ID	10	23	31	31	15	31	0.00%	2.2/sec	5.88
View Result	10	501	437	1078	282	1078	0.00%	2.1/sec	8.54
Return to Menu	10	36	32	79	15	79	0.00%	2.2/sec	4.70
TOTAL	40	149	32	453	15	1078	0.00%	8.1/sec	22.46

Table 5.14: Clustered Environment, 10 Requests

5.5.2.1.5 Test 3: 25 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec

Menu Screen	250	260	156	609	15	2500	0.00%	5.5/sec	11.68
Enter Result ID	250	230	125	484	15	1766	0.00%	5.5/sec	14.54
View Result	250	2653	2329	5922	282	9937	0.00%	5.4/sec	22.41
Return to Menu	250	256	141	578	15	2031	0.00%	5.5/sec	11.67
TOTAL	1000	850	218	2797	15	9937	0.00%	21.6/sec	59.87

Table 5.15: Clustered Environment, 25 Requests

5.5.2.1.6 Test 3: 50 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	500	815	532	2093	15	5219	0.00%	5.8/sec	12.43
Enter Result ID	500	782	406	2203	15	5984	0.00%	5.9/sec	15.71
View Result	500	3829	3688	6563	438	8797	0.00%	5.9/sec	24.29
Return to Menu	500	873	515	2281	15	6062	0.00%	6.0/sec	12.72
TOTAL	2000	1575	797	4438	15	8797	0.00%	23.1/sec	64.00

Table 5.16: Clustered Environment, 50 Requests

5.5.2.1.7 Test 4: 75 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	750	80	31	125	0	1641	0.00%	3.6/sec	7.71
Enter Result ID	750	121	32	250	15	2000	0.00%	3.6/sec	9.60
View Result	750	1653	1312	3391	266	5250	0.00%	3.6/sec	14.88
Return to Menu	750	135	46	297	0	2156	0.00%	3.6/sec	7.71
TOTAL	3000	497	47	1718	0	5250	0.00%	13.4/sec	37.03

Table 5.17: Clustered Environment, 75 Requests

5.5.2.1.8 Test 5: 100 User Requests on a 5 second period, repeated 10 times.

Task	#Samples	Avg	Median	90%Line	Min	Max	Err	Throughput	KB/Sec
Menu Screen	523	101	31	125	15	1156	5.54%	5.6/sec	11.75
Enter Result ID	502	206	31	953	15	2484	10.36%	5.4/sec	13.65
View Result	472	1017	625	1437	250	6968	27.33%	5.1/sec	17.05
Return to Menu	471	186	31	922	0	1203	14.65%	5.2/sec	10.53
TOTAL	1968	368	32	1000	0	6968	14.18%	19.7/sec	49.14

Table 5.18: Clustered Environment, 100 Requests

5.6 Performance Test Results Analysis

5.6.1 Overview

The performance test results were analysed to evaluate their request throughput, average response times, rate of error generation and cause of error over the series of results generated. The results were analysed based first on their environment configuration and then incomparision with each other, to draw a valid conclusion on the capacity of the application for scalability to meet future application demands.

5.6.2 Simple Environment Analysis

In regards to the maximum expected usage, the simple application environment met the specified requirements. As seen from the figure 5.6, the throughput decreases from 9 requests per second to 7.3 reuests per second, as the load increases. In addion, figure 5.6, the decrease in response time is the result of an increase in time taken for the task "Retrieve & View Result" to complete. This is not unexpected, as this call results in all web-service calls and business logic execution, retireving the reslt from the legacy system and performing the "added value" tasks via other web-service calls.

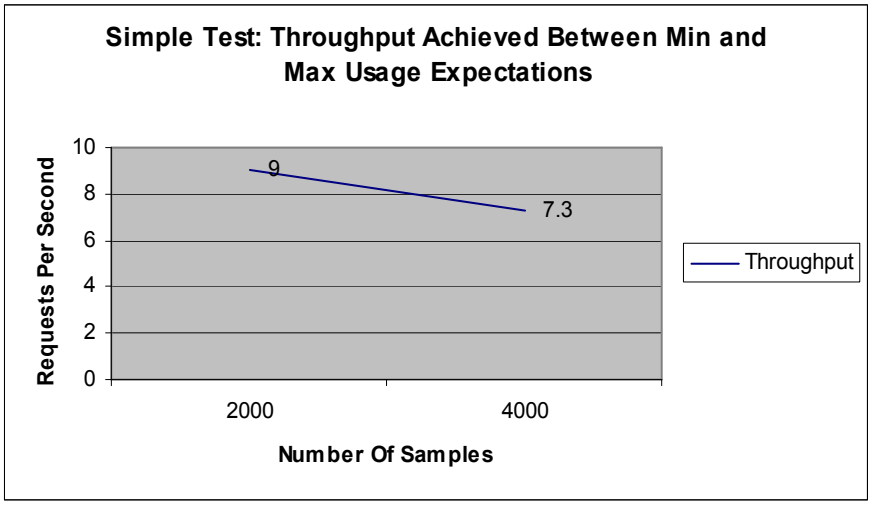


Figure 5.6: Simple Environment - Application Thoughput Rate

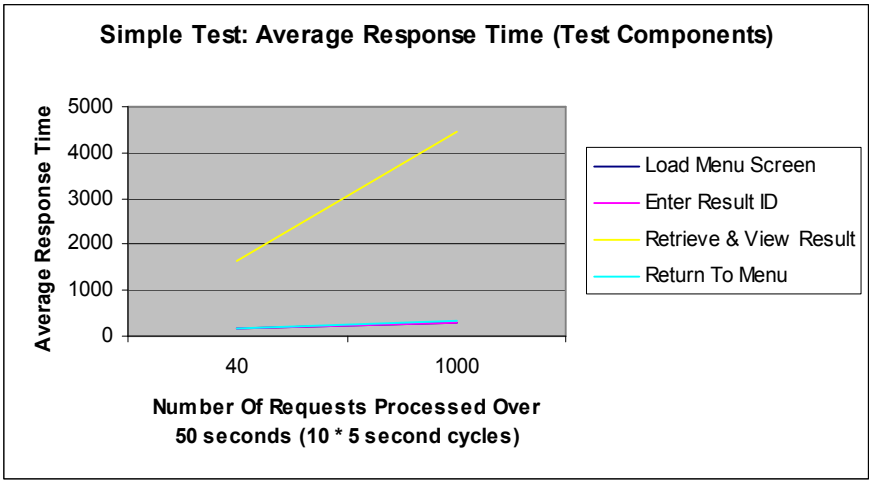


Figure 5.7: Simple Environment - Average Response Time

An analysis of the stress testing results indicate the point of load failure, upon examination of error rate (figure 5.9), errors begin to appear once load approaches 75 requests per 5 second test cycle. In coordinace with this, the throughput of requests (figure 5.8) drops from over 7 requests per second to under 5 requests per second. In addition, the response time of the request "Retrieve And Display Result" continues to increase. Through examination of the application server logs and application outputs, the source of this degradation in performance was revealed to be caused by the exhaustion of resources in the database connection pool.

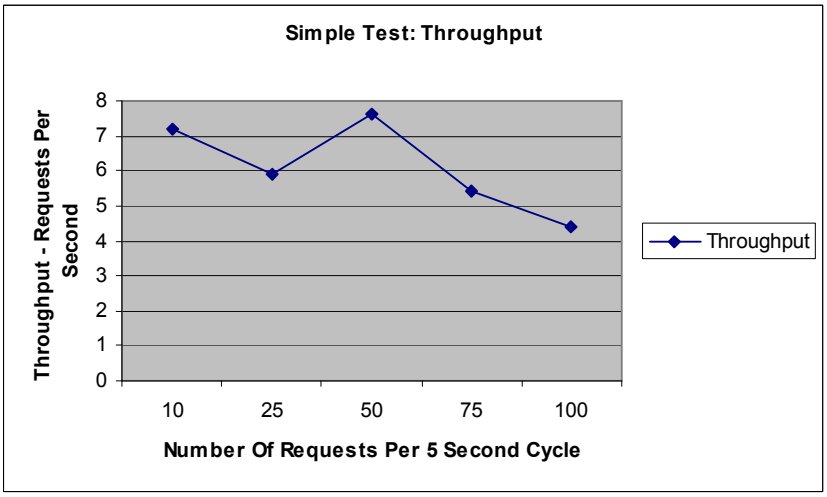


Figure 5.8: Simple Environment Stress Test - Throughput

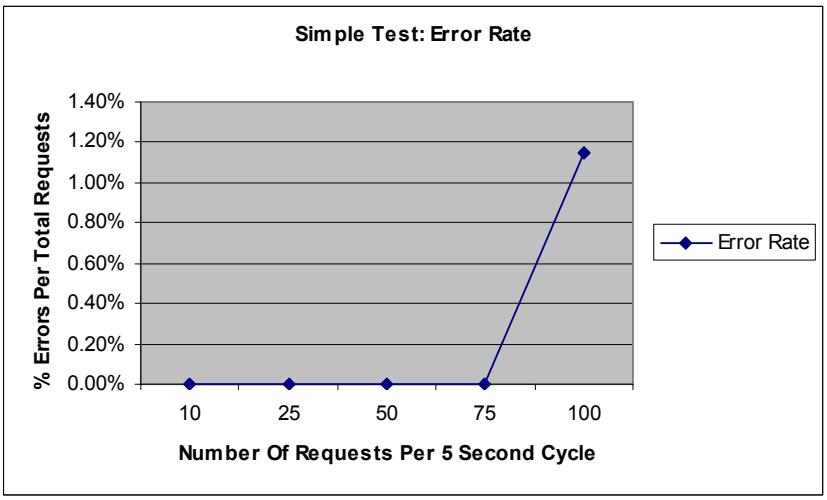


Figure 5.9: Simple Environment Stress Test – Error Rate

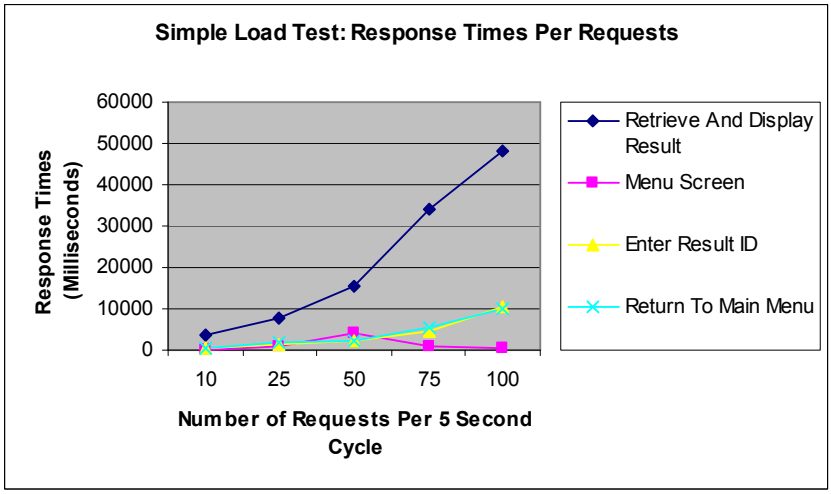


Figure 5.10: Simple Environment Stress Test – Respose Times

For the purpose of comparison with the clustered environment results, results from the additional tests performed on the simple environment, including routing through a software load balancer, revealed a higher rate of throughput (figure 5.11). However on further analysis, this result was misleading, as the higher rate is inline with a higher rate of error generated (figure 5.12). The error rate approaches 50% of all requests. Similarly, the response time for "Retrieve And Display Result" (figure 5.13) rapidly decrease. The increase in throughput exceeding the level of 16 requests per second was then determined as the direct result of the error being invoked. The source of the error was the communication between Jmeter invoking the tests and the software load balancer. The load balancer had reached its capacity in terms of open connections and as a result, the failure was returned to Jmeter in a much shorted period of time as no execution on the application was performed.

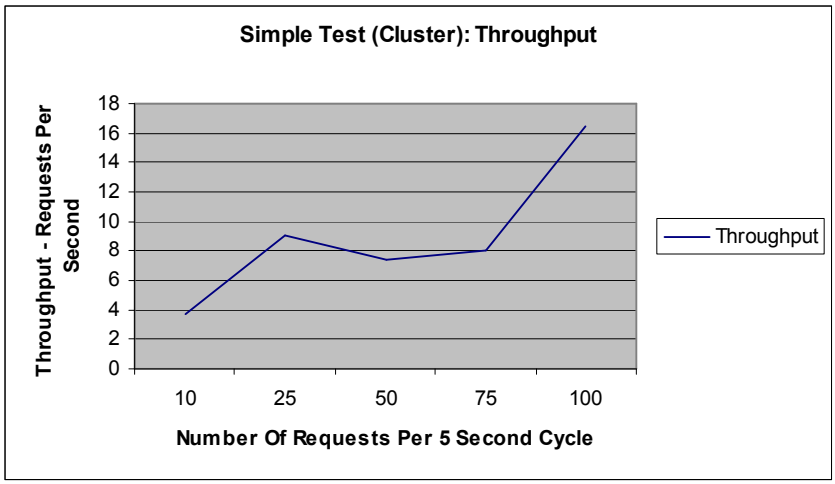


Figure 5.11: Simple Environment (Clustered) - Throughput

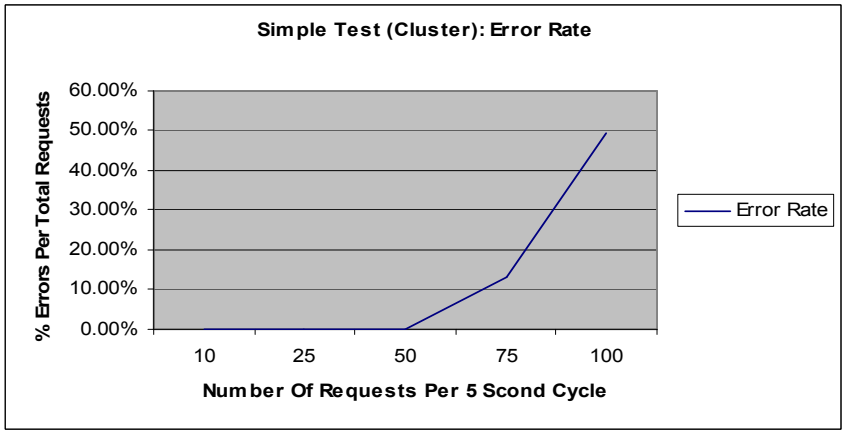


Figure 5.12: Simple Environment (Clustered) – Error Rate

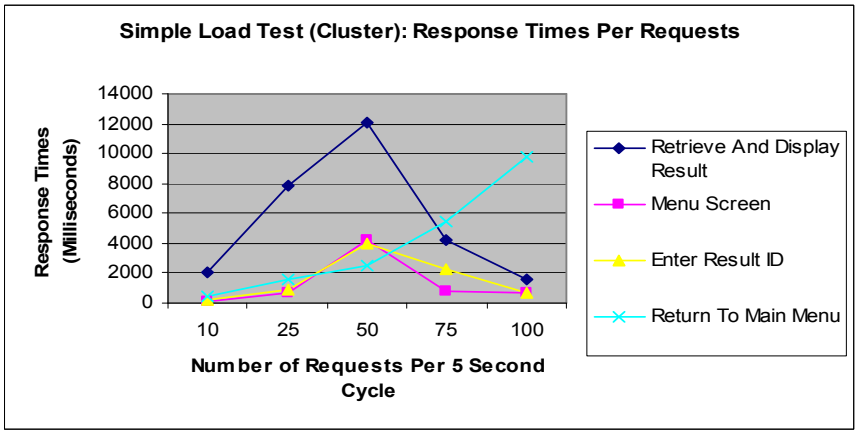


Figure 5.13: Simple Environment (Clustered) – Response Times Per Request

5.6.3 Cluster environment analysis

In regards to the maximum expected usage, the cluster application environment met the specified

requirements. As seen from the figure 5.14, the throughput approaches 23.7 requests per second. In addion a decrease in response time is observed, with “Retireve And View Results” approaching 1200 milliseconds. (figure 5.15)

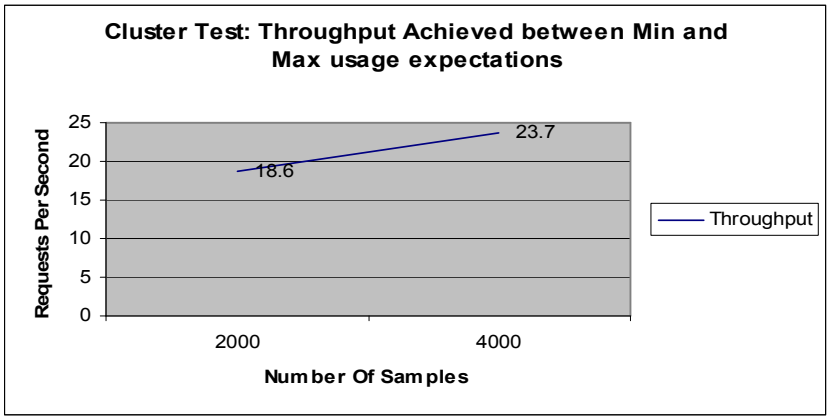


Figure 5.14: Cluster Test – Application Throughput Rate

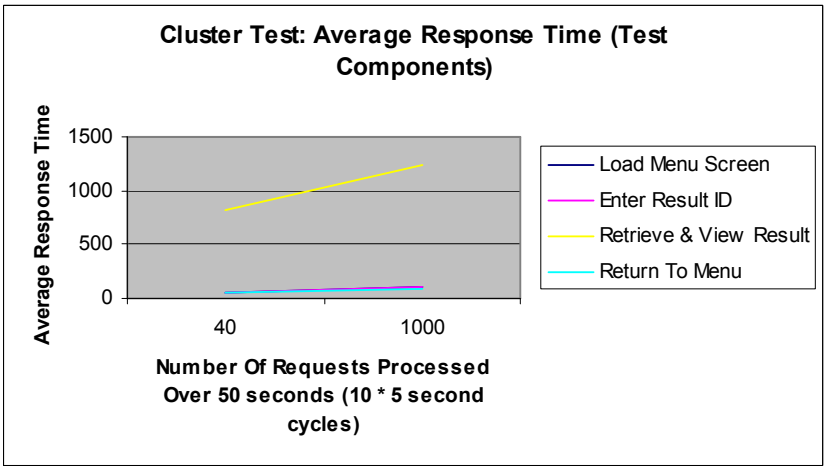


Figure 5.15: Cluster Test – Average Response Time

An analysis of the stress testing results on the clusted enviroment indicate the point of load failure, upon examination of error rate (figure 5.17), errors begin to appear once load approaches 75 requests per 5 second test cycle. Analysis of both throughput (figure 5.16) and response times (5.18) show a rapid decrease in the response times for the “Retireve And Display Result” test component.

Examination of the Jmeter log indicate the source error was the result of an exhaustion of available connections between Jmeter and the software load balancer. The failure returned from the load balancer occurs quickly as the application logic call is not invoked.

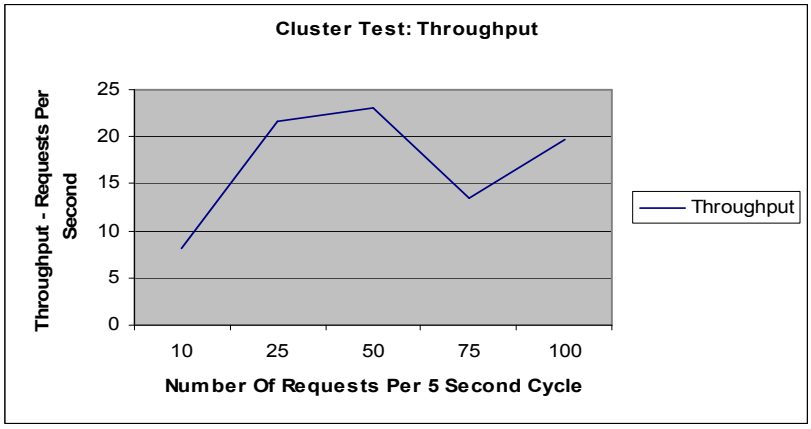


Figure 5.16: Clustered Environment – Throughput

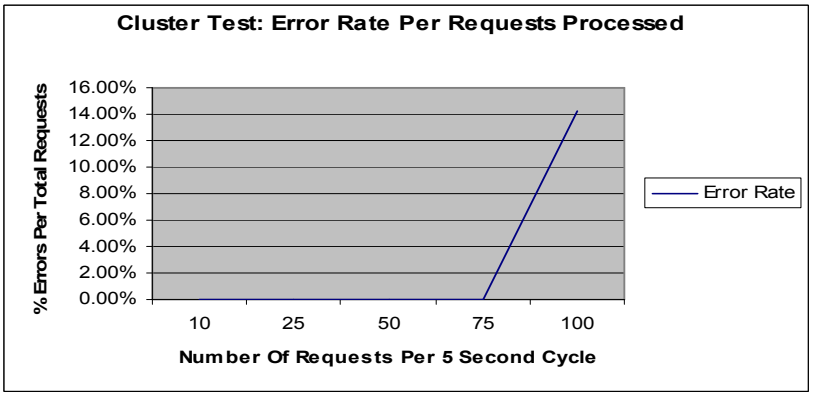


Figure 5.17: Clustered Environment – Error Rate

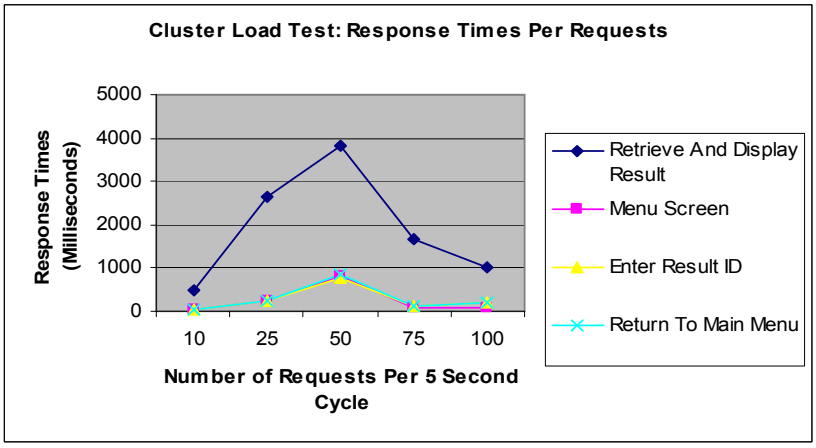


Figure 5.18: Clustered Environment – Average Response Times

5.6.4 Expansion Capacity

In performing a comparative analysis between the simple configuration and the clustered environment, the gain in throughput is immediately evident, reaching 22 requests per second

(Figure 5.19). This gain is noted up to the point of failure, noted at around 50 requests per 5 second test cycle.

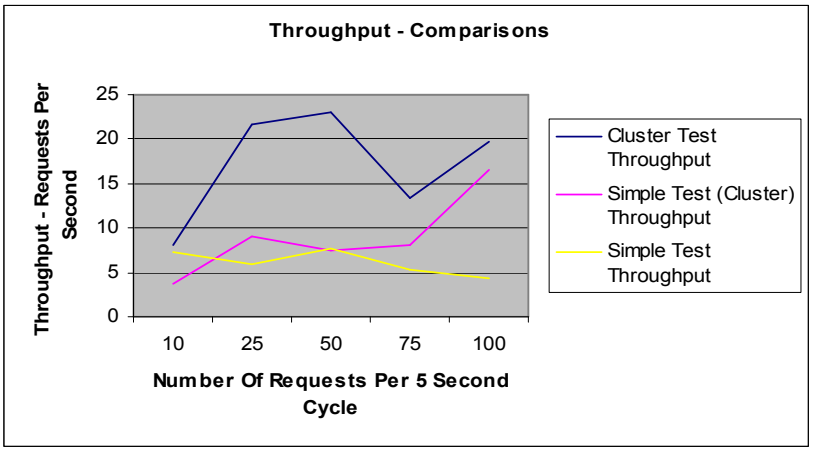


Figure 5.19: Comparison of Throughput Results

As the point of failure was determined as the software loadbalancer, it is reasonable to state that a hardware based loadbalancer would allow the scalability of the architecutre to grow. Similarly, the simple environment reached point of failure at 75 requests per 5 second cycle (figure 5.20). This point of failure remained low and was caused by an exhaustion in database connections within the application server. The low level was attributed to the freeing of database resources and their reuse.

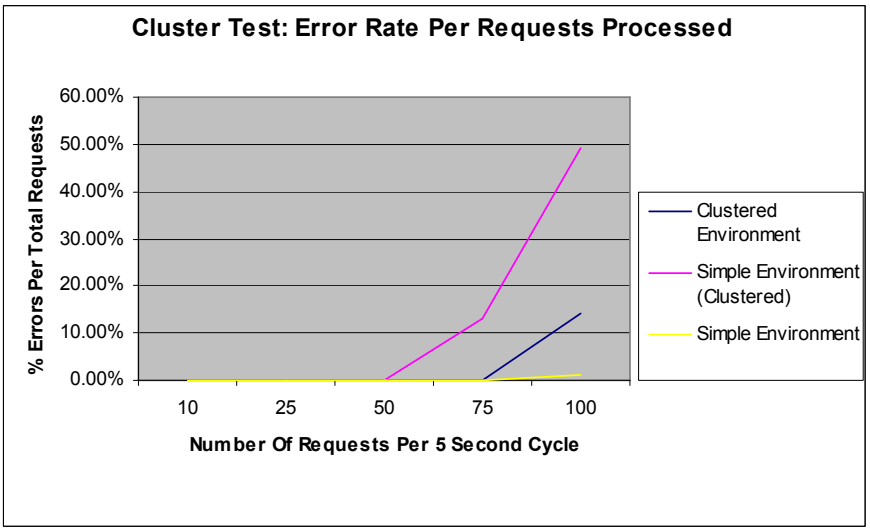


Figure 5.20: Comparison of Point of Failure and Rate of Failure

Through application of these factors, should the loadbalancer be replaced with a higher performance loadbalancer, the environment would support 225 requests per 5 second cycle, until each applications database connections become exhausted.

To improve scalability, an exercise in application and application environment tuning would be performed. The code of the concept application was not analysed and refactored to improve performace. There are several elements in the application environment which could also be tuned to improve performance. The source of failure, the application servers database connections could be updated to provide greater capacity. Weblogic, the application server, assigns a default maxium of 15 JDBC connections to the database connection pool. This value could be increased to grow the pool size, improving the scalability of the application. This change alone would reset the point of failure for the simple test environment. Further environmental tuning, such as the Java Virtual Machines (JVM) heap size could also be changed to provide a larger execution area for the application, improving scalability. It is also noted that while the use of different JDBC drivers will affect the performance of the application, JDBC performance issues were not encounter during the execution of this set of tests.

5.7 Summary

This chapter has provided an details on performance testing the SOA application, in terms of the quantification of performace test parameters, the design of the tests applied, the environment in which the tests were executed, the results of the tests and finally, an analysis of the test results.

The next chapter presents the goals, design, composition, result and analysis of a series of workshops based on the suggested migration strategy, targeted at three distinct groups of medical orientated stakeholders.

6 SOA Migration Workshops

This chapter details the goals, design, composition, result and analysis of a series of workshops, targeted at distinct groups of medical orientated stakeholders. The chapter outlines the design of each workshop, its target audience, goals, composition and an analysis of each groups' results. Finally, a comparative analysis is provides across all workshop groups.

The principal goal of workshop sessions is to obtain an overall view of the migration approach and SOA adoption in the medical domain across the three separate groups, Managerial Professionals (7), IT Professionals (7) and Medical Professionals (10) (End Users). Workshops consisted of a presentation outlining the migration approach and SOA, a demonstration of the proof of concept application, a questionnaire and a questions and answers session. All workshops were held in an informal manner, leaving scope for open discussion throughout. Although initially the prospect of recording the audio of workshops was proposed, this was deemed unsuitable as participants would feel scrutinized during the questionnaire and questions and answers session.

6.1 Managerial Professionals Workshop

6.1.1 Managerial Professional Workshop Group Details

This workshop covers the sector of Managerial personnel. Workshop candidates include medical system IT managers from various medical organisations (St. James hospital, The Matter hospital, and Cavan Public hospital) and consulting management staff (Accenture). Stakeholders include decision makers, financial controllers and project schedule creators.

6.1.2 Managerial Professional Workshop Goals

The goal of the managerial workshop is to gather the opinions of managerial personnel involved in the planning, development and rollout of medical projects. The data collected from this workshop covers the managerial aspects of the project development cycle, additionally, examining areas such as project group stakeholder makeup and foreseen barriers to adoption of the new system.

6.1.3 Managerial Professional Workshop Composition

Following a presentation on the managerial aspects of the concept system (Appendix D), and demonstration of the system, the workshop questionnaire was performed consists of the following subject areas:

- Cost / Benefit Comparison: Questions were structured to collect the participants view of planning an upgrade or migration and how a cost / benefit analysis would be approached.
- Factors determining project progression: Questions were structured to gather the participants

view of any significant factors which would dictate the projects progression, determination of the value gained from the extra cost involved with an initial SOA rollout and long term organisational goals. In addition, the use of SOA and the participants trust in SOA as an established technology were discussed.

- Project Planning: This section of the workshop collects data to determine how the participant would plan time requirements for both a legacy and an SOA project.
- Factors to be considered when managing implementation of both SOA and Legacy projects: This section of the workshop collects data on factors such as policy towards outsourcing, use of in-house legacy knowledge and the adoption of new technologies and techniques such as Agile Development / Extreme Programming.
- Use of socio-technical methodology: The workshop then gathers data on the use of social-technological methodology, such as its advantages / disadvantages when used in the context of the concept application and the number of non-technical stakeholders envisioned to be involved in a project of this size.
- Barriers to adoption of the system: Finally, this workshop collects information on foreseen barriers to adoption of the system in terms of a managerial stakeholders view.

6.1.4 Managerial Professional Workshop Results

6.1.4.1 Cost / Benefit Comparison

Upon discussion on the task in calculating the cost of a legacy upgrade, 100% of participants responded that while calculation is straight forward, based on hours to plan and release new functionality, 28% of participants also responded that additional factors such as historic knowledge of the system and the required skills must be taken into account, as the system may be a black box system.

100% of participants responded that planning an SOA based project would be performed as planning a new project which may result in increased costs. In addition, 48% of participants responded that while costing SOA, unknown factors within SOA present a high risk. 28% of participants responded that performing a cost benefit analysis would prove a lengthy process and that the requirements document should be split into patient and organisational level benefits and that while the soft benefits may be difficult to determine and hard to justify. At this stage, 14% of participants noted the benefit that each SOA service delivered should be considered as long term benefits as it constitutes as a possible deliverable in other projects.

Only 28% of participants stated that factors such as extendibility and the modular aspect of SOA

would be considered when determining benefits. 56% of participants stated that the determination of costs/benefits would require national level input, given the long term goals of shared services.

When questioned at which point the benefits of SOA outweigh additional costs, 42% of participants responded that benefits outweigh costs when the number of systems integrated is greater than any integrated system elsewhere in the organisation. Other participants commonly responded that scalability and reusability developed for future work would be accounted as long term benefits.

6.1.4.2 Factors Determining Project Progression

In discussing the factors determining progression, all participants agreed that a clear demonstration of the savings made through development of reusable services in the form of economies of scale would greatly assist in justification of costs and allow for progression. 14% of participants added that when considered on both a regional and national scale, cost justification would be affected based on the developments fit with long term goals, stating that "There is no point in wrapping systems in-house without considering the interaction with new primary care entities", such as the HSE shared services and the possibility of a national heath care record.

100% of participants agreed that the organisations long term plans would prove a factor in project progression. 42% of participants stated that should interoperability be proven and the long term strategy requiring system integration, consideration could be made on the forwarding of resources for projects planned at a later stage, in areas where the work performed would remove tasks for future project. Specifically, this refers to the wrapping of legacy systems for the purpose of interoperability.

With regards to the overrun of budgets acting as a deterrent on project progression, 100% of participants stated that the longer term goals of the organisational would take precedent over a shorter term project budget. One participant noted that there must be a balance to this scenario, at least one unit of work demonstrating the inclusion of long term functionality (interoperability) should be demonstrated long before any issues arise from budget overrun and that the overall deliverables resulting from budget overrun should fall in line with realistic organisational timelines.

6.1.4.3 Project Planning

100% of participants agreed that when planning a legacy upgrade, the plan would consist of incorporating the current requirements, without considering the long term migration of the system. As a result, planning is a straight forward process as there are "more knowns than unknowns".

In planning an SOA system, 72% of participants agreed that the system should be planned as a new software rollout and that future budgets and plans could be taken into account. One participant

stated that the use of future budgets would require the comprehension of integration at a national level given the rollout of shared services by the HSE.

In discussing the effect of the introduction of new screens on work practice, adding complexity and possibly introducing work flow changes, 42% of participants stated that this consideration should be accounted for by change management in the planning stages of the project and falls outside the scope of project progression. Remaining participants agreed that such change may require training and time for end users to adopt.

6.1.4.4 Considerations in Managing Implementation

In managing a legacy implementation, all participants agreed that project planning and implementation would include unit testing, despite that fact that the legacy implementation would consist of "patching" an existing system. Participants also stated that in dealing with the lack of documentation on the current legacy system, outsourcing to a specialist (56%) or performing a gap analysis, functional investigation and production of high level explanatory docs for each areas such that they can be understood by everyone(28%), were the preferred options.

In testing the legacy patch developed, 100% of participants agreed that unit and integration testing would be performed on both the patched code and the functional code of the legacy system affected would be tested. 28% of participants also stated that further testing in the form of workflow testing and regression testing of the system would be performed. The majority of participants 72% stated that they would outsource to specialists if system was not developed and not known internally.

Upon discussion of the management of an SOA implementation, participants stated that they would deal with the lack of information of the legacy components to be wrapped in the same manner as in developing a patch on an unknown legacy system, outsource to specialists (56%) or gain the required knowledge in-house (28%). 58% of participants agreed, that should the development of SOA components be performed in-house, current development methodologies could be dropped and the introduction of new methodologies (Extreme programming / agile methods) could be adopted. One participant noted that this factor would depend on current in-house expertise and that the adoption of new practices should only occur upon proving that the current in-house methodologies were deficient and that proposed methodologies added value to the process.

With regards to adopting an outsourcing strategy through the use of SOA, the general consensus was that the adoption of SOA is not a direct justification to outsource, although it might simplify outsourcing, costs would be considered as a stronger factor.

6.1.4.5 Use of Socio-Technical Methodology

All participants responded that a Socio-Technical methodology would be adopted in upgrading a legacy system. Participants from one organisation (42%) stated that such methodologies, though titled differently with their organisation, this approach would be insisted upon from project conception. In addition, all participants responded that this approach be used when developing a new system, despite that fact that the new system closely resemble the legacy system in terms of functionality and user interface.

In considering that the project team would consist of 4 full time developers, 2 test engineers, 1 architect/team lead and 1 project manager, participants responded with a variety of estimates(4, 10, 6) in the number of additional members on the project team from a socio-technological perspective. All participants also stated that this number was open to debate and strongly influenced by the nature of the development. In addition, all participants agreed that the socio-technical group would consist of personnel from different departments, who would be involved in workflow planning and end user interaction with the system.

6.1.4.6 Barriers to SOA Adoption

In examining the barriers to adoption of an SOA based system participants note that complexity, ownership and determination of boundaries to a new architectural layer(42%) would prove barriers to adoption. 42% of participants stated that contractual arrangements and resistance to change would prove to be barriers. 14% of participants stated that complexities from new communication interfaces and a lack of skills required to implement, combined with the risk of adopting a new technology set would also act as barriers to adoption.

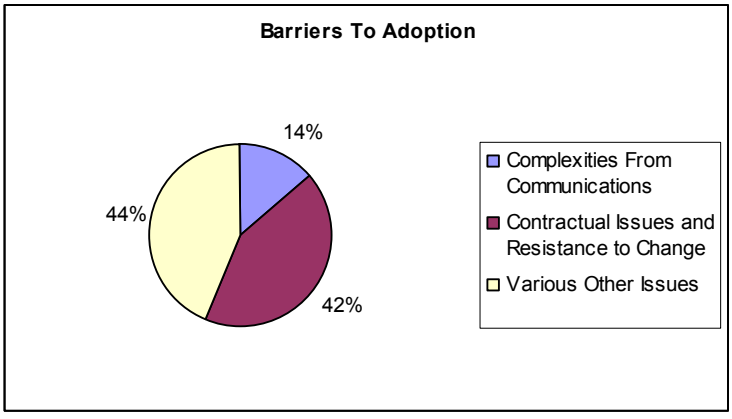


Figure 6.1- Barriers to Adoption

6.1.4.7 Post Launch - System Operation & Maintenance

A majority (58%) of participants agreed that there are little adoption costs of a legacy upgrade as

the upgrade is primarily based on existing, functional technology but there may be higher running costs. 42% of participants disagreed with this statement, stating that in their experience, the costs associated with running the current legacy system were not high and that the running costs are dictated by the "system level of complexity".

Regarding a SOA system, while all participants agreed that the platform is an improvement on the legacy system in terms of reliability and expansion capacity, 72% of participants viewed SOA as requiring more personnel for maintenance in the longer term, in addition to a high initial implementation cost. 42% of participants stated that this cost would be directly related to national and local planning. For example, if the majority of SOA services were housed in the physical location and maintained by the same team, savings would be realised.

6.1.4.8 Participants View of Migration Approach

Participants commented favourably on this migration approach, comparing it to the phased rollout of other technologies, stating this "technically makes sense" and is justifiable. It was also stated that given this approach obtaining user buy-in should not prove difficult. The gradual transition of data from the legacy system to a modern system at a time convenient to the organisation was welcomed with the understanding that "the less data conversion to be done the better". The concept of SOA, while new to some participants, was simply taken as a mechanism for interoperability, with the movement of business rules into a modern technology set. The capacity to move the business rules out of the legacy system over time, reducing the final migration effort to a data migration was regarded as a core strength of this approach.

One participant did question the validity of the approach in terms of its true function, sighting that interface engines between technology sets and interface wrappers are current off-the-shelf products, which do not require heavy implementation and that any migration from the legacy system would be contracted to the legacy vendors who may provide SOA interfaces as an extra feature and not as the core interfacing mechanism.

Participants also stated that while they may be in favour of waiting for the appropriate time to perform the migration, for example, when the organisation is migrating other systems, this approach would prove a good pilot for the gradual migration to SOA. As a pilot approach, it would assist in the reduction of risk and understanding of the unique constraints involved.

Overall, users felt that in terms of cost / benefit, this approach would be a viable option; one participant stated that a pilot would be required to prove viability and another stated that SOA is currently an unproven technology in the medical domain. Other participants (42%) noted that the introduction of such technology through this approach would greatly assist in the reduction of dual

and triple data entry, so the technology itself, regardless of the approach to its adoption would make its introduction viable. Finally, one participant noted the while technically sound, the approaches viability would be down to budget, if the approach fitted with the longer term migration aims of the organisation and could be spread over multiple financial years.

6.1.5 Managerial Professional Workshop Analysis

Upon examination of the Management Workshop results, the following key points were revealed:

- Planning a legacy upgrade would prove a simpler task than an SOA project, as the SOA project constitutes a brand new project.
- While some concern was raised on having the appropriate legacy skills available, this would not appear to be a deterrent in comparison to the unknown factors of SOA.
- The overall strengths of SOA were considered benefits which could justify costs and allow for progression of the project, though a demonstration of the SOA application would greatly assist with this justification.
- Planning a legacy upgrade would focus on the current requirements, while an SOA plan would incorporate longer term organisational goals.
- Outsourcing complex or unknown legacy functionality for implementation was a viable option in performing a legacy upgrade or SOA development, where legacy components were not fully understood in-house.
- New development methodologies such as extreme programming and methodologies, such as the spring framework, would be incorporated with a new SOA project.
- The introduction of SOA does directly relate to the organisations outsourcing strategy.
- A socio-technological methodology would be adopted with both legacy and SOA developments.
- Application complexity, component ownership, contractual issues and general resistance to change are the primary barriers of adoption to SOA.
- In general, management have different views on the cost and complexity of maintaining a legacy system.
- Management currently view SOA as a high maintenance technology, due to its component

orientated nature.

- Management stated that the approach to SOA adoption suggested by this project "technically makes sense". Gradual migration from legacy systems is viewed as a good approach as the final data migration from the legacy system should consist of a small data set.
- Management viewed the approach as a good pilot candidate for SOA technology in a medical setting, as the technology is currently unproven.
- Regardless of technological advantages, the organisations long term goals and budgets influence the viability of this migration approach.

6.2 IT Professionals Workshop

6.2.1 IT Professional Workshop Group Details

This workshop group consist of IT personnel involved in all stages of the software's lifecycle. Workshop candidates include staff from various medical organisations and consulting organisations. Participants are from development, architecture, testing and operations backgrounds and medical system IT personnel.

6.2.2 IT Professional Workshop Goals

The goal of this workshop was to obtain the insight of IT professionals (Designers, Developers. Operations / Maintenance personnel) on the migration of a legacy application to an SOA through the addition of new value added service components.

6.2.3 IT Professional Workshop Composition

Both development and operational staff were exposed to the technical makeup of the system to gauge their reaction to wrapping the legacy system Vs rewriting. This provided an insight to the total cost of ownership of the system in terms of initial development and keeping the system functional as per an agreed Service Licence Agreement (SLA).

The workshop consists of a presentation (Appendix F), a demonstration of the proof of concept application and then an interview, following the development lifecycle and examines legacy wrapping vs. and SOA rewrite at each applicable stage. This covers all aspects of design, implementation, deployment and maintenance of a solution. The feedback for each section is a combination of the users past experience and anticipation based on experience and knowledge.

- Requirements Gathering: The workshop gathered data in terms of unique constraints, influences and limitations by the underlying technology at this stage of the development lifecycle.

- Requirements Analysis: Next, the workshop gathered details such as limitations applied to the analysis of the system given prior knowledge of the target technology. (Examples: User Interface requirements; Communications Requirements; Load requirements.)
- System & Software Design: Data on the different approaches to designing software for both technologies was collected.
- Implementation & Unit Testing:
    - In terms of a legacy upgrade, data on the view of implementation being a quick task due to the fact that an existing system is being patched and limitations applied at this stage of the lifecycle as a result of the quality of legacy code or target environments was collected.
    - In terms of an SOA development, data on the view that the wide choice of platforms and technologies is seen as an advantage, of implementation being seen as a more arduous task as the project is a brand new build, and are there advantages perceived from the use of the latest technologies is collected.
- Integration & System Testing

    - In terms of a legacy upgrade, data on the view of integration as a simple process as the system environment exists, pitfalls experienced in integration of new legacy components with the existing legacy system and the level of system testing is gathered.
    - In terms of an SOA development, data on integration of web-services and view of complexity associated with composite application integration and environment configuration is gathered. In addition, details on the level and type of testing performed at this stage were gathered.

- Deployment: In deployment of an upgraded legacy system and SOA based system, data was collected on the deployment process used, the amount of downtime anticipated and the risks associated.
- Post Implementation (Operations / Maintenance): Data was collected on operational /

maintenance issues and resource requirements anticipated with both legacy and SOA based systems.

6.2.4 IT Professional Workshop Results

6.2.4.1 Requirements Gathering

On considering requirements gathering for a legacy system, all participants stated that unique constraints exist. Such constraints included system resources, access to the required system for analysis, impact of changing old system interfaces, effects on changing the existing data, limitations on what the legacy system can technically accomplish and limitations on processing and storage capacity. All participants stated that the underling technology has an effect at this stage of the lifecycle. Examples of this included the fact that the system is old and might not be able to process the required volume of data and the ability of the system to communicate with external data sources. Participants did not foresee any limitation based on legacy resources, as the organization already deploys a legacy application, they are sure to have access to such expertise either in-house or through contracts with vendors.

Participants did not foresee any unique constraints with SOA at this stage of the software life cycle. On the contrary, features such as the flexibility of the technology, its ability to interact with other SOA services and produce any user interface (web, native windows interface, interface to emulate legacy system look and feel etc) were noted as advantages of SOA. 42% of participants also stated that this level of flexibility should be viewed as an advantage at this stage of the lifecycle as it could be used to demonstrate the possibilities of the technologies when collecting requirements from the end user group. All participants agreed that there was a sense of "we can provide more" with SOA.

6.2.4.2 Requirements Analysis

In discussing technological limitations of a Legacy development at the requirements analysis stage, all participants agreed that there may be limitations in what can be provided by the legacy user interface. All participants also agreed that there may be limitations in connecting the legacy system to other systems. One participant stated that if the required data could be replicated locally, this solution would potentially only require LAN communications. However, if local replication was not possible there would be the need to communicate with external sources which would involve a heavy development effort. Finally, all participants agreed that there are restrictions defined through the anticipated throughput of the system, as the legacy system may not be capable of processing the calculated load.

Similarly, in discussing limitations of an SOA development at this stage of the lifecycle,

participants did not note any restriction regarding the user interface requirements. One participant noted that the communications infrastructure for web services should be a secured resource between all services. This may apply limitations based on the organizations network structure. In addition, while all participants stated that load on the system would not be an issue; participants agreed that the anticipated load should be calculated, in order to define the hardware and software requirements for the system.

6.2.4.3 System & Software Design

All participants agreed that the approach to software and system design is fundamentally different between a legacy patch and SOA solution. One is a "single application" and the other is a "services based solution".

In designing the legacy based solution, participants agreed that the effort involved was highly influenced by the quality of existing documentation and the understanding of the current system.

28% of participants stated that the development could be either easier or more complex based upon the requirements, regardless of the legacy system. 42% of participant reported that the design of the legacy components would be influenced by the level of expertise at hand and thus the effort in design is directly affected by the resources available. An example was given of a developer with little experience in COBOL extending an interface and batch processing application to incorporate a functional change based on the addition of a field in a position delimited file. This task took an experienced COBOL developer less than half the time to complete.

With the SOA solution, 56% of participants agreed that the solution is easier to design as it is a brand new build. One participant stated that the key factor was the development/use of a suitable API to leverage existing systems. Participants disagreeing with this statement sighted the fact that new skills would be required by personnel and support for these personnel may be required from external sources. All participants stated that new development practices and methodologies, such as those topical with SOA (Extreme programming, Spring Framework etc) should be brought in with the adoption of SOA at the design phase. One participant also noted that "In the case of SOA when you are 'wrapping' an existing system, you are never really working from the ground up."

6.2.4.4 Implementation & Unit Testing

On discussion of implementation and unit testing of a legacy development, participants provided a mixture of responses to the question "Is implementation seen as a quick task because an existing system is being patched?" While 28% of participants responded "Yes", the remainder stated that depends on the requirement to integrate with new data sources. All participants agreed that implementation is greatly affected or restricted by the quality of the current code and the coding

limitations are inherent with legacy systems. In terms of testing all participants stated that unit testing would be required, 28% of participants stated that some degree of integration testing should be performed at this stage and one participant stated that testing should be performed in line with the projects milestones.

With regards to SOA development, 86% of participants agreed that the wide choice of platforms and technologies would be advantageous. One participant pointed out that the there "is danger if you choose a tech/standard that is not widely adopted and supported in the future".

58% of participants stated that SOA implementation would not be seen as slower because of brand new build, as although initial setup may be slow rapid development practices and open source libraries (apache libraries etc) can be applied to speed up development. One participant noted that the speed of implementation depends on the quality of APIs of external services to be incorporated, and the API to the existing legacy system. While majority (72%) of participants stated that there are advantages perceived from the new technologies and methodologies applicable for SOA development, 28% of participants stated that there may be skepticism about new technologies used in implementation, as stability is very important in healthcare. All participants stated that unit testing would be applied at this stage of the project lifecycle, with some (28%) integration testing.

6.2.4.5 Integration & System Testing

As with implementation, on the topic of legacy integration seen as a simple task, 28% of participants responded "Yes", the remainder stated that depends on the requirement to integrate with new data sources or resource. 42% of participants noted that pitfalls experienced in integration of the new legacy components with the existing legacy system include network resources integration (asynchronous RMI, EJB, DCOM etc) while other participants stated that the same problems with introducing new features to any system would be experienced, such as data access, data flow, performance issues. All participants stated that integration testing would be performed on a legacy development at this stage of the lifecycle, 28% of participants also stated that UAT testing would also be performed.

In terms of SOA integrations, participants consider web-services integration a more complex task than a single application as the integration is performed with potentially different back end systems and remote services with 24% stating that additional environment configuration required may act as a deterrent from using SOA. With regards to testing all participants stated that, as with legacy development, integration testing would be performed at this stage of the lifecycle, 28% of participants also stated that UAT testing would also be performed.

6.2.4.6 Deployment

On discussing system deployment, the majority of participants (86%) agreed that a full deployment of the legacy system would be required, with 28% of participants adding that the system should be test run on replica system and then released into the live environment. One participant stated that the deployment mechanism depends on legacy infrastructure. All participants agreed that downtime would be required, estimated at anywhere from 3 to 5 hours but based on the complexity of the system. Risks defined with such a deployment were listed as failure of the new system/patch to deploy and the back end systems not responding. In such cases, a full roll back is required, which may take several hours. One participant stated that the risk was negligible as the release would be performed during a defined system downtime, which would include contingency for a full system rollback.

Regarding SOA developments, all participants stated that an initial full system deployment would be required; however subsequent deployments may only require the deployment of specific components. All participants agreed that downtime would be required for this deployment; the estimate for this would depend on the number of remote systems which require simultaneous deployments. As with the legacy deployment, all but one participant stated that risks include failure of the deployment of any one system and that rollback could take several hours.

6.2.4.7 Post Implementation (Operations / Maintenance)

Participants noted that a lack of knowledge of the code-base, little documentation, flat file / legacy storage, data indexing issues and a lack of legacy experience as unique operational / maintenance issues associated with legacy systems. As there is a reliance on legacy knowledge, "when something breaks it may be difficult to fix". All participants agreed that additional costs in terms of personnel and system resources in the form of specialist external consulting may be required.

Higher maintenance costs due to a more convoluted architecture, and a lack of support for open source software were noted as issues unique to operational and maintenance of an SOA application. Participants agreed that the failure of one service central to other services and the time taken to discover the faulty service / service connection is the biggest problem faced with a live SOA application. Participants noted additional costs in system monitoring and specialist personnel however 42% of participants noted that this can be reduced if all the systems are monitored by one operations team. Such ownership would also assist in the quick resolution in the single component causing system failure issue.

6.2.4.8 General View of Migration Approach

Participants stated that the migration approach suggested is performed at present by some

organizations, though typically performed as short term step and migration of the legacy system sooner rather than later. The migration approach was also commented as “more hassle initially, but wrapping the existing system is better for the users in terms of interaction with the system, and also doesn't replicate an existing system that is working perfectly adequately” and “favorable as the migration can happen over a long period of time without effect on end user”.

In discussing the strengths in other approaches such as cold turkey, participants noted that cold turkey has the advantage that the migration is completed in one big migration, despite the fact that this can be expensive and risky. One participant stated that the migration approach taken may depend on the personal preference of the architect but ultimately it is “your resources that dictate your approach”.

In terms of personnel, system and financial resources, 72% of participants stated that this approach would be a viable, with 28% stating that it should be a proven solution with cost and patient benefits proven in advanced. Another 28% stated that financial resources would be the deciding factor; the participants would “need to see some comparative figures, that's why we have the tender process”.

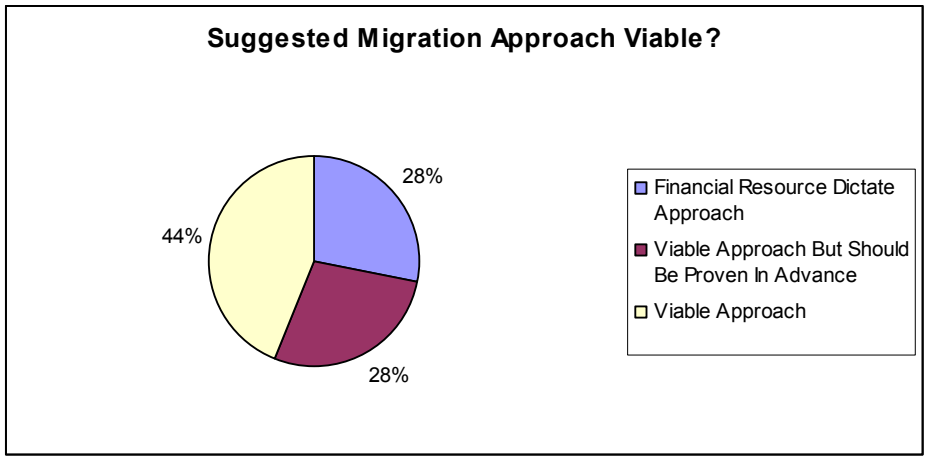


Figure 6.2: Opinion of IT Professional on Viability of Suggested Migration Approach.

6.2.5 IT Professional Workshop Analysis

Upon examination of the IT Professional Workshop results, the following key points were revealed:

- Upon requirements gathering, the limitations applied by legacy technology should be kept in mind. Limitations include the user interface, ability to interoperate with other systems and the legacy systems maximum throughput.
- Flexibility in terms of user and system interfaces are seen as an advantage at the

requirements gathering stage of a software lifecycle.

- As with requirements analysis, legacy systems are seen as restrictive when dealing with user interfaces and inter system communication during requirements analysis and software design stages.
- The effort in patching a legacy system with new functionality is directly affected by the quality of the current legacy code and documentation and the availability of experienced staff in-house.
- The process of designing software for legacy patching is fundamentally different from designing an SOA application.
- Less effort is required in the design of an SOA application as the IT professional is dealing with a new, clean build, though external factors such as network infrastructure and available hardware will influence this.
- Implementation of a legacy "patch" is seen as a quick task, as the current software and environment exist.
- Implementation of a new SOA application is seen as a slow task, given the initial environment setup and configuration.
- SOA technology has the advantage of new supporting development methodologies and tools.
- During implementation, IT professionals would perform unit testing on both legacy and SOA developments, in addition to a low level of integration testing.
- The process of implementation and integration of a legacy patch can be complicated by its requirements to interoperate with external systems.
- Integration of an SOA application is considered a complex task, which may deter some IT professionals from adopting SOA technology.
- The deployment of both SOA and legacy solutions would require system downtime, which ay last several hours, though this can minimised by the prior deployment on test platforms.
- A lack of understanding of a legacy system, poor documentation and the older data storage mechanisms (flat files etc) are the primary concerns of operations personnel, post launch.
- Additional costs are anticipated through the use of specialist personnel for the live maintenance of both legacy and SOA applications.

- The migration approach suggested by this project is similar to that in use by some organisations, with the exception that the move from the legacy system is usually performed in a short cycle.
- The suggested slower migration to SOA is “favourable as the migration can happen over a longer period of time without effect on end user”.
- The suggest migration approach is noted as viable, particularly if the benefits can be proven in advance.

6.3 Medical Professionals Workshop

6.3.1 Medical Professional Survey Group Details

This group consisted of end users of the system. Workshop candidates will include medical staff from various backgrounds including ward staff, administrators and other allied medical personnel.

6.3.2 Medical Professional Workshop Goals

The goal of this workshop was to obtain the insight of medical professionals (End-Users) on the migration of a legacy application into a Service Orientated Architecture (SOA) through the addition of new value added service components. The areas of concern by this stakeholder group was primarily user centric issues, as any change which impacts their daily work routine improving it for the better or degrading the practice. Such areas were divided as User Interfaces, Training Requirements, Integration with current work practices and other barriers to adoption of the system.

6.3.3 Medical Professional Workshop Composition

The workshop consists of a presentation (Appendix H), a demonstration of the proof of concept application and then an interview.

The workshop interview focused on the following areas.

- User Interfaces: The workshop was designed to collect data on the merits of maintaining a familiar look and feel of a current system and the benefits of introducing a familiar looking set of new screens. Data on the users acceptance a new application, factors allowing for an easier adoption and how ease of use affect the users’ attitude to the system was also gathered.
- Training Requirements: Data on the difference in training for a familiar interfaced system and for a new system, resistance to adopting due to training requirements, users attitude to the system based on the effort required in training and the benefits to designing new systems

with a familiarity to the know legacy interface was gathered.

- Integration with current work practices: The workshop then gathered information on the current integration of legacy systems in work practices and the effect on the user and work practice which arises from the introduction of new software.
- Barriers to Adoption of New Software: Finally, the workshop collected information on the problems encountered with the introduction of new software from the users' perspective. This includes how users perceive new software, such as how the immediate benefit to treatment of a patient being acknowledged immediately results in a positive view of work practice changes and quicker embracement by staff. In addition, data on the impact on users of the approach taken to the design, development and release of new software and the role of adequate training in user adoption of the system was collected.

6.3.4 Medical Professional Workshop Results

6.3.4.1 User Interfaces

All participants agreed that a familiar look and feel to the new screens does allow for easier adoption of the system, with one participant stating this is the case as long as the users are familiar with and use the existing legacy system. Complexities to the work practices and some initial confusion are expected with the introduction of a new screen as part of the work practice, due to the introduction of some unfamiliarity, in an otherwise familiar user environment. 60% of participants agreed that, if the new system used a web browser interface, participants responded that adoption of the system may be simpler. 40% of participants stated that this depends on the users experience with using web browsers and the internet. Participants also responded that the user will get use to any interface given training and motivation. One participant stated that although the ease of use would affect the users' attitude to the system and thus indirectly affect their use of the system, if mandatory use was required, the system would be used regardless.

6.3.4.2 Training Requirements

If the current legacy system were extended to include the new functionality, 90% of participants agreed that little training would be required, however one participant noted that this depends on the functionality change, as sometimes small changes to familiar systems can cause difficulties, in particular if the user interacts with the system "automatically" and without fully understanding how the system works. As such, any alterations may not be expected and will cause confusion.

Participants agreed that there may be substantial training required with introduction of a new

system and that if the interfaces are "vaguely familiar" training will not be a major task, depending on the level of expertise of staff. One participant noted that with similar screen shots "staff can be less apprehensive and so more receptive to minor changes". All participants agreed that there would be substantial training required with introduction of a new system especially if the interfaces are completely new. While 80% of participants agreed with the statement "the amount of training required would affect the users' attitude to the system and thus indirectly affect their use of the system", some conflicting views were collected. One participant stated that the less training necessary the better and that the use of the system is ultimately down to the users confidence in the system itself, while another participant stated that more comprehensive the training would created more confidence in using the system.

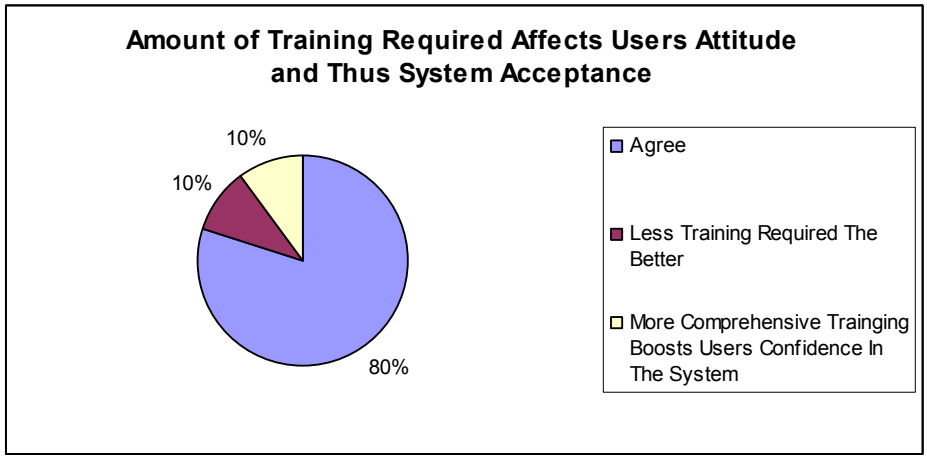


Figure 6.3: Amount of Training Required Affects Attitude and Acceptance

6.3.4.3 Integration with current work practices

60% of participants reported that while in general there is little affect of a minor upgrade on a legacy system to work practices, 40% of participants stated that this depends on power of the data. An example given was the addition of an MRSA flag. This is small piece of data integrated but has severe implications on the integration with the work practice. One participant noted that work practice change can be positive as the legacy application may have been restricting the work practices effectiveness. The majority (90%) of participants agree that the introduction of new screens in a work practice changes the process flow and increases complexity and that such change would require training and time to adopt. However one participant noted that this is down to the feature and the initial training provided, and that the lesser the complexity and closer the match to the workflow the easier the adoption would be.

All participants agree that the system should be designed around the work practices currently in use and not the other way around. Participants also pointed out that if there is an opportunity to improve

a work practice resulting in better patient care, it should be taken.

6.3.4.4 Barriers to Adoption of New Software

In discussing that changes in work practices are generally perceived as negative and users will resist change, all participants agreed but noted that this depends on the work practice and articulation of change, interface groups makes a significant difference; as does managerial enthusiasm in introducing a new system which impacts established practices.

All participants agreed that if immediate improvements to patient care were seen as the result of new software, changes in work practices are perceived as positive and the changes will be embraced. One participant also pointed out that this may be an ideal world scenario. Depending on the context of the change and the potential impact on the patient, if the change severely impacted the user in a negative manner, increasing the complexity or time taken to complete a task, the change might not be viewed favourably.

All participants agreed that training was a factor in preventing adoption of the system, sighting that training a critical mass operating the system and involved in the workflow is required for adoption to succeed, and that if the system was not intuitive or training does not provide confidence in the system, adoption is more likely to be slow.

Other Comments / View Of Migration Approach

Upon final discussion in the workshop, the participants reinforced their views, stating that adoption depends on the system, it can be "neat" if adding to a current system, especially in a multi-disciplinary environment. Training and perception on new features is important, and minimal training may be required on these new features, which could be run at a ward level in the form of "on the job" training. The initial communication of the extra features carries a lot of weight in the users attitude towards the sytem and any changes in the workflow which it accompanies.

The issue of error prevention was also discussed, one participant stated that in their experience, moving from one system to another familar looking system assistsed in the reduction of erroneous error, simply because they could quickly adopt the new systems "way of doing things".

All participants viewed the migration from legacy to SOA as positive, as one complete system would result.The prospect of a gradual move with both the legacy and SOA solution in parallel operation was well received as it allowed for the gradual training and migration of staff.

Finally, the role and attitude of the user was discussed, given the nature of medical users, which a huge variation in backgrounds, skills, age and experience, some users will always resist change on both a political level and all effects which political change invokes. This includes knock on effects

on workflow practices and the introduction of new technology.

6.3.5 Medical Professional Workshop Analysis

Upon examination of the IT Professional Workshop results, the following key points were revealed:

- A familiar look and feel of new screens allows for an easier adoption of an application.
- The use of a web browser interface would make adoption simpler, for those who regularly use the internet.
- Ease of adoption of the user interface affects the users' attitude and willingness to use a new system.
- The addition of new screens, affecting the users' workflow will cause some initial confusion.
- The expansion of a legacy system currently in use would require little training.
- Substantial training would be required with the introduction of a new system, though the use of familiar screen layouts in comparison to the current legacy system would reduce the training required.
- Users did not hold a similar view on the need for training; some users prefer as little training as possible, while others prefer substantial training, as this will increase their confidence in using the system.
- Minor upgrades to a legacy system can have profound effects on the users' workflow. For example, the introduction of an MRSA flag on a patients profile screen.
- A system should be designed around the user workflow and not the other way around. Changes in the users' workflow will take time to adopt.
- Changes to the users' workflow are generally seen as negative and are therefore barriers to a systems adoption.
- Immediate improvements to a patient's health as the result of system and workflow changes will greatly increase system adoption.
- Appropriate training, management articulation and a sense of ownership are important factors in system adoption.
- The use of an SOA system which used a familiar user interface assists in the prevention of error, typically resulting in using a new system.

- The migration approach suggested by this project was viewed as positive as it allowed for the parallel running of the legacy and new SOA systems, which would allow for the gradual migration of users.

6.4 Analysis of Combined Results

Cross analysis of each group of workshop results revealed a number of both anticipated and unexpected points of view. Differences in opinion were primarily driven by the stakeholder groups association with technology, patient care and budgeting.

All groups agreed that the suggested migration approach is viable. Their basis for this opinion differed between the overall eventual effort in migration, the fact that a slow migration was possible, allowing for gradual adoption by users and the fact that savings would be made through reusable components.

The migration from legacy systems to SOA was widely positive by technology stakeholders as expected, as IT staff are generally the first group to adopt new technologies to enhance application performance, quality and maintenance.

Managerial stakeholders were less favourable of the adoption of new technologies, surprisingly; this was more due to the fact that the technology set (SOA) is relatively new and regarded as unproven in the medical domain. It had been anticipated that budgeting would have proven a stronger factor in their reception to adopting SOA. As SOA is a proven technology by key players in industry, as documented in chapter 2 and chapter 3 and as the technology is being rapidly adopted by other industries such as finance, it would appear that medical managerial professionally are quite cautious in adopting new technologies, waiting until it has been widely accepted before considering its introduction to their organisation. In the context of the medical domain, this may be viewed as safeguarding the patient's wellbeing as opposed to a lack of forward thinking and innovation.

Comparatively, the views of medical professionals differed to an unexpected degree. It was found that some medical professionals were inline with technology professionals in their favourable approach to adopting new technologies which might add value to the process. Laboratory technicians, who dealt with third party laboratories, were favourable of SOA as a means to better integrate laboratory processing systems. Some medical professionals, including ward staff end users, were completely impartial to the technology used, their focus was on the patient's health and their view of the proof of concept application did not progress past the screens presented to them. They were generally satisfied with the concept of taking on new technologies to improve patient care, provided that appropriate training was given.

6.5 Summary

This chapter has described a series of workshops, carried out by 27 stakeholders from medical management, information technology and medical professional backgrounds. The chapter has described the design of each workshop, its target audience, goals, composition, an analysis of each group's results and a comparative analysis across all workshop groups.

The next chapter details a cost / benefit analysis on the development and deployment of SOA solutions Vs a legacy system upgrade.

7 Cost / Benefit Analysis

This chapter documents the performance and analysis of a cost / benefit analysis, performed to detail the cost / benefit of undertaking the migration approach suggested by this project. The chapter states the SOA and legacy systems used in the comparison, an investigation on how the proposed SOA system meets the organisations requirements, an examination of hard cash costs, soft cash saving, cost avoidance, benefits and finally, a comparison of these findings.

"A Cost Benefit Analysis is the analysis of an opportunity to demonstrate the benefits in cost savings in order to receive management commitment and support to implementation" [51]. Where applicable, the analysis is based upon a comparison of the SOA development vs. the patching of the legacy system.

In terms of a medical system, the deliverables of a cost/benefit analysis may be produced as:

- Hard Cash Savings: These are savings represented through the reduction and elimination of real costs. Examples of this may include hardware and software costs, licence costs, specialist support costs, calculated savings from improved drug prescriptions etc.
- Soft Cash Savings: Savings which are difficult to quantify but very real to the organisation. Such saving may include savings in resource time, improvement in patient health and better throughput in ward staff workflows.
- Cost Avoidance: Savings in costs resulting from use of the system. While difficult to quantify, examples include less inpatient time due to better treatment, less re-admittance resulting from inadequate medication prescriptions and legal saving from mitigations.

In performing a cost / benefit analysis, it is assumed that a long term strategy exists within the organisation to upgrade the legacy system to a modern architecture or patch the legacy system to allow interconnection between applications. Such applications include a new interface for lab technicians, updates to the organisations patient administration system and updates to the inpatient management system. It is anticipated that each of these systems will interact with central data repositories.

7.1 Cost / Benefit Analysis Execution

For the purpose of this project, the process of generating an analysis consists of the following stages.

- State the systems used in generating the analysis.
- State how the suggested system (SOA) meets the requirements set out by the organisation.

- List data related to hard cash costs and comparative savings for both legacy upgrade and SOA approach.
- List data related to soft cash savings.
- List data related to cost avoidance.
- Determine the benefits of each approach.
- State a conclusion based on cost / benefit comparison.

7.1.1 Systems used in Generating the Analysis.

7.1.1.1 SOA Solution

The SOA solution used for the purpse of a cost beneft analysis is defined as follows:

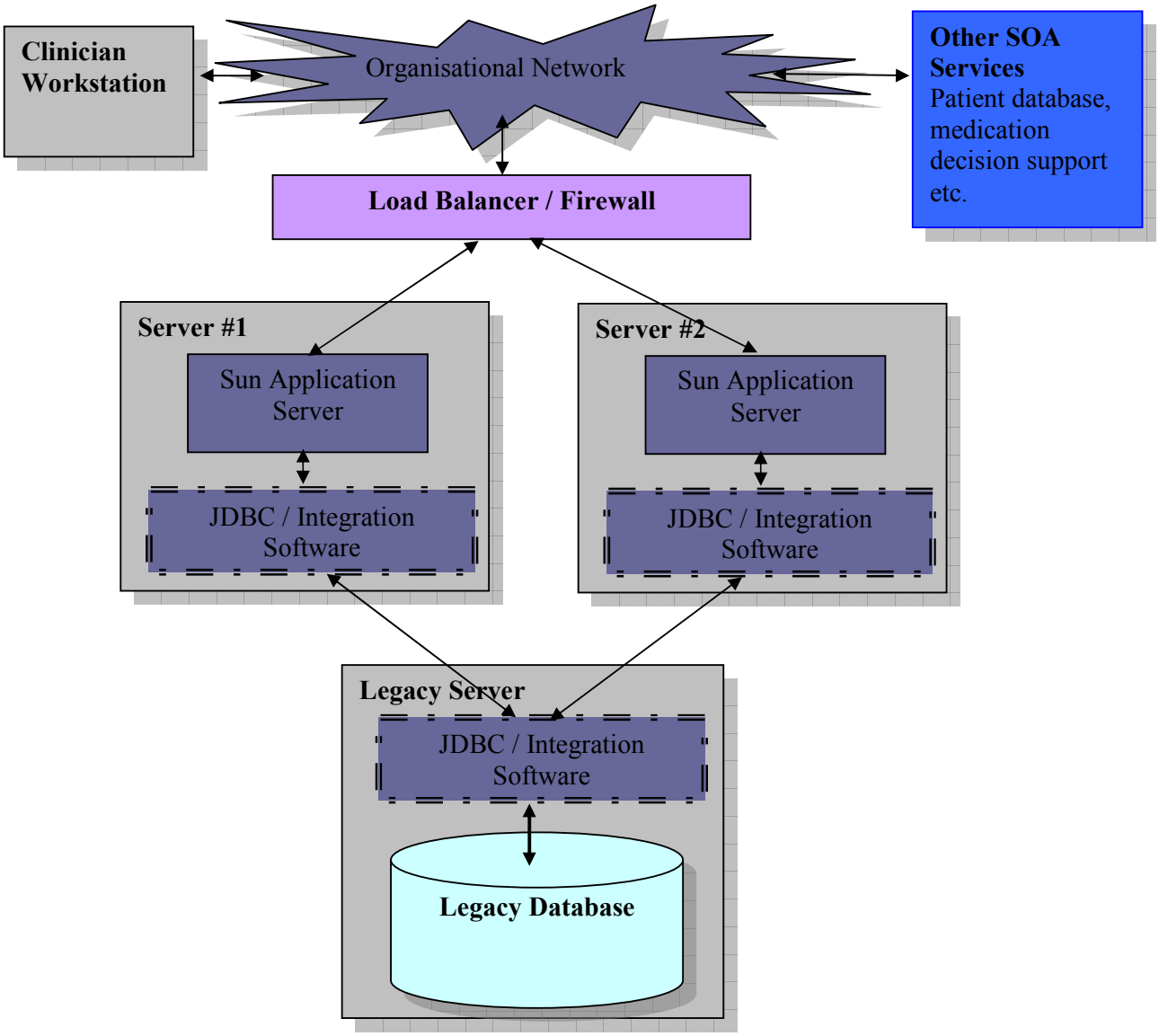


Figure 7.1: Proof of concept appication architecturre for cost / benefit analysis

7.1.1.2 Legacy Solution

The legacy solution used for the purpose of a cost benefit analysis has been defined as an extension of the current working solution, i.e. the current legacy system will be patched. To capture the requirement of interaction with patient data from other sources, it is assumed that either interaction with remote data is achieved through the use of an intermediately application or that the relevant data is loaded periodically into the legacy system.

7.1.2 Investigation of How the System meets the Organisations Requirements.

Doyle M.D. defines a guideline, executed below, to highlight the technological fit of the proposed technology into the organisation. [52] Such an analysis is designed to gain the support of stakeholders who may not immediately see the benefits of technological changes.

7.1.2.1 Categorised System Requirements

7.1.2.1.1 Efficiency

The system is proven to act efficiently through its interaction with the legacy data source, interact with other data sources in the organisation and apply business logic as defined by the system requirements. Pilot runs in the working environment would prove efficiencies in determining and checking patient medication recommendations based on the lab results retrieved.

7.1.2.1.2 Safety

Through the system design, a series of integration tests and performance tests are used to prove system safety. Primarily, the system would be tested to retrieve the correct lab result. Interaction with other systems to provide value added services are tested through pilot tests. Given the nature of the value added services implemented, judgement on medication alerts is reserved to the medical professional. In addition, the performance tests applied to replicate user requests also provide result verification, proving that not only did the system respond to the request, but responded with the anticipated result.

7.1.2.1.3 Security

Security is provided for the application on several fronts:

- Data Security: The data access on the legacy system is performed through use of a JDBC connection, this connection is secured through a set of user credentials, for which access can be limited as appropriate. In addition, as the legacy system continues to be used, the same level of security is provided as currently exists through sole use of the legacy application.
- Depending on the database server used, additional security may be provided through the

database encryption. Servers such as Oracle provide record level encryption. This may be considered in the migration / introduction of new databases, which are not part of the legacy infrastructure and as such are not required to interact with the legacy components.

- User authentication can be performed on a user level through interaction with the organisations current authentication provider. This may include LDAP systems, Vendor specific authentication system etc. This level of authentication can also be provided to interaction of other systems, such as the patient database information, medication cross-checkers etc.
- Using J2EE technology as the basis for the SOA system, user authentication can be extended to the EJB call level layer. The J2EE specification defines authentication which may be applied to method level access within the application.
- Customisations may be applied to the system to audit user activity. This may include the addition of log level auditing, recording the actions of an individual user, or database level logging, recoding the data access by the user.
- Security across the network is provided via the use of industry standard Secure Socket Layer (SSL) encryption. All data transmitted between servers and the database would be encrypted, this included both Web-service calls and responses and EJB calls.

7.1.2.1.4 Financial

- Analysis of the clinician's workflow will have estimated cost savings in cross analysis of a patient's records, medication history, lab results and safety guidelines. The use of a pilot test would further prove these savings. A pilot scheme would also allow for the observation of the applications fit into the users work practice, highlighting or eliminating concerns over the cost of training and the general changes in the time spent to carry out the work practice as a result of the new system.
- The use of log level auditing also allows for the generation of reports to show the users use of the system. Statistics gathered can then be validated against inpatient records, indicating financial factors resulting from the systems use. One such example would be an increase in the systems use (high ratio of lab result retrievals vs. inpatient admissions) and a reduction in reported calls to the lab, proving that users are availing of the system and not making unnecessary calls to the lab for result retrieval. Such practices save both time and money for the clinician and lab departments.

7.1.2.1.5 Miscellaneous

- Other requirements developed through the application and technology selection include the use of industry standard, open protocols and technology. The use of open standards greatly increases the systems capacity for expansion and integration with future generation systems.
- Basing the system on SOA allows for seem-less integration with other third party software, reducing the level of integration code to be written in future projects.
- As the data transmitted between services is based upon XML, translation into industry standards such as HL7 through the development of a translation gateway and the use of XLST is possible. While medication codes used within the sample application are customary, such translation can include the use of medical vocabularies such as IDC9, snowmed etc.

7.1.2.2 Long Term Fit with Organisational Strategy.

7.1.2.2.1 Lifespan

Given the nature of an SOA application, the system output is typically XML, responding to requests over industry standard protocols. The application itself is build on the J2EE specification and though tested on BEA Weblogic, is portable across all vendor implementations which are compliant with the J2EE specification. As such the current lifespan of the application may be seen as long terms.

7.1.2.2.2 Upgradeability

The SOA design is driven by interconnectivity and that an application is "the sum of its parts". As such, an application is not tied to any one technology or platform. J2EE applications are both upgradeable and portable. Indeed, the execution of the application in a clustered environment demonstrates the ability of the application to "grow", in order to meet the increasing requirements of the organisation.

7.1.2.2.3 Compatibility

The application is compatible and interoperable with other systems through the use of web-services. Incoming client calls are performed though an industry standard http request, with interactions between applications at the back end is performed via web services. The use of the web-services standards ensures compatibility at a transport and application level. On the application logic level, the use of and translation of medical vocabularies allows compatibility between applications. Though the sample application used with this project uses its own medication and medical condition codes, these may be translated into other

messaging standards such as IDC9, snowmed etc.

Access to data sources is provided via JDBC, with interaction between the application and the database performed using an industry standard set of SQL. In the event of database migration in the future (the migration of the legacy database to a modern database system), new JDBC drivers are used to manage the connection, application logic will work seamlessly with the new system.

7.1.2.2.4 Data Input / Output

The issues of data input / output and throughput may be addressed on three fronts:

Data Input: The capacity of the legacy database must be observed, the new system does not dictate the capacity of the legacy system or other connecting systems used. In addition, the new system does not store any additional information on the patient or medications. Meta and operational data stored by the new system, such logging information, is disposable as defined by the organisations data retention practices.

Data Throughput: The use of performance testing, as demonstrated in chapter 5 defines the safe volume of data throughput. These tests demonstrate the use of the system within the organisations requirements and the capacity for expansion in the future in line with long term organisational goals.

Data Output: While the system does not store any business data, Meta data and operational data is logged and may be used for reporting and auditing. As the system uses JDBC to connect to other data-sources, business data may be written off to databases to meet future requirements.

7.1.2.2.5 Customization

Implemented using J2EE, web-services and an SOA architecture, the system is customizable with minimal effort. Properly documented through UML and commented using Javadoc, customizing software components to meet new requirements on the systems business logic or an alteration on interconnected web-services requires minimal effort.

7.1.2.2.6 Portability of Data

As the system wraps the current legacy system and connects to other systems on a wrapped basis, the scope of data portability falls outside this consideration. However, the system itself may be expanded to provide data extraction and transform through use of the data access wrappers, should no other mechanism exist for the final migration of the legacy data to a more modern system as part of the organisations long term strategy.

7.1.2.2.7 Recoverability / Reliability

As with portability, the new system wraps the current systems and therefore the issues of data recoverability and reliability fall outside this consideration. The application itself deployed in a clustered environment, in the event of application failure, the cluster manager or load balancer will detect the failure and stop issuing requests to the application. The system will proceed to operate as normal but on a reduced capacity. As the operating environments capacity has been established based on a requirement exceeding the maximum daily throughput, the loss of capacity within limits will not pose an effect on the organisations operations.

7.1.2.2.8 Security

As detailed under system requirements, security is met on a number of fronts covering internal execution of the application, user permissions, data access and network level security. Within the J2EE framework, the application is customisable to use new security providers. External changes, such as the migration or adoption an LDAP based security provider will not result in significant redevelopment of the application. Similarly, should the modern database system to be adopted in place of the legacy system provide internal security capabilities, such as row level encryption, additional customisation of the application will not be required beyond the level of connectivity configuration.

7.1.2.2.9 Technical Support

The applications implementation and supporting software and hardware allows for both in-house and external application management. The application is J2EE compliant and may run, with little modification on a variety of J2EE containers (Examples include Bea Weblogic, Sun One Application Server, JBOSS and Oracles OC4J) and on a variety of operating systems (Windows, UNIX and variations such as Mac OSX and Linux) As a result, little specialist knowledge of the hardware, software and application is required. These factors allow for the organisation to avail of application management in the most appropriate long term manner. For example, with many more applications migrating to SOA, a centralised data centre could provide economies of scale, with UNIX administrators, database administrators and Java Application server administrators maintaining several systems and operating in close proximity. This may be a preferred option to disparate localised computer centres in terms of cost, connectivity and access to dedicated IT professionals.

7.1.2.2.10 Vendor Factors

In considering vendors and products, the openness and portability of the application works to the organisations advantage. On a long term scale, migration from one provider to another (Operating system, database or application server) is possible without a full rewrite of the application. As the external components are industry standard, large providers such as Sun, Oracle, IBM may be used to acquire the required systems. As seen on the cost benefit analyses, estimates were taken from Sun Microsystems, providing hardware, operating systems and application servers.

7.1.3 Define Hard Cash Costs

7.1.3.1 SOA Application Costs

7.1.3.1.1 Hardware costs

- Servers: To meet the operational requirements, a cluster of two applications would be employed, handing client requests on a round-robin basis. The total cost of suitable servers is approximately €46,000. This price is based on 1 Sun Blade 8000 Modular System Chassis, 2 x 2.8 GHz Opteron Model 8220 Server Modules, 32 GB (8 x 4 GB DIMMs), DDR2/667 Memory, 2 per Socket, 4 x 73 GB SAS Disk Drives, 1 x 20-Port GbE Network Express Module, 6 Power Supplies.
- Load Balancer: 1 Baracuda 340 Load Balancer, €3,399
- Network: The organisations current network infrastructure, as used by the terminals accessing the legacy system would be used; no additional costs network costs are incurred.
- Development Environments: High end PC's required for implementation amount to a total of €7000, based on current market prices.

7.1.3.1.2 Software costs

- Open source tools such as the Eclipse IDE or Sun Microsystems Netbeans IDE are used to develop the application. As a result of the suite of tools and plugins available to these IDE's there are no additional costs for developer environments.
- Licences: Licence Costs have been determined as follows:
    - Each application server costs €3048, perpetual licence, totalling €6096 (Source: Sun Microsystems)
    - €11,313 per application server, perpetual licence, totalling €22,616. (Source: JDBC drivers for IBM AS400/DB2)

7.1.3.1.3 Implemetnation costs

- Consider that the development for both the legacy wrapper layers and business logic implementation would consist of 1 project manager, 1 architect/team lead, 4 developers and 2 test engineers, operating full time for 4 months to deliver the application. This amounts to 584 man days. (21 man days per month * 7 employees * 4 months) In addition, a further 66 man days is required for post launch support. (14 man days management + 2 development resources operating full time for 1 month) This results in a total implementation man cost of 650 days.

- Based upon current contract rates at IT Jobs Watch (www.itjobswatch.com), which analyses rates offered for contracts in each job category, the implementation costs of this solution are €417,274, based on the calculated man days may be broken down as follows:

Category	Cost Per Day (Euro)	Man Days Required	Total
Project Manager	616	91	56056
Software Architect	700	84	58800
J2EE Developer	603	397	239391
Test Engineer	375	168	63000
Total			417274

Table 7.1: SOA Development - Personnel Costs

7.1.3.1.4 Annualization costs

- Maintenacnce Implementation: It is anticipated that annual implemetation costs amounting to €15379 exist through maintenance and minor upgrades (such as the integration of new business rules, refinement of the current system, update of the XML interfacing to meet backend updates) amount to , as defined below.

Category	Cost Per Day (Euro)	Man Days Required	Total
Project Manager	616	7	4312
J2EE Developer	603	14	8442
Test Engineer	375	7	2625
Total			15379

Table 7.2: SOA Maintenance - Personnel Costs

- Support Contracts
    - Annual hardware and operating system support for the purchased servers amounts to €5208 (Sun Gold Support).
    - Annual support for JDBC / middleware software (AS400/DB2 drivers) amount to €2262.

- Load balancer support / instance replacement cover amount to €749

7.1.3.2 Legacy Application Costs

7.1.3.2.1 Hardware costs

- Servers: As the current legacy hardware can be used for development, no additional servers are required.
- Network: The organisations current network infrastructure, as used by the terminals accessing the legacy system would be used; no additional network costs are incurred.
- Development Environments: High end PC's required for implementation amount to a total of €5000, based on current market prices.

7.1.3.2.2 Software costs

- As the system upgrade is based upon the current lecacy system, there are no additional software licences or other components required.

7.1.3.2.3 Implemetnation costs

- Consider that the development of additional business logic to the legacy system and the integration of new data sources, impleentation would consist of 1 project manager, 1 architect/team lead, 4 developers (2 for application logic and 2 for integration with other data sources) and 2 test engineers, operating full time for 4 months to deliver the application. This amounts to 584 mann days. (21 man days per month * 7 employees * 4 months) In addition, a further 28 man days are are required for post launch support. (7 man days management + 1 development resources operating full time for 1 month) This results in a total implemenation man cost of 612 days.
- Based upon current contract rates at IT Jobs Watch (www.itjobswatch.com), the implementation costs amount to €398,985, based on the calculated mandays may be broken down as follows:

Category	Cost Per Day (Euro)	Man Days Required	Total
Project Manager	616	91	56056
Software Architect	700	84	58800
Mainframe Developer	557	397	221129
Test Engineer	375	168	63000
Total			398985

Table 7.3: Legacy Development - Personnel Costs

7.1.3.2.4 Annualization costs

- Licences: As the solution is based on current software, there are no additional licencing

costs.

- Maintenacnce Implementation: It is anticipated that an annual implemetation costs through maintenance and minor upgrades (such as the integration of new business rules etc) amount to €14725, as defined below:

Category	Cost Per Day (Euro)	Man Days Required	Total
Project Manager	616	7	4312
Mainframe Developer	557	14	7798
Test Engineer	375	7	2625
Total			14735

Table 7.4: Legacy Maintenance - Personnel Costs

7.1.4 Soft Cash Savings

The premise of the concept application is that the business has decided to upgrade the legacy system, either through patching or migrating through the use of SOA wrapping. As such, soft cash savings such as savings in clinician time, savings from improvements in patients heath, savings in record retrieval time and savings from correct allocation of medication are not applicable for review. However, soft cash savings are applicable in areas such as system maintenance costs. These maintenance costs are based on operational maintenance, covering live issue resolution, emergency patching etc and not implementation maintenance listed as a hard cash costs. These costs may be measured in terms of operational staff time.

7.1.4.1 SOA Solution

Deployed, maintenace operations should be minimal and based on general system maintenance tasks and application server tasks. Considering the size of the proof of concept application, these tasks will consists of patch level updates which are typically applied over the course of a working day once per month. As the system interacts with the legacy data via data wrappers, there is no further maintenance tasks created for the legacy system.

7.1.4.2 Legacy Solution

Maintenance tasks should be minimal in relation to deployed application, though additional maintenace as a result of system failure of the patched system may require significant time and effort for resolution, as the legacy solution is embeded in the legacy system. In addtion, due to the lack of inherent interconnecting capabilities, additional daily batch tasks and operational tasks may be required to integrate the data accross the disparate data sources involved in the upgraded application. This is dependant on the integration solutions choosen, real time interfacing through the use of other software components such as an integration bus, or daily loading of the required data into the legacy system through nightly batch runs. In eithercase, the load on operational staff is

increased on a daily basis.

7.1.5 Cost Avoidance.

As with soft cost savings, due to the fact that the organisation has decided to integrate additional functionality regardless of the technology set, cost avoidance savings such as the costs saved from improved prescription allocation resulting in a lower level of patient re-administration and cost avoidance in litigation costs [53] are not in scope for this analysis. The following avoidance and reduction of costs are identified:

7.1.5.1 Staff Training

Through the application of consistent user interface design, matching the interface design closely with the legacy systems design, minimises the amount of staff training required. As the solution based on the legacy system consists of a legacy system upgrade, the user interface does not change, with the exception of some additional information on the patient being displayed. However, the SOA application consists of a new set of interfaces and though resembling the current leacy system, training will be required. It is estimated, from workshop discussions that cross training to use the new application would take approximately 30 minutes per user and that training could be performed in groups of appromimately 10 users.

7.1.5.2 Gradual Legacy Migration Costs

In assuming that the organisations long term strategy allows for the integration of SOA or some other form of architecture to support the move to a services based treatment model, the cost of wrapping current legacy systems is avoided, as this task is required to be performed once only per legacy system. Should the legacy system involve a detailed set of business logic rules, these rules may be wrapped as appropriate. Inline with the concept of migration through use of value added service developments, services required at the composite application business rules layer have already been performed. In summary, the development of this application also acts as an initial development of wrapped services and a prototype for the overall migration of the organisations legacy system.

7.1.6 Beneit Examination

7.1.6.1 SOA Benefits

Reduction in the gradual migration costs, as stated under section 7.1.5.2, is the principal benefit of this approach. Other benefits include staff moral and up-skilling from the introduction of new technology sets, the division of the application into separate managable enitities and the introduction of modern, low cost, industry standard hardware and software.

7.1.6.2 Legacy Upgrade Benefits

The principal benefit with this approach would appear to be the maintenance of the status quo. There are no new hardware or software costs, although software costs may be introduced based upon the mechanism selected to connect to other data sources. There development costs are in total lower than SOA and there is a smaller amount of risk as no radical changes to the legacy system are being introduced.

7.2 Cost / Benefit Analysis Results

Examination of the costs indicates an overall difference of €87,768 between the SOA migration approach and the legacy patching solution.

	Legacy	SOA
Hardware		
Serves	0	46,000
Load Balancer	0	3399
Developer Environments	5000	7000
Software		
Application Servers	0	6096
JDBC Drivers	0	11313
Personnel		
Implementation	398958	417274
Annualisation	14735	15379
Total	**418693**	**506461**

Figure 7.6: Costs Differences

Proportionally, these costs are distributed as follows:

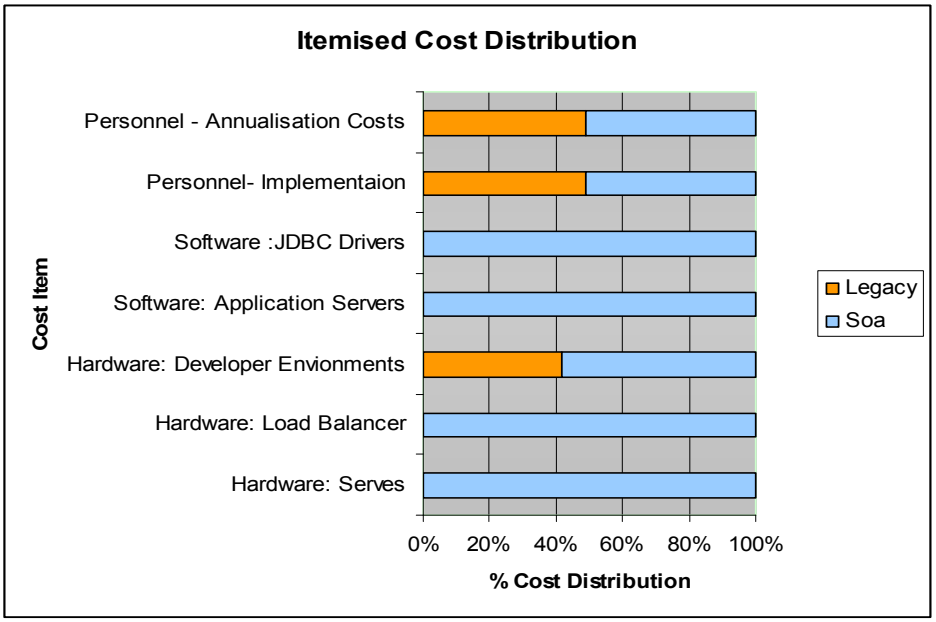


Figure 7.7: Cost Distribution

Despite the obvious domination of SOA in itemised costs, as indicated in figure 7.8, the actual cost difference is an approximate 5% overrun on the cost of patching the legacy system. This is a result of the high cost of personnel for both developments, in relation to other costs (Figure 7.9).

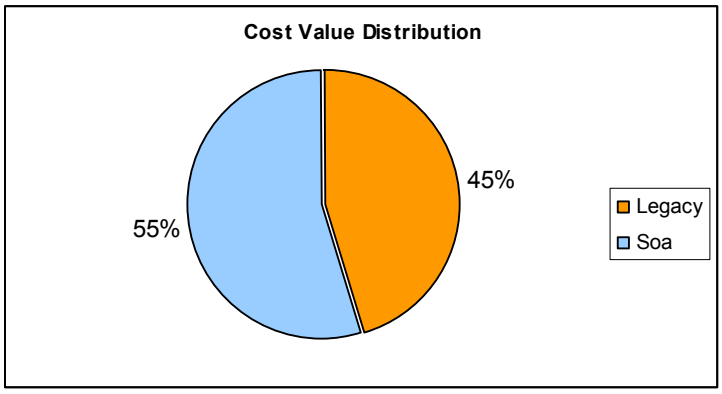


Figure 7.8 Cost Value Distributions

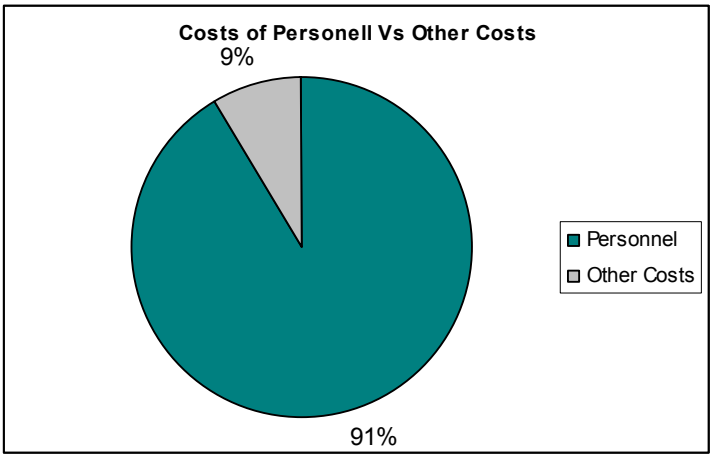


Figure 7.9 Personnel Vs. Other Costs

Given that the principal benefits of the migration approach is the development of reusable functionality for further projects and a neater long term migration, it would appear that the additional 5% costing for this implementation is worth while, should the level of risk prove acceptable.

7.3 Summary

The chapter has presented a cost benefit analysis performed to detail the cost / benefit of undertaking the migration approach suggested by this project. This included details of the SOA and legacy systems used in the comparison, an investigation on how the proposed SOA system meets the organisations requirements, an examination of hard cash costs, soft cash saving, cost avoidance, benefits and a comparison of these findings.

The next and final chapter provides an evaluation of the migration approach and a conclusion.

8 Evaluation & Conclusion

This chapter provides an evaluation of the migration through value added services approach and draws a conclusion of the viability of this approach for the medical domain. The methodology of the evaluation is explained and an evaluation of the state of the art comparison, application development exercise, performance test results, workshop results and cost benefit analysis are detailed, followed by the dissertations conclusion. The chapter and dissertation then close on an analysis of the conclusion and dissertation critiques.

8.1 Evaluation Methodology

Evaluation of the migration strategy is determined through a combination of the outcomes from the individual investigations performed throughout this dissertation.

- From examining the state of the art technologies and practices, the current migration approaches can be viewed in comparison to the suggested approach.
- The analysis of the proof of concept application development provides an insight to the work required to integrate SOA within an existing legacy infrastructure, as defined though the migration approach.
- An examination of the outcomes from performance testing allows for an establishment of the applications capacity and proof of scalability, therefore the practicality of using this migration approach.
- The outcomes of the workshops present the opinion of professionals in the managerial, IT and medical domain on the applicability of migration approach from different viewpoints.
- Finally, the outcome of the cost / benefit analysis provides a sense of realism with regards to the financial costs of this approach being adopted

8.2 State of the Art Comparison

As described in Section 3.1.3, the migration strategy investigated by this dissertation is similar to the component wrapping approach, in that legacy elements are wrapped and continue to be used. This projects migration approach is potentially executed over a longer cycle than the component approach, as systems become interoperable in a platform neutral manner. With the wrapped systems in place, the legacy systems become interoperable and business logic may be implemented or re-implemented within individual components. With iterations of development into components, the function of the legacy system becomes smaller in scope and migration becomes a less arduous task.

When a final migration of the legacy system is required, the composite database approach, in which the legacy applications are gradually rebuilt on the target platform, would complement this strategy; however a cold turkey migration of the remaining data could similarly be performed.

8.3 Application Development Evaluation

Development of the proof of concept application has proven that the migration strategy is easily implemented, enabling web-services as an extension of a J2EE application, with the wrapping of the emulated legacy data-source through JDBC. Though J2EE was the chosen platform for the development of the proof of concept application, other technologies such as .NET or the spring framework could have been applied, given the interoperable nature of SOA. In addition the migration of data from one system to another was proven to be without issue.

8.4 Performance Test Evaluation

The use of performance testing has proven that the architecture used to deploy the SOA proof of concept application performs well and is scalable. Testing on a desktop based clustered environment achieved up to 75 requests over 5 seconds without error and allows for the addition of cheap, industry standard resources by the organisation, inline with organisational growth and system usage.

8.5 Workshop Evaluation

8.5.1 Managerial Stakeholders

As shown by the medical management questionnaires, this approach is technically sound but may not be immediately viable as SOA is regarded as unproven in the medical domain. However, the strategy is viewed as a good pilot candidate to prove the technology.

8.5.2 IT & Operational Stakeholders

The migration strategy has been proven viable with IT professionals as it is similar to some existing strategies proven in industry and favourable as it has minimal impact on end users. While the technologies applied are generally viewed as more labour intensive post launch, this factor should not effect evaluation of the methodology, as the technology applied is industry standard with a high level of support. In addition the support for new development methodologies and tools should result in higher quality application resulting in an application that is easier to maintain in the long term.

8.5.3 End User Stakeholders (Medical Staff)

The migration approach is viable to end users, primarily due to the fact that the new system can look and feel like the legacy system and run in parallel with the legacy system. Familiar look and feel results in easier adoption, less training and a lower rate of errors. The fact that the legacy system and an SOA application could run in parallel allows for a more gradual adoption by end users. Additionally, if work flow change resulted from the migration strategy being employed, an improvement in patient care resulting from this change would assist in migrating staff from the legacy system to SOA.

8.6 Cost Benefit Analysis Evaluation

In terms of cost and benefit, the approach appears viable in that a difference of 5% in costs was established between the SOA implementation and the legacy system upgrade. There are also considerable potential savings on future implementation of systems which share common legacy components.

8.7 Conclusion

Upon review, the strategy of migration suggested by this dissertation appears viable, the approach is not exceptionally different from other strategies in use today, it has been proven through application development and performance testing, is noted as viable by Medical, IT and Managerial professionals and appears to be cost effective. However, SOA is a new technology and at present unproven in the eyes of medical decision makers. This factor alone detracts from its advantages in any domain. Based on this reality and the comments of managerial professionals, it may be concluded that while the suggested migration approach is technically feasible, it would not be taken onboard until the migration strategy had proved itself and SOA is more widely proven in the medical domain. To that effect, the proof of concept application would make a good candidate for both a pilot migration strategy and introduction of SOA to the organisation.

8.8 Conclusion Analysis

The conclusion that the approach is viable for the medical domain but not currently feasible is viewed as fair. The “newest and greatest” technology will always carry a sense of enthusiasm and is cyclic in process. Just as SOA and development frameworks such as spring are topical at time of writing, other technologies such as COBOL, Tuxedo, CORBA and MUMPS have had their moment in the limelight. While each has had their own particular strengths, new technologies will always be developed to deal with the changing needs of information systems.

Technology is lead by business requirements and while the service orientated architecture paradigm is suitable to the shared services model which medical organisations appear to be leaning towards today, this may not be true in the future. However, the proven advantages of SOA in other industry sectors, namely financial services and the migrations strategies similarity to other migration approaches used in industry would enforce my opinion that this approach is viable and could be fast tracked through implementation within a medical organisation given the appropriate business case.

8.9 Critique

In the event that this dissertation was to be repeated, either in part or in its entirety, it is the authors' opinion that the project would benefit from changes to the following elements:

8.9.1 Performance Testing

The process of performance testing should be changed to incorporate an iterative process, each introducing refinements to improve the applications performance. This process would eliminate simple factors affecting performance, such as the failure of the software load balancer to meet the testing requirements or the lack of database connection resources on the application server.

In addition, the cluster test environment should be reconfigured to more closely resemble the actual production environment. Such an environment would segregate the database server(s), load testing and load balancing applications and application server onto separate machines. (Figure 9.1)

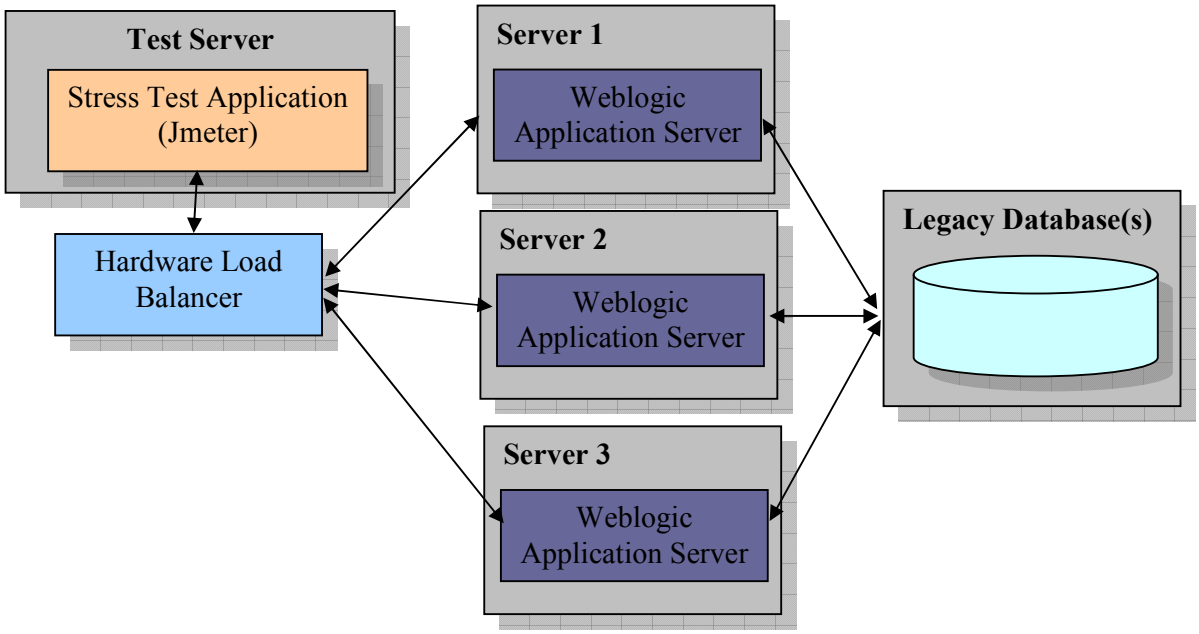


Figure 9.1: Alternative Cluster Application Environment

8.9.2 Workshop Composition

To shorten the workshop process on participants, while enabling the same detailed collection of data, a trial run of each workshop could be carried out on a subset of participants. The first set of workshops performed would result in the redevelopment of the workshop presentations and questionnaire. For example, in the performance of this projects presentations, the initial Medical Professional workshop proved difficult for participants to follow given the technical concepts being portrayed to the end used. A more graphical presentation with increased reference to the end users environment proved more effective, reducing the overall time taken for each workshop.

8.9.3 Application Development

Development access to a real legacy system would enhance the development experience. The uninhibited use of industrial JDBC drivers for the specific legacy system and the configuration of the SOA proof of concept in a live or replicated test environment would assist in proving the viability of the application and migration strategy. In addition, the execution of performance tests in this environment would provide a more realistic set of results and highlight any performance issues surrounding the use of a particular JDBC based data wrapper.

8.9.4 Cost Benefit Analysis

Upon the performance of a cost benefit analysis, costs associated with personnel could be gathered and averaged from several sources. The project schedule and costs defined for the proof of concept and legacy applications could undergo external review by managerial workshop participants. Such a review would improve the accuracy of costing each solution.

9 References

The following is a list of material referenced throughout this dissertation.

1. Sun_Microsystems. *Implementing Health Information Technology For RHIO Success*. 2005 [cited Sept 2007]; Available from: http://www.sun.com/software/whitepapers/integration_suite/rhio_healthercare_wp.pdf.
2. HSE. *National Service Plan*. Health Services Executive Publication 2006 [cited Sept 2007]; Available from: http://www.hse.ie/en/Publications/HSEPublications/FiletoUpload,2829,en.pdf.
3. Lucia, A.D., A.R. Fasolino, and E. Pompella. *A Decisional Framework for Legacy System Management*. in *Software Maintenance, 2001. Proceedings. IEEE International Conference on*. 2001. Florence, Italy.
4. Brodie, M.L. and M. Stonebraker, *Migrating Legacy Systems: Gateways, Interfaces, and the Incremental Approach*. 1995: Morgan Kaufmann Publishers.
5. Barnett, G. and M. Gilbert. *Legacy renewal strategies*. 2006 [cited Sept 2007]; Available from: download.microsoft.com/download/7/a/f/7af60b74-df6f-4636-86a9-270f6aa52be5/Legacy_renewal.pdf.
6. Lublinsky. *SOA Design: Meeting in the Middle*. 2004 [cited Sept 2007]; Available from: www.ftponline.com/javapro/2004_10/magazine/features/blublinsky/default_pf.aspx.
7. Lawrence, C. *Legacy transformation: Finding new business value in older applications*. 2007 [cited Sept 2007]; Available from: http://www.ibm.com/developerworks/webservices/library/ws-soa-adaptleg/.
8. Erlikh, L. *Leveraging Legacy System Dollars for E-Business*. 2000 [cited Sept 2007]; Available from: http://www.bridge-quest.com/news/articles/legacydollars.pdf.
9. McCormick, J. and D. Gage. *Case Study: Cincinnati Children's Hospital* 2004 [cited Sept 2007]; Available from: http://www.eweek.com/article2/0,1895,1631308,00.asp.
10. IBM. *Legacy Transformation: Attacking the problem with new competitive weapons.* 2004 [cited Sept 2007]; Available from: https://www-304.ibm.com/jct03004c/businesscenter/smb/us/en/contenttemplate/!!/gcl_xmlid=45990/.
11. Sherwood_International. *Leveraging Legacy: Strategic Tools For Survival*. 2003 [cited.
12. Bisbal, J., et al. *Legacy Information System Migration: A Bref Review of Problems, Solutions and Research Issues*. in *Joint 1997 Asia Pacific Software Engineering Conference and International Computer Science Conference*

1999. Hong Kong, China.

13. HP, I. *Healthcare Mainframe Alternatives: Know Your Options*. 2006 [cited Sept 2007]; Available from: http://h20331.www2.hp.com/enterprise/downloads/Whitepaper_121806_aap_12PM.pdf.
14. Accenture. *Service Orientated Architecture: Fit for Every Business Purpose.* 2006 [cited Sept 2007; Available from: http://www.accenture.com/Global/Technology/Service_oriented_Architecture/SOABusinessPurpose.htm.
15. Onabajo A, B.I., Jahnke J, *Wrapping Legacy Medical Systems for Integrated Health Network.* 2003.
16. HP. *HP Expands Service-oriented Architecture Services, Opens Centers to Help Customers Enhance Business Performance*. 2005 [cited Sept 2007]; Available from: http://www.hp.com/hpinfo/newsroom/press/2005/050628c.html.
17. IBM. *Providing what you need to get started with SOA.* 2005 [cited Sept 2007]; Available from: http://t1d.www-03.cacheibm.com/industries/media/doc/content/bin/SOA_Foundation_G224-7540-00.pdf.
18. Capgemini. *Service-Oriented Archiecture Supports Electronic Healht Records*. 2007 [cited Sept 2007]; Available from: http://www.dk.capgemini.com/sektorer/sundhedssektoren/?d=2BA70950-EDEF-3B1D-8ACD-691C7A6A25B4.
19. Sun_Microsystems. *Assessing Your SOA Readiness*. 2004 [cited Sept 2007]; Available from: http://www.sun.com/software/whitepapers/webservices/soa_ready.pdf.
20. Tibco. *Designng Services in an SOA using TIBCO Business Works*. 2005 [cited; Available from: http://www.tibco.com/resources/solutions/soa/soa_ref_arch_wp.pdf.
21. Microsoft. *Microsoft Advocates Real-World Approach to SOA for Increased Business Value*. 2006 [cited Sept 2007]; Available from: http://www.microsoft.com/presspass/features/2006/oct06/10-04SOA.mspx.
22. BEA. *Enabling the Business Services Lifecycle for SOA.* 2005 [cited; Available from: http://www.webservices.org/categories/technology/registry_uddi/enabling_the_business_services_lifecycle_for_soa#.
23. SAP. *Accenture And SAP to Team on New Collaberative Health Network Solution*. 2006 [cited Sept 2007]; Available from: http://www.sap.com/company/press/press.epx?pressid=6750.
24. Lockhart, H. *Demystifying Security Standards*. 2005 [cited Sept 2007]; Available from: http://dev2dev.bea.com/pub/a/2005/10/security_standards.html.

25. Cormella-Dora, S., et al. *A Survey of Legacy System Modernization Approaches*. April 2000 [cited Sept 2007]; Available from: http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tn003.pdf.
26. WRQ_Inc. *Why Screen-Interface Access is Better Than Direct Database Access*. 2003 [cited Sept 2007]; Available from: http://uk.attachmate.com/NR/rdonlyres/CB479648-80A6-4B15-93C9-A0FD6B560706/0/literature_0873.pdf.
27. ASNA_Incorporated. *Screen Scraping: A Wolf in Sheep's Clothing*. 2004 [cited Sept 2007]; Available from: www.asna.com/NR/rdonlyres/B03190DC-DB65-4205-AFC7-0331BFBD4065/0/ScreenScrapingrebuttal.pdf
28. Onabajo, A., I. Biltkh, and J. Jahnke. *Wrapping Legacy Medical Systems for Integrated Health Network*. 2003 [cited Sept 2007]; Available from: http://citeseer.ist.psu.edu/cache/papers/cs/31332/http:zSzzSzwww.netobjectdays.orgzSznode03zSzdezSzConfzSzpublishzSz..zSz..zSz..zSz..zSzpdfzSz03zSzpaperszSzws-mellszSz544.pdf/wrapping-legacy-medical-systems.pdf.
29. Ying Zou, K.A.K. *Web-based Legacy System Migration and Integration*. 1999 [cited; Available from: http://www.swen.uwaterloo.ca/~kostas/publications/conferences/C5-2000.pdf.
30. Bisbal J, L.D., Wu B, Grimson J et al. *An Overview of Legacy Information System Migration*. in *Joint 1997 Asia Pacific Software Engineering Conference and International Computer Science Conference*. 1997. Hong Kong, China.
31. Wu, b., et al. *Legacy System Migration - A Method and its Tool-kit Framework*. 1997 [cited Sept 2007]; Available from: https://www.cs.tcd.ie/publications/tech-reports/reports.99/TCD-CS-1999-16.pdf.
32. Stevenson, C. and A. Pols. *An Agile Approach to a Legacy System*. 2003 [cited Sept 2007]; Available from: http://www.skizz.biz/whitepapers/an-agile-approach-to-a-legacy-system.pdf.
33. Heuvel, W.-J.v.d. *To Adapt or Not to Adapt, That is the Question: Towards a Decision Framework for Legacy to Business Component Alignment Strategies*. 2006 [cited Sept 2007]; Available from: http://infolab.uvt.nl/~wjheuvel/publ/i-esa06.pdf.
34. Good, D. *Legacy Transformation*. 2002 [cited Sept 2007]; Available from: http://www.semdesigns.com/Company/Publications/Legacy%20Transformation.pdf.
35. Schoenbery, R., et al., *Weaving The Web Into Legacy Information Systems.* AMERICAN MEDICAL INFORMATICS ASSOCIATION, 2000. **VOL 7**(SUPP): p. 769-773.
36. Maass, M. and O. Eriksson, *Challenges in the Adoption of Medical Information Systems.*

Proceedings of the 39th Hawaii International Conference on System Sciences - 2006, 2006.

37. Van der Eijk P. *Norwegian e-Health Infrastructure based on XML. ebXML and PKI*. 2004 [cited Sept 2007]; Available from: http://www.oasis-open.org/casestudies/Trygdeetaten-A4.pdf.
38. Cruz, A.S., *E-Health Platforms And Architectures.* 2007.
39. Verma, R. and D. Cardenas. *Migration of Legacy Applications at Los Angeles County - Public Health to a PHIN and HL7(V3) Architecture*. 2005 [cited Sept 2007]; Available from: http://www.omg.org/news/meetings/workshops/HC_2005_Proceedings/05-1M_Verma_etal.pdf.
40. Sun_Microsystems. *BLUE CROSS BLUE SHIELD OF MASSACHUSETTS*. 2005 [cited Sept 2007]; Available from: http://www.sun.com/products/soa/bcbs_of_mass.pdf.
41. Oracle_Corporation. *National Health Service Shared Business Services Set to Cut Costs by 20%.* 2006 [cited Sept 2007]; Available from: http://www.oracle.com/customers/snapshots/nhs-ebs-casestudy.pdf.
42. Altman, R. *UK NHS Care Record System and Australia Health Authority / New South Wales EHR*. 2006 [cited; Available from: http://ca.sun.com/en/events/presentations/2006/singlepatient/r-altman-regional-and-national-healthcare.pdf.
43. Strass, K. *TIBCO Moves Lufthansa Cargo from a Legacy System to a Flexible Future*. TIBCO Software Inc. (Whitepapers) 2007 [cited Sept 2007].
44. Foundation, T.A.S. *Apache Beehive Documentation*. 2007 [cited Sept 2007; Available from: http://beehive.apache.org/docs/1.0.2/.
45. Foundation, T.A.S. *Struts 2 Documentation*. 2007 [cited Sept 2007; Available from: http://struts.apache.org/2.0.9/index.html.
46. Clinton A. Sprauve, I. *Implementing a Performance Test Strategy Using Open Source Software*. 2004 [cited Sept 2007]; Available from: http://www.psqtconference.com/2004east/tracks/Tuesday/Performance_Test_Strategy_Open Source.pdf.
47. Liesche, S. and S. Uhlig. *Using portal analytics with open-source reporting tools*. IBM WebSphere Developer Technical Journal 2006 [cited Sept 2007]; Available from: http://www.ibm.com/developerworks/websphere/techjournal/0609_liesche/0609_liesche.html.
48. Lurie, M. *WebSphere tuning for the impatient: How to get 80% of the performance improvement with 20% of the effort*. IBM WebSphere Developer Technical Journal 2006

[cited Sept 2007]; Available from: http://www.ibm.com/developerworks/websphere/library/techarticles/0602_lurie/0602_lurie.html?ca=dnw-705.

49. Nevedrov, D. *Using JMeter to Performance Test Web Services*. BEA Dev To Dev Articles 2006 [cited Sept 2007]; Available from: http://dev2dev.bea.com/pub/a/2006/08/jmeter-performance-testing.html.
50. Burris, S. *Mainframe to SOA: The Next Step in Data Center Evolution Opportunities and Challenge*. Z/Journal 2006 [cited Sept 2007]; Available from: http://www.zjournal.com/index.cfm?section=article&aid=328.
51. Dmytrenko, A.L. *Cost benefit analysis*. The Information Management Journal ARMA Records Management Quarterly (Association for Information Management Professionals) 1997 [cited Sept 2007; Available from: http://findarticles.com/p/articles/mi_qa3691/is_199810/ai_n8810826.
52. Christine A. Doyle, M.D. *Choosing An Automated Medical Record System*. American Society of Anesthesiologists (Website) 2004 [cited Sept 2007]; Available from: www.asahq.org/clinical/ChoosinganAutomatedMedicalRecordsSystem.pdf.
53. Wang, S.J., *A Cost-Benefit Analysis of Electronic Medical Records in Primary Care.* THE AMERICAN JOURNAL OF MEDICINE, 2003. **114**: p. 397-403.

10 Appendices

The following appendices are relevant to, or referenced throughout this dissertation:

Appendix A – Bibliography

Details of documentation not directly referenced throughout the dissertation.

Appendix B – Proof of concept application WSDL

Detailed WSDL developed as part of the proof of concept application. Chapter 4 provides a graphical representation of the web-services developed, this section provides the textual details.

Appendix C - Proof of concept application Javadoc

Javadoc for a subset of the classes used in sequence diagrams in chapter 4.

Appendix D – Managerial workshop presentation.

This presentation is referred to in chapter 6.

Appendix E – Managerial workshop questionnaire.

Questionnaire presented to managerial participants.

Appendix F – IT professional workshop presentation.

This presentation is referred to in chapter 6.

Appendix G – IT professional workshop questionnaire.

Questionnaire presented to IT professional participants.

Appendix H – Medical professional workshop presentation.

This presentation is referred to in chapter 6.

Appendix I – Medical professional workshop questionnaire.

Questionnaire presented to medical professional participants.

Appendix J – Jmeter performance test design and configuration.

Screen shots of the Jmeter test created for performance testing, as referenced in chapter 5.

10.1 Appendix A: Bibliography

The following is a list of material read but not referenced directly for this dissertation.

- Seagul_Software. *Where Legacy Meets SOA*. 2006 [cited Sept 2007]; Available from: http://www.seagullsoftware.com/assets/products/LegaSuiteBrochure.pdf
- Richardson, R., et al. *A Survey of Research into Legacy System Migration*. 1997 [cited Sept 2007]; Available from: http://coblitz.codeen.org:3125/citeseer.ist.psu.edu/cache/papers/cs/3444/ftp:zSzzSzftp.cs.tcd.iezSzpubzSztech-reportszSzreports.97zSzTCD-CS-1997-01.pdf/a-survey-of-research.pdf.
- Sun Microsystems. Assessing Your SOA Readiness. 2004 [cited Sept 2007]; Available from: http://www.sun.com/software/whitepapers/webservices/soa_ready.pdf.
- Chappell, D. *Enterprise Service Bus and Java Business Integration Infrastructure for Enterprise SOA*. 2005 [cited Sept 2007]; Available from: http://64.233.183.104/search?q=cache:558bvcK0xhkJ:gceclub.sun.com.cn/java_one_online/2005/TS-1428/ts-1428.pdf+Enterprise+Service+Bus+and+Java+Business+Integration+Infrastructure+for+Enterprise+SOA&hl=en&ct=clnk&cd=2.
- Oracle_Corporation. *Oracle Application Server 10g ESB*. July 2005 [cited Sept 2007]; Available from: http://www.oracle.com/technology/products/integration/esb/pdf/ds_esb_v10_1_2.pdf.
- IBM. *Patterns SOA with an Enterprise Service Bus in WebSphere Application Server V6*. 2005 [cited Sept 2007]; Available from: http://www.redbooks.ibm.com/redbooks/pdfs/sg246494.pdf
- Genender, J. *The Buzz About Enterprise Service Bus (ESB)*. Enterprise Open Source Journal 2006 [cited Sept 2007]; Available from: http://www.virtuas.com/files/genender%20j-f.pdf
- Bisbal, J., et al. Legacy Information System Migration: A Bref Review of Problems, Solutions and Research Issues. in Joint 1997 Asia Pacific Software Engineering Conference and International Computer Science Conference 1999. Hong Kong, China.
- Wu, b., et al. *Legacy System Migration - A Method and its Tool-kit Framework*. 1997 [cited Sept 2007]; Available from: https://www.cs.tcd.ie/publications/tech-reports/reports.99/TCD-CS-1999-16.pdf

- Stroulia, E., et al. Legacy Systems Migration in CelLEST. in Software Engineering, 2000. Proceedings of the 2000 International Conference. 2000. Limerick, Ireland.
- Onabajo, A., I. Bilykh, and J. Jahnke. *Wrapping Legacy Medical Systems for Integrated Health Network*. 2003 [cited Sept 2007]; Available from: http://www.old.netobjectdays.org/pdf/03/papers/ws-mells/544.pdf.
- Colosimo, M., et al. MELIS: an Eclipse Based Environment for the Migration of Legacy Systems to the Web in 13th Working Conference on Reverse Engineering. 2006.
- Ying Zou, K.A.K. *Web-based Legacy System Migration and Integration*. 1999 [cited; Available from: http://www.swen.uwaterloo.ca/~kostas/publications/conferences/C5-2000.pdf

10.2 Appendix B: Proof Of Concept Application WSDL

The following is a subset of the WSDL written for the development of web-services for this project. All other WSLD created can be found on the accompanying CDMedication Checker WSDL

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions name="MedicalCheckerServiceServiceDefinitions" targetNamespace="http://services" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:s0="http://services" xmlns:s1="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" targetNamespace="http://services" xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="medicationCheckerToXMLString">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="szSourceMedicaion" type="xs:string"/>
            <xs:element name="szTargetMedication" type="xs:string"/>
            <xs:element name="patientBio" type="java:PatientBIO" xmlns:java="java:ie.ak.msc.utils"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="medicationCheckerToXMLStringResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="return" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="checkMedicationMix">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="szSourceMedicaion" type="xs:string"/>
            <xs:element name="szTargetMedication" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="checkMedicationMixResponse">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="return" type="java:MedReaction" xmlns:java="java:ie.ak.msc.utils"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="medicationMixCheckerToXMLString">
        <xs:complexType>
          <xs:sequence>
```

```
        <xs:element name="szSourceMedicaion" type="xs:string"/>
        <xs:element name="szTargetMedication" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="medicationMixCheckerToXMLStringResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="return" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="checkMedicationBioMix">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="medCode" nillable="true" type="xs:int"/>
        <xs:element name="patientBio" type="java:PatientBIO" xmlns:java="java:ie.ak.msc.utils"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="checkMedicationBioMixResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="return" type="java:Vector" xmlns:java="java:java.util"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="medicationBIOCheckerToXMLString">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="szSourceMedicaion" type="xs:string"/>
        <xs:element name="patientBio" type="java:PatientBIO" xmlns:java="java:ie.ak.msc.utils"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="medicationBIOCheckerToXMLStringResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="return" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
 </xs:schema>
 <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" targetNamespace="java:ie.ak.msc.utils"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="PatientBIO">
   <xs:annotation>
```

```
        <xs:documentation>/**
 * Value Object for Patient Biological Details
 * @author Alan Kiernan
 */</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element minOccurs="1" name="Age" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="BloodPressure" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="Sex" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="Weight" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="Glucose" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="Height" nillable="true" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="MedReaction">
      <xs:annotation>
        <xs:documentation>/**
 * Value Object for medical reaction entity bean.
 * @author Alan Kiernan
 */</xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element minOccurs="1" name="CreateDate" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="MedCodeSource" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="MedCodeTarget" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="MedReactionId" nillable="true" type="xs:int"/>
        <xs:element minOccurs="1" name="ReactionDetails" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="SourceOfInfo" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="SourceCategory" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="SourceDrugName" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="SourceProductName" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="TargetCategory" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="TargetDrugName" nillable="true" type="xs:string"/>
        <xs:element minOccurs="1" name="TargetProductName" nillable="true" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
  <xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" targetNamespace="java:java.util"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:complexType name="Vector">
      <xs:complexContent>
        <xs:extension base="java:AbstractList_E_" xmlns:java="java:java.util"/>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="AbstractList_E_">
      <xs:complexContent>
        <xs:extension base="java:AbstractCollection_E_" xmlns:java="java:java.util"/>
```

```
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="AbstractCollection_E_"/>
  </xs:schema>
 </types>
 <message name="medicationCheckerToXMLString">
  <part element="s0:medicationCheckerToXMLString" name="parameters"/>
 </message>
 <message name="medicationCheckerToXMLStringResponse">
  <part element="s0:medicationCheckerToXMLStringResponse" name="parameters"/>
 </message>
 <message name="checkMedicationMix">
  <part element="s0:checkMedicationMix" name="parameters"/>
 </message>
 <message name="checkMedicationMixResponse">
  <part element="s0:checkMedicationMixResponse" name="parameters"/>
 </message>
 <message name="medicationMixCheckerToXMLString">
  <part element="s0:medicationMixCheckerToXMLString" name="parameters"/>
 </message>
 <message name="medicationMixCheckerToXMLStringResponse">
  <part element="s0:medicationMixCheckerToXMLStringResponse" name="parameters"/>
 </message>
 <message name="checkMedicationBioMix">
  <part element="s0:checkMedicationBioMix" name="parameters"/>
 </message>
 <message name="checkMedicationBioMixResponse">
  <part element="s0:checkMedicationBioMixResponse" name="parameters"/>
 </message>
 <message name="medicationBIOCheckerToXMLString">
  <part element="s0:medicationBIOCheckerToXMLString" name="parameters"/>
 </message>
 <message name="medicationBIOCheckerToXMLStringResponse">
  <part element="s0:medicationBIOCheckerToXMLStringResponse" name="parameters"/>
 </message>
 <portType name="MedicalCheckerService">
  <operation name="medicationCheckerToXMLString" parameterOrder="parameters">
   <input message="s0:medicationCheckerToXMLString"/>
   <output message="s0:medicationCheckerToXMLStringResponse"/>
  </operation>
  <operation name="checkMedicationMix" parameterOrder="parameters">
   <input message="s0:checkMedicationMix"/>
   <output message="s0:checkMedicationMixResponse"/>
  </operation>
  <operation name="medicationMixCheckerToXMLString" parameterOrder="parameters">
   <input message="s0:medicationMixCheckerToXMLString"/>
   <output message="s0:medicationMixCheckerToXMLStringResponse"/>
```

```
    </operation>
    <operation name="checkMedicationBioMix" parameterOrder="parameters">
      <input message="s0:checkMedicationBioMix"/>
      <output message="s0:checkMedicationBioMixResponse"/>
    </operation>
    <operation name="medicationBIOCheckerToXMLString" parameterOrder="parameters">
      <input message="s0:medicationBIOCheckerToXMLString"/>
      <output message="s0:medicationBIOCheckerToXMLStringResponse"/>
    </operation>
  </portType>
  <binding name="MedicalCheckerServiceServiceSoapBinding" type="s0:MedicalCheckerService">
    <s1:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="medicationCheckerToXMLString">
      <s1:operation soapAction="" style="document"/>
      <input>
        <s1:body parts="parameters" use="literal"/>
      </input>
      <output>
        <s1:body parts="parameters" use="literal"/>
      </output>
    </operation>
    <operation name="checkMedicationMix">
      <s1:operation soapAction="" style="document"/>
      <input>
        <s1:body parts="parameters" use="literal"/>
      </input>
      <output>
        <s1:body parts="parameters" use="literal"/>
      </output>
    </operation>
    <operation name="medicationMixCheckerToXMLString">
      <s1:operation soapAction="" style="document"/>
      <input>
        <s1:body parts="parameters" use="literal"/>
      </input>
      <output>
        <s1:body parts="parameters" use="literal"/>
      </output>
    </operation>
    <operation name="checkMedicationBioMix">
      <s1:operation soapAction="" style="document"/>
      <input>
        <s1:body parts="parameters" use="literal"/>
      </input>
      <output>
        <s1:body parts="parameters" use="literal"/>
      </output>
```

```
    </operation>
    <operation name="medicationBIOCheckerToXMLString">
      <s1:operation soapAction="" style="document"/>
      <input>
        <s1:body parts="parameters" use="literal"/>
      </input>
      <output>
        <s1:body parts="parameters" use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="MedicalCheckerServiceService">
    <port binding="s0:MedicalCheckerServiceServiceSoapBinding" name="MedicalCheckerServiceSoapPort">
      <s1:address location="http://localhost:7001/MedicationCheckerWebService/MedicalCheckerService"/>
    </port>
  </service>
</definitions>
```

10.3 Appendix C: Proof of Concept Application Javadoc

The following is a subset of the Javadoc for code written in the development of the proof of concept application for this project. This Javadoc covers the medication checker component. All other Javadoc for the application code can be found on the accompanying CD.

10.3.1 ie.ak.msc.medicationchecker
Class MedicationCheckerBusiness

```
java.lang.Object
  └weblogic.ejb.GenericEnterpriseBean
      └weblogic.ejb.GenericSessionBean
          └ie.ak.msc.medicationchecker.MedicationCheckerBusiness
```

All Implemented Interfaces:

java.io.Serializable, javax.ejb.EnterpriseBean, javax.ejb.SessionBean

---

```
public class MedicationCheckerBusiness
extends weblogic.ejb.GenericSessionBean
implements javax.ejb.SessionBean

Medication Checker Business Session Bean
```

Returns medication mix and treatment guide data to caller - refer to methods "toXml" for details on XML return.

See Also:

Serialized Form

Constructor Summary

MedicationCheckerBusiness()	

Method Summary

java.util.Vector	**checkMedicationBioMix**(java.lang.Integer medCode, PatientBIO patientBio) Check medication bio mix - check for valid medication bio mix.
MedReaction	**checkMedicationMix**(java.lang.String szSourceMedicaion, java.lang.String szTargetMedication) Check medication mix - check for medication mix alert if know reaction exists, return reaction details.
void	**ejbCreate**()
java.lang.String	**getStringFromDocument**(org.w3c.dom.Document doc) Generate String from given XML Document object.
java.lang.String	**medicationBIOCheckerToXMLString**(java.lang.String szSourceMedicaion, PatientBIO patientBio)

	Retrieve medication bioligal mix result, convert to XML and return as String for webservice use.
java.lang.String	**medicationCheckerToXMLString**(java.lang.String szSourceMedicaion, java.lang.String szTargetMedication, PatientBIO patientBio) Check medication bio mix - check for valid medication bio mix.
java.lang.String	**medicationMixCheckerToXMLString**(java.lang.String szSourceMedicaion, java.lang.String szTargetMedication) Generate medication mix checked XML string
org.w3c.dom.Document	**toXML**(MedReaction medReaction) Generate XML Document Object from med reaction object
org.w3c.dom.Document	**toXML**(MedReaction medReaction, java.util.Vector medBioMix) Generate XML Document obect form medreaction object and medication bio mix collection
org.w3c.dom.Document	**toXML**(java.util.Vector medBioMix) Converts vector of medication bio mix objects to a single XML string.
java.lang.String	**toXMLString**(MedReaction medReaction) Generate XML String for medication reaction object
java.lang.String	**toXMLString**(MedReaction medReaction, java.util.Vector medBioMix) Generate XML from document object.
java.lang.String	**toXMLString**(java.util.Vector medBioMix) Convert a vector of paient bioogical details medication mix results to XML.

10.3.2 ie.ak.msc.medicationchecker
Class MedReactionEntityBean

```
java.lang.Object
  └weblogic.ejb.GenericEnterpriseBean
      └weblogic.ejb.GenericEntityBean
          └ie.ak.msc.medicationchecker.MedReactionEntityBean
```

All Implemented Interfaces:

java.io.Serializable, javax.ejb.EnterpriseBean, javax.ejb.EntityBean

---

```
public abstract class MedReactionEntityBean
extends weblogic.ejb.GenericEntityBean
implements javax.ejb.EntityBean
```

MedReactionEntityBean Representation of a record on the table MED_REACTION. This class represents and maps to a record on the table MED_REACTION. The datasource is connected via the JNDI listing medcrosschecker-pb and the primary key class is java.lang.Integer. In addition to the default findByPRinaryKey, there is one custom Finder defined: findByMedReaction.

See Also:

Serialized Form

Constructor Summary

MedReactionEntityBean()

Method Summary

java.lang.Integer	**ejbCreate**(java.lang.Integer medReactionId)
void	**ejbPostCreate**(java.lang.Integer key)
abstract java.lang.String	**getCreateDate**()
abstract java.lang.String	**getMedCodeSource**()
abstract java.lang.String	**getMedCodeTarget**()
abstract java.lang.Integer	**getMedReactionId**()
MedReaction	**getMedReactionObj**()
abstract java.lang.String	**getReactionDetails**()
abstract java.lang.String	**getSourceCategory**()
abstract java.lang.String	**getSourceDrugName**()
abstract java.lang.String	**getSourceOfInfo**()
abstract java.lang.String	**getSourceProductName**()
abstract java.lang.String	**getTargetCategory**()
abstract java.lang.String	**getTargetDrugName**()
abstract java.lang.String	**getTargetProductName**()
abstract void	**setCreateDate**(java.lang.String createDate)
abstract void	**setMedCodeSource**(java.lang.String medCodeSource)
abstract void	**setMedCodeTarget**(java.lang.String medCodeTarget)
abstract void	**setMedReactionId**(java.lang.Integer medReactionId)
abstract void	**setReactionDetails**(java.lang.String reactionDetails)
abstract void	**setSourceCategory**(java.lang.String sourceCategory)
abstract void	**setSourceDrugName**(java.lang.String sourceDrugName)

abstract void	**setSourceOfInfo**(java.lang.String sourceOfInfo)
abstract void	**setSourceProductName**(java.lang.String sourceProductName)
abstract void	**setTargetCategory**(java.lang.String targetCategory)
abstract void	**setTargetDrugName**(java.lang.String targetDrugName)
abstract void	**setTargetProductName**(java.lang.String targetProductName)

10.3.3 ie.ak.msc.medicationchecker
Class MedReactionEntityBeanValue

java.lang.Object
  └ **ie.ak.msc.medicationchecker.MedReactionEntityBeanValue**

All Implemented Interfaces:

java.io.Serializable

---

public class **MedReactionEntityBeanValue**
extends java.lang.Object
implements java.io.Serializable

See Also:

Serialized Form

Constructor Summary

MedReactionEntityBeanValue()
MedReactionEntityBeanValue(java.lang.String createDate, java.lang.String medCodeSource, java.lang.String medCodeTarget, java.lang.Integer medReactionId, java.lang.String reactionDetails, java.lang.String sourceCategory, java.lang.String sourceDrugName, java.lang.String sourceOfInfo, java.lang.String sourceProductName, java.lang.String targetCategory, java.lang.String targetDrugName, java.lang.String targetProductName)

Method Summary

boolean	**equals**(java.lang.Object other)
java.lang.String	**getCreateDate**()
java.lang.String	**getMedCodeSource**()
java.lang.String	**getMedCodeTarget**()
java.lang.Integer	**getMedReactionId**()

java.lang.String	**getReactionDetails**()
java.lang.String	**getSourceCategory**()
java.lang.String	**getSourceDrugName**()
java.lang.String	**getSourceOfInfo**()
java.lang.String	**getSourceProductName**()
java.lang.String	**getTargetCategory**()
java.lang.String	**getTargetDrugName**()
java.lang.String	**getTargetProductName**()
int	**hashCode**()
void	**setCreateDate**(java.lang.String n)
void	**setMedCodeSource**(java.lang.String n)
void	**setMedCodeTarget**(java.lang.String n)
void	**setMedReactionId**(java.lang.Integer n)
void	**setReactionDetails**(java.lang.String n)
void	**setSourceCategory**(java.lang.String n)
void	**setSourceDrugName**(java.lang.String n)
void	**setSourceOfInfo**(java.lang.String n)
void	**setSourceProductName**(java.lang.String n)
void	**setTargetCategory**(java.lang.String n)
void	**setTargetDrugName**(java.lang.String n)
void	**setTargetProductName**(java.lang.String n)
java.lang.Integer	**toPK**()

java.lang.String	**toString**()

10.3.4 ie.ak.msc.medicationchecker
Class MedTreatmentGuideEntityBean

```
java.lang.Object
  └weblogic.ejb.GenericEnterpriseBean
      └weblogic.ejb.GenericEntityBean
          └ie.ak.msc.medicationchecker.MedTreatmentGuideEntityBean
```

All Implemented Interfaces:

java.io.Serializable, javax.ejb.EnterpriseBean, javax.ejb.EntityBean

---

```
public abstract class MedTreatmentGuideEntityBean
extends weblogic.ejb.GenericEntityBean
implements javax.ejb.EntityBean
```

MedTreatmentGuideEntityBean Representation of a record on the table MED_TREATMENT_GUIDE. This class represents and maps to a record on the table MED_TREATMENT_GUIDE. The datasource is connected via the JNDI listing medcrosschecker-pb and the primary key class is java.lang.Integer. In addition to the default findByPRinaryKey, there is one custom Finder defined: findByMedTreatment.

See Also:

Serialized Form

Constructor Summary

MedTreatmentGuideEntityBean()

Method Summary

java.lang.Integer	**ejbCreate**(java.lang.Integer medGuideId)
void	**ejbPostCreate**(java.lang.Integer key)
BioTreatmentReaction	**getBioGuideObj**() Create and return value object representation of the entity bean.
abstract java.lang.Integer	**getMaxAge**()
abstract java.lang.Integer	**getMaxBloodPressure**()
abstract java.lang.Integer	**getMaxWeight**()
abstract java.lang.String	**getMedCode**()
abstract java.lang.Integer	**getMedGuideId**()

abstract java.lang.Integer	**getMinAge**()
abstract java.lang.Integer	**getMinBloodPressure**()
abstract java.lang.Integer	**getMinWeight**()
abstract java.lang.String	**getSex**()
abstract void	**setMaxAge**(java.lang.Integer maxAge)
abstract void	**setMaxBloodPressure**(java.lang.Integer maxBloodPressure)
abstract void	**setMaxWeight**(java.lang.Integer MaxWeight)
abstract void	**setMedCode**(java.lang.String medCode)
abstract void	**setMedGuideId**(java.lang.Integer medGuideId)
abstract void	**setMinAge**(java.lang.Integer minAge)
abstract void	**setMinBloodPressure**(java.lang.Integer minBloodPressure)
abstract void	**setMinWeight**(java.lang.Integer minWeight)
abstract void	**setSex**(java.lang.String Sex)

10.3.5 ie.ak.msc.medicationchecker

Class MedTreatmentGuideEntityBeanValue

java.lang.Object
  └ **ie.ak.msc.medicationchecker.MedTreatmentGuideEntityBeanValue**

All Implemented Interfaces:

java.io.Serializable

---

public class **MedTreatmentGuideEntityBeanValue**
extends java.lang.Object
implements java.io.Serializable

See Also:

Serialized Form

Constructor Summary

MedTreatmentGuideEntityBeanValue()
MedTreatmentGuideEntityBeanValue(java.lang.Integer maxAge,

```
java.lang.Integer maxBloodPressure, java.lang.Integer maxWeight,
java.lang.String medCode, java.lang.Integer medGuideId,
java.lang.Integer minAge, java.lang.Integer minBloodPressure,
java.lang.Integer minWeight, java.lang.String sex)
```

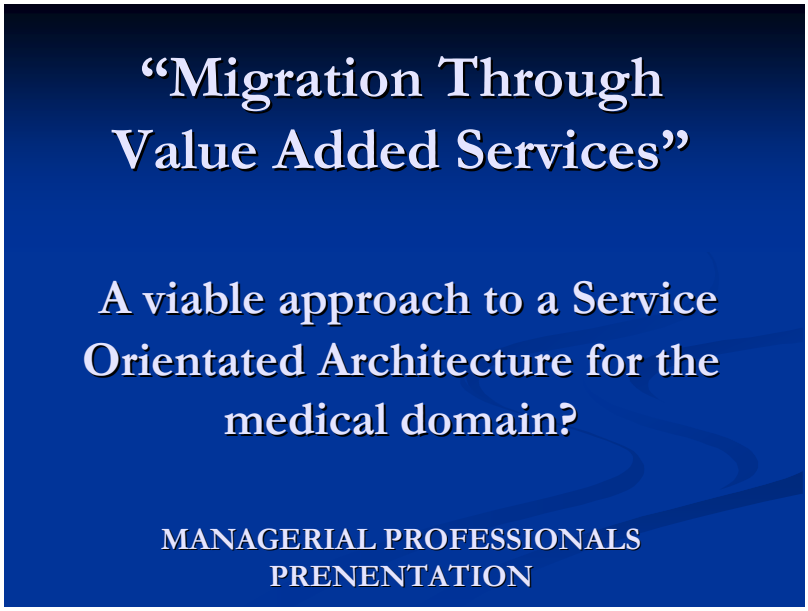
Method Summary

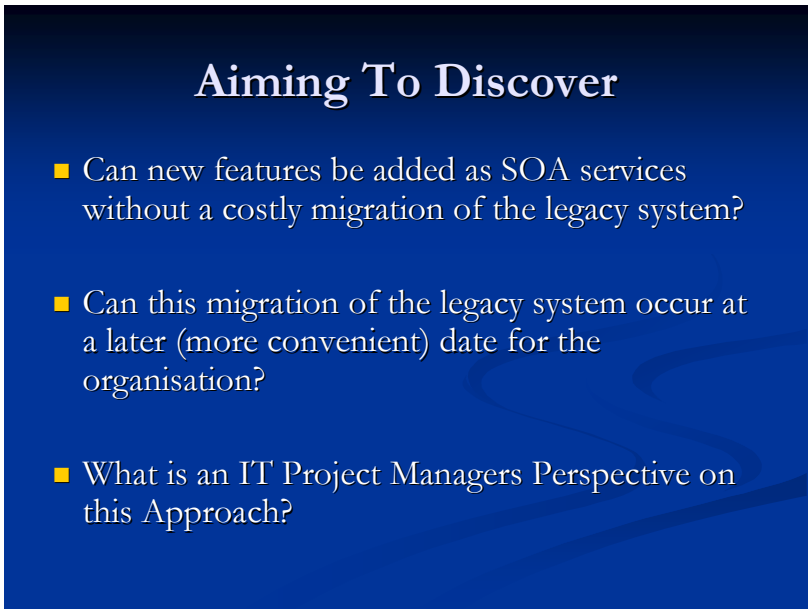
boolean	**equals**(java.lang.Object other)
java.lang.Integer	**getMaxAge**()
java.lang.Integer	**getMaxBloodPressure**()
java.lang.Integer	**getMaxWeight**()
java.lang.String	**getMedCode**()
java.lang.Integer	**getMedGuideId**()
java.lang.Integer	**getMinAge**()
java.lang.Integer	**getMinBloodPressure**()
java.lang.Integer	**getMinWeight**()
java.lang.String	**getSex**()
int	**hashCode**()
void	**setMaxAge**(java.lang.Integer n)
void	**setMaxBloodPressure**(java.lang.Integer n)
void	**setMaxWeight**(java.lang.Integer n)
void	**setMedCode**(java.lang.String n)
void	**setMedGuideId**(java.lang.Integer n)
void	**setMinAge**(java.lang.Integer n)
void	**setMinBloodPressure**(java.lang.Integer n)
void	**setMinWeight**(java.lang.Integer n)

void	**setSex**(java.lang.String n)
java.lang.Integer	**toPK**()
java.lang.String	**toString**()

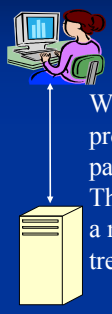
10.4 Appendix D: Managerial Workshop Presentation

The following is a copy the presentation given to managerial professionals as part of the workshops performed with this project.





Scenario



Ward staff use a program to retrieve pathology results. The result advises of a medication to use to treat an infection

It has been decided that the system should cross check the recommended medication with the patients BIO data and other medical data to check for allergic reactions, contra-indicators etc.

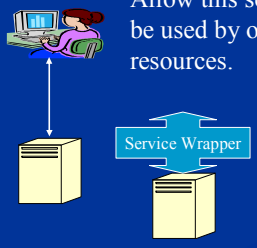
& Present these to the ward staff as part of the result.

Problem: The system does not have access to this data.

Solutions

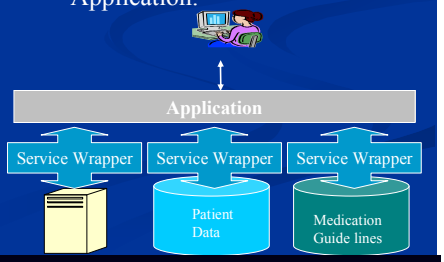
3. Leave the current (Legacy) system as is – it has a proven record at performing its current task. **It provides a particular service.**

Allow this service to be used by other resources.



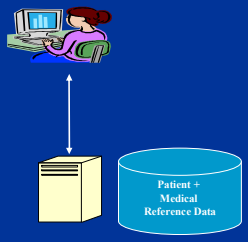
Rewrite a small application to access this service and other services to obtain the data required.

This is a Service Orientated Application.



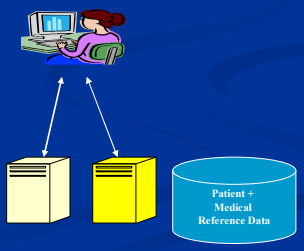
Solutions

1. Update the current (Legacy) system with the new features and provide it with access to the required data.



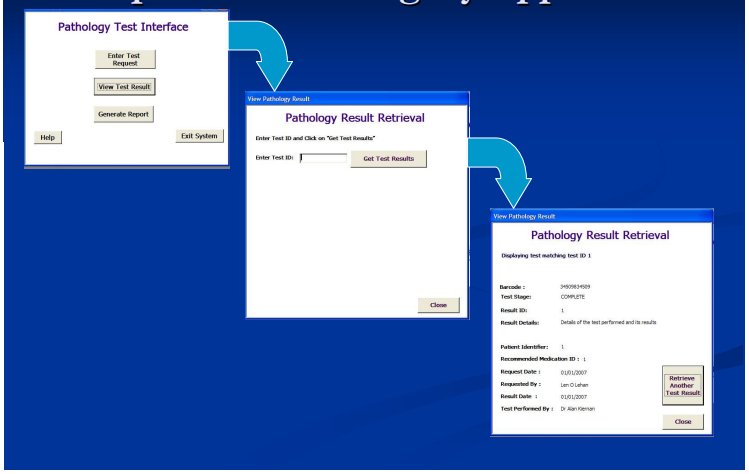
Update + Add Data

2. Replace the current (Legacy) system with the a new system to meet these requirements.

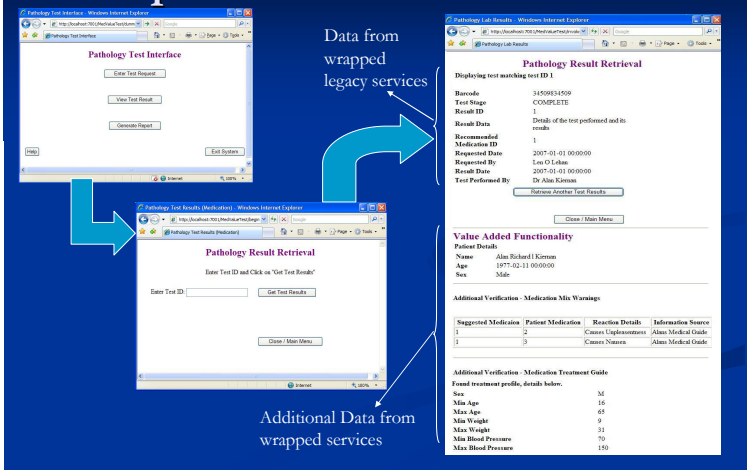


Rewrite + Add Data

Sample Scenario Legacy Application



Sample Scenario Solution - SOA



A Managerial Perspective

- Scenario – Adding new functionality to a legacy system. Can use this as an opportunity to wrap the system in an SOA application?
- "Two Birds with One Stone" ? – Initial install and configuration of an SOA platform plus extend the life of the legacy platform.
- What issues would you take into consideration in determining if this is a viable approach?

Issues To Consider

- Cost/Benefit Analysis & Comparison.
- Justification - Factors determining project progression – is the gain worth the effort?
- Project planning – how to plan time requirements for the project (SOA Vs. Legacy)
- How to manage implementation – factors involved in SOA Vs. Legacy.
- Use of socio-technical methodology.
- Barriers to adoption of the system.
- The aftermath – considerations in system maintenance.

Survey

- The survey attached quickly moves through issues which may be important to management, for both the legacy system scenario solution (solution 1) and the SOA solution (solution 3)
- Its designed to collect your opinion, there is no right or wrong answer and will be treated anonymously. (Your name is collected to prove a real person was surveyed)

10.5 Appendix E: Managerial Workshop Questionnaire

The following is a copy the questionnaire completed by managerial professionals as part of the workshops performed with this project.

Participant Details
Name
Organisation
Role
Experience
This survey is used to assess your opinion of a piecemeal migration of a legacy system to a new SOA system, solving the scenario outlined in the presentation.
Cost / Benefit Analysis & Comparison

Legacy – Calculation is straight forward, based on hours to plan and release new functionality.

SOA – Planning an entire new project?

What factor(s) would be considered in determining benefits?

At what point would benefits outweigh additional costs – SOA and platform migration justify this?

Factors Determining Project Progression

How would you go about justifying the additional costs in migrating to an SOA at this stage?

Would factors include the IT department's long term plan for migration from legacy systems and known future requirements of the system?

Would the approach be avoided to keep budgets in line with organizational planning, regardless of a possible migration from a legacy system in the future?

Project planning – How to Plan Time Requirements for the Project (SOA vs. Legacy)

Legacy – Planning time for an upgrade, ignoring future time planning. (Planning 1 part of the legacy systems future, not its migration) Agree?

The introduction of new screens in a work practice changes the process flow and increases complexity. Such change may require training and time to adopt. Agree?

SOA – Planning a new software rollout, can future budgeting be availed of based on the development of core components which will reduce future development times? (Eg. Data wrapping of Legacy System, initial ESB configuration)

How to Manage Implementation – Factors Involved in SOA Vs Legacy
Legacy – Treat project as software upgrade – little stress on resources

Would project planning and implementation include unit testing and documentation?

How would you deal with lack of documentation on system requiring upgrade?

How would you test the system? Would you use unit tests covering current core functionality which may be affected by new code?

Would you outsource to specialists if system is unknown internally?

SOA – Treat project as new software

How would you deal with the lack of documentation on system requiring wrapping? (Assign investigation resources?)

Would you use current in-house development and management practices or introduce new practices (Extreme programming / Agile methods)

Would you use SOA direction as an opportunity to outsource?

Use of Socio-Technical Methodology

Is a socio-technological approach taken to a legacy upgrade / patching project?

Should this approach be taken to a new system, even if the new system would closely resemble the legacy system?

Consider that the development would have 4 full time developers, 2 test engineers, 1 architect/team lead and 1 project manager, how many additional members of the project team would be required from a socio-technological perspective.

Would this determined by the changes to the business workflow?

Would you use personnel from different departments as part of socio-technological approach?

Barriers & Pitfalls to adoption of the System

What barriers do you foresee in adoption of the SOA based system?

Aftermath – System Operational Maintenance

Legacy- Little adoption as this is the current system, but high running costs? Agree?

SOA – Heavy initial maintenance costs but overall a better platform in terms of cost, reliability and expansion capacity? Agree?

General View of Migration Approach

What is your general view of this approach as a means of migration to SOA and away from legacy systems?

Is there concern of the costs in achieving this goal? Is this better undertaken as a separate project in parallel to development of new features for the legacy system?

Are you in favor of waiting until the legacy system can be replaced instead? (Full legacy to modern system migration)

In terms of costs and benefits – would this piecemeal migration approach be a viable option?

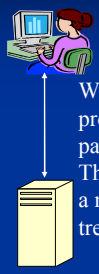
Open Feedback

Any other comments you may have.

10.6 Appendix F: IT Professional Workshop Presentation

The following is a copy the presentation given to information technology professionals as part of the workshops performed with this project.

Scenario



Ward staff use a program to retrieve pathology results. The result advises of a medication to use to treat an infection

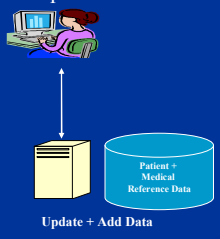
It has been decided that the system should cross check the recommended medication with the patients BIO data and other medical data to check for allergic reactions, contra-indicators etc.

& Present these to the ward staff as part of the result.

Problem: The system does not have access to this data.

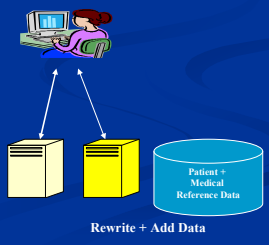
Solutions

1. Update the current (Legacy) system with the new features and provide it with access to the required data.



Update + Add Data

2. Replace the current (Legacy) system with the a new system to meet these requirements.

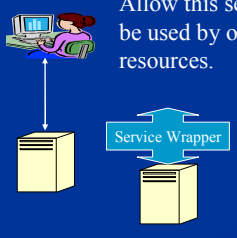


Rewrite + Add Data

Solutions

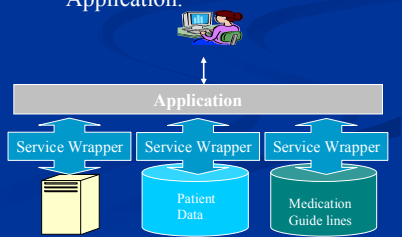
3. Leave the current (Legacy) system as is – it has a proven record at performing its current task. **It provides a particular service.**

Allow this service to be used by other resources.

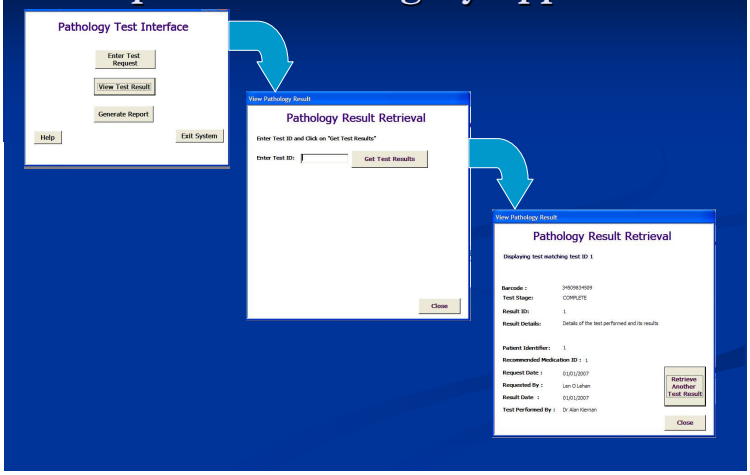


Rewrite a small application to access this service and other services to obtain the data required.

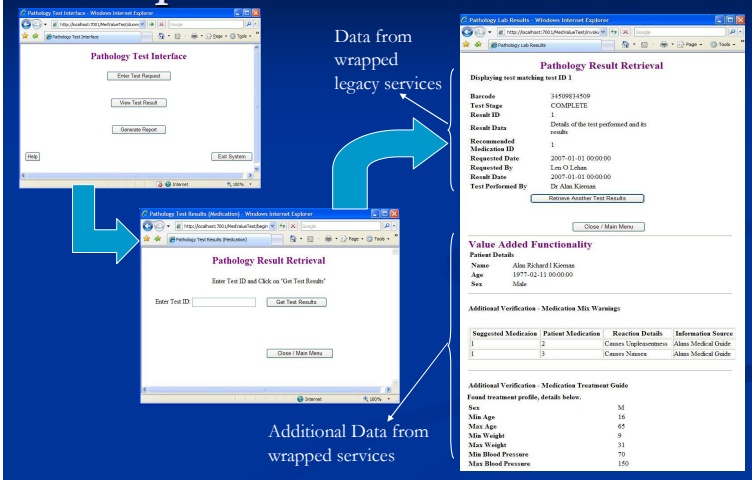
This is a Service Orientated Application.



Sample Scenario Legacy Application



Sample Scenario Solution - SOA



Example Legacy System:

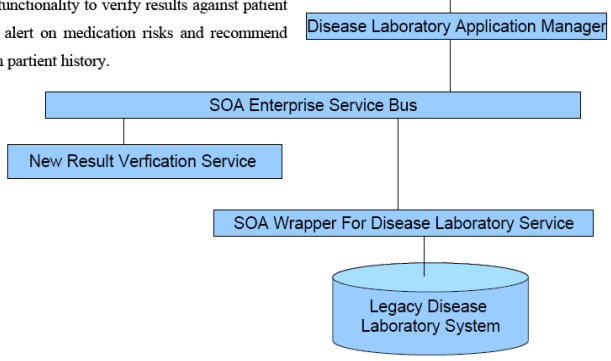
Disease laboratory processing system

Ward staff submits tests and results are issued electronically.

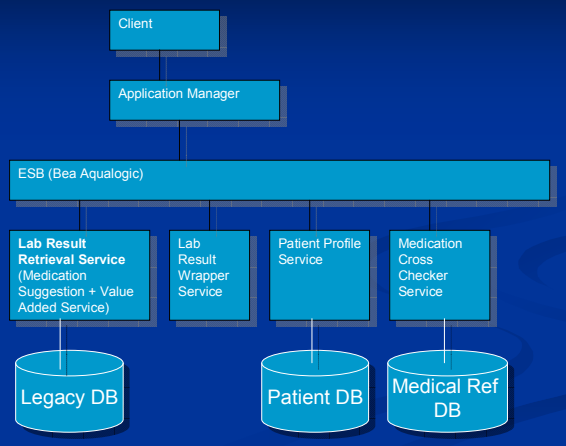
Ward staff may recall previous test results.

Proposed Value Added Service:

Additional functionality to verify results against patient history and alert on medication risks and recommend treatment on partient history.



Prototype System Architecture



Survey

- The survey attached quickly moves through the Software Development Lifecycle for both the legacy system scenario solution (solution 1) and the SOA solution (solution 3)
- Its designed to collect your opinion, there is no right or wrong answer and will be treated anonymously. (Your name is collected to prove a real person was surveyed)

10.7 Appendix G: IT Professional Workshop Questionnaire

The following is a copy the questionnaire completed by information technology professionals as part of the workshops performed with this project.

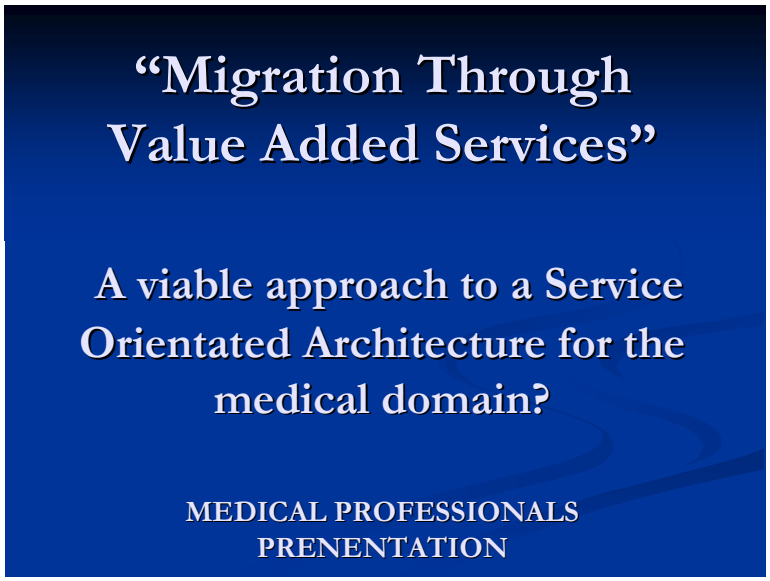
Participant Details
Name
Organisation
Role
IT / Development Experience
This survey follows the path of a system lifecycle, in terms of using a legacy system solution Vs and SOA solution to the problem outlined in the presentation.
Requirements Gathering (Planning for the system)
Legacy System Solution
Are there any unique considerations at this stage?
Does the underling technology have any effect at this stage of the lifecycle?
Is there a limitation applied to the business expert / solution through the restriction of using a legacy system?
SOA Solution
Are there any unique considerations at this stage?
Does the underling technology have any effect at this stage of the lifecycle?
Is there a sense of "we can provide more" through using SOA
Requirements Analysis
Both
Are limitations applied to the analysis of the system given prior knowledge of the target technology…?
Are there limitations under the following? (please state)
User Interface requirements?
Communications Requirements?
Load requirements?
System & Software Design
Both
Is there a fundamental different approach to designing the software for both technologies?
Legacy Solution
In building new components and/or patching software to tie in with the new components, is this an easier approach? Does this depend on the quality of existing documentation?
SOA Solution
In using new software- is this easier to design because working from the ground up?

System & Software Implementation
Legacy
Is implementation seen as a quick task because an existing system is being patched?
Is implementation greatly affected or restricted by the quality of the current code?
Are limitations in coding likely because of legacy systems?
What level of testing would be applied at this stage of the project lifecycle?
SOA
Is the wide choice of platforms and technologies and advantage?
Is implementation seen as slower because of brand new build?
Are there advantages perceived because of new technologies applicable?
Are there advantages perceived because of new software methodologies (such as Agile software, extreme programming etc) applicable?
What level of testing would be applied at this stage of the project lifecycle?
Integration & System Testing
Legacy
Is integration a simple process because an existing system environment exists?
Are there pitfalls experienced in integration of the new legacy components with the existing legacy system? If so, what are they?
What level of system testing would be performed for this application?
SOA
Are web services integration considered a more complex or less complex task than a single application?
Are the additional environment configuration tasks a deterrent from using SOA?
What level of system testing would be performed for this application?
Live Application Deployment
Legacy
What deployment process would be used in deploying the live system?
Does this require downtime?, how much (estimated)
What are the risks involved?
SOA
What deployment process would be used in deploying the system?
Does this require downtime?, how much (estimated)
What are the risks involved?
Post Implementation (Operations / Maintenance)
Legacy
What operational / maintenance issues have you experienced which are unique to legacy systems?
Are there additional costs in terms of personnel and system resources required?
SOA
What operational / maintenance issues have you experienced/would you expect which are unique to SOA systems?
Are the additional costs in terms of personnel and system resources required?

General View of Migration Approach
Treating the use of wrapping a legacy system as part of an SOA application as an approach for phased migration from a legacy system (you can replace the wrapped system at a later stage) - what do you think of this approach?
What is the user's general view of this approach as a means of migration to SOA?
Are there strengths in other approaches such as cold turkey?
In terms of personnel, system and financial resources – would this be a viable approach?
Open Feedback
Any other comments you would like to share.

10.8 Appendix H: Medical Professional Workshop Presentation

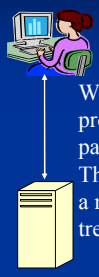
The following is a copy the presentation given to medical professionals as part of the workshops performed with this project.



Aiming To Discover

- Can new features be added as SOA services without a costly migration of the legacy system?
- Can this migration of the legacy system occur at a later (more convenient) date for the organisation?
- How are you affected by this technological change?

Scenario



Ward staff use a program to retrieve pathology results. The result advises of a medication to use to treat an infection

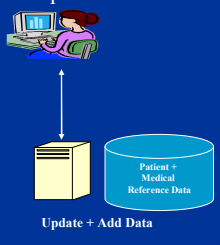
It has been decided that the system should cross check the recommended medication with the patients BIO data and other medical data to check for allergic reactions, contra-indicators etc.

& Present these to the ward staff as part of the result.

Problem: The system does not have access to this data.

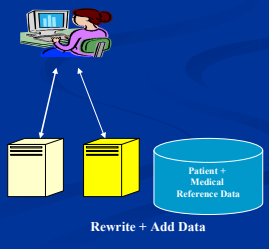
Solutions

1. Update the current (Legacy) system with the new features and provide it with access to the required data.



Update + Add Data

2. Replace the current (Legacy) system with the a new system to meet these requirements.

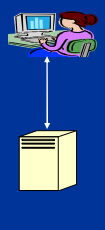


Rewrite + Add Data

Solutions

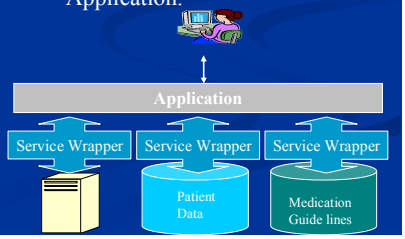
3. Leave the current (Legacy) system as is – it has a proven record at performing its current task. **It provides a particular service.**

Allow this service to be used by other resources.

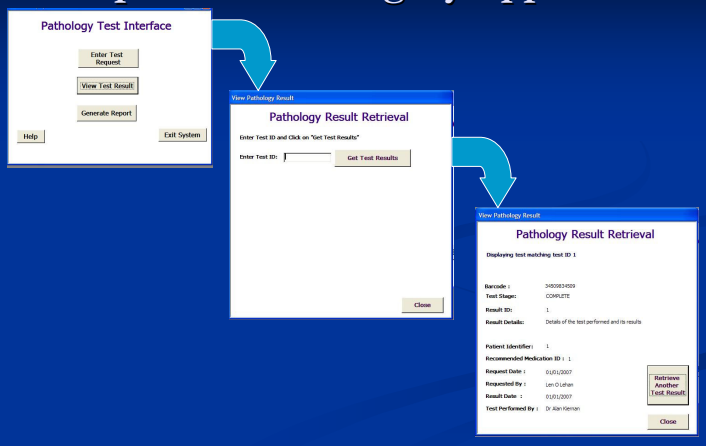


Rewrite a small application to access this service and other services to obtain the data required.

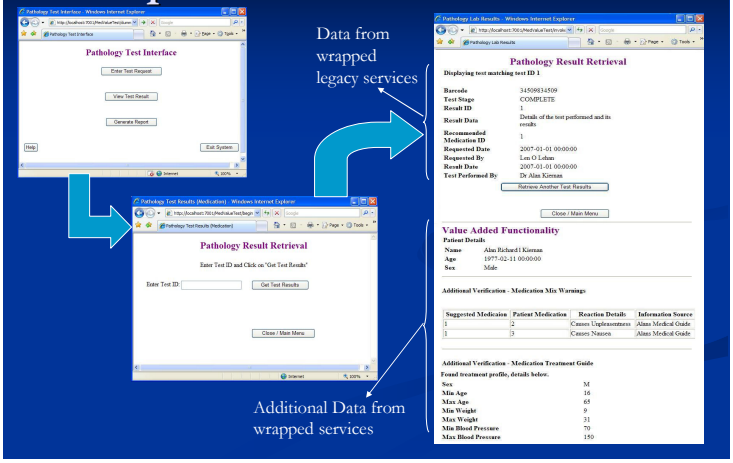
This is a Service Orientated Application.



Sample Scenario Legacy Application



Sample Scenario Solution - SOA



How Are You Affected?

- "Users prefer familiar interfaces even at the price of lesser functionality. This is especially true for new, additional functionality" (Schoenbery, Nathanson et al. 2000)
- What issues would you encounter with the introduction of a new system?

Imagine Your Current System – With Changes (Solutions 1 & 2)

- Why change the current system?
- Keep the same "Look and Feel" to the system – you know what you are dealing with.
- Is Additional functionality is easily accessed?
- There is little training required
- There is little or no change in current work practices
- The current level of care is maintained

If You Introduce A New System (Solution 2 & 3)

- Created to closely resemble the old system – a good approach?
- There is a probable learning curve in using the system
- There is a probable change to work practices to integrate the system in daily work
- The potential improvement in care – worth the effort?
- Is there a risk of mistakes from use of unfamiliar system?
- Would integration/acceptance be a simpler process if a peer was involved in the design and rollout of the application?

Survey

- The survey attached quickly moves through issues which may be important to a medical end user (such as easy of use, training etc), for both the legacy system scenario solution (solution 1) and the SOA solution (solution 3)
- Its designed to collect your opinion, there is no right or wrong answer and will be treated anonymously. (Your name is collected to prove a real person was surveyed)

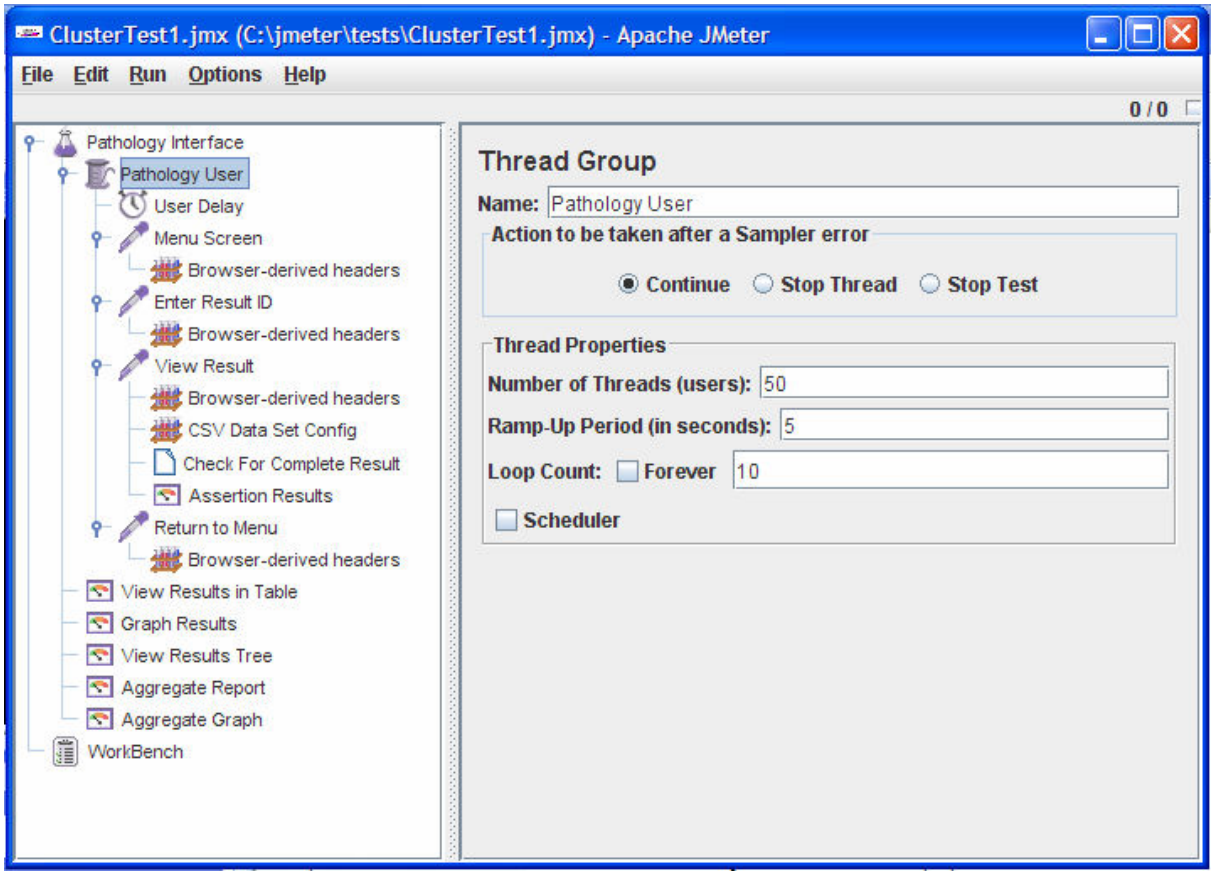
10.9 Appendix I: Medical Professional Workshop Questionnaire

The following is a copy the questionnaire completed by medical professionals as part of the workshops performed with this project.

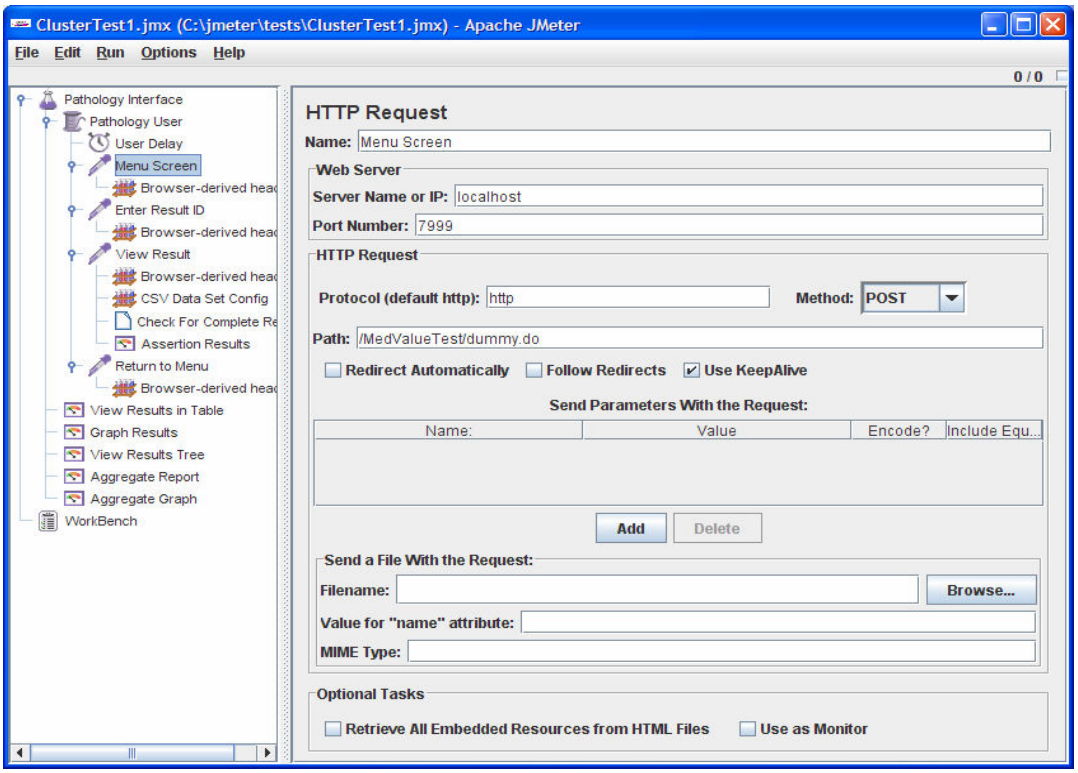
Participant Details
Name
Organisation
Role
User Interfaces
Does the use of familiar looking screens allow for an easier adoption of the system?
What would be the outcome in a case where the new system cannot provide a familiar interface (Example: one screen now contains a new set of images which the old system could not support)
If a new system used a web browser interface, would this be more easily adopted should the user be familiar with using the internet?
The ease of use will affect the user's attitude to the system and thus indirectly affect their use of the system. Agree?
Training Requirements
There is little training required with new functionality on the legacy system. Agree?
There may be substantial training required with introduction of a new system, however if the interfaces are "vaguely familiar" training will not be a major task.
There may be substantial training required with introduction of a new system especially if the interfaces are completely new.
The amount of training required will affect the user's attitude to the system and thus indirectly affect their use of the system. Agree?
Theme: Integration with current work practices
There is little affect of a minor upgrade on a legacy system to work practices. Agree?
The introduction of new screens in a work practice changes the process flow and increases complexity. Such change may require training and time to adopt. Agree?
The system should be designed around the work practices currently in use and not the other way around. Agree?
Theme: Barriers to adoption of the system.
Changes in work practices are generally perceived as negative and users will resist change. Agree?
If a treatment benefit can been immediately seen, changes in work practices are generally perceived as positive and users will embrace change. Agree?
The group / department presented with the new system would more easily accept the system I they had a sense of ownership of the system.
If training time and resources are unavailable, a lack of training may prevent adoption
Integration & System Testing
Is integration a simple process because the system environment exists?
Are there pitfalls experienced in integration of the new legacy components with the existing legacy system?
What level of system testing would be performed for this application?
Open Feedback
Other comments, points articulated by the interviewee.

10.10 Appendix J: Jmeter Performance Test

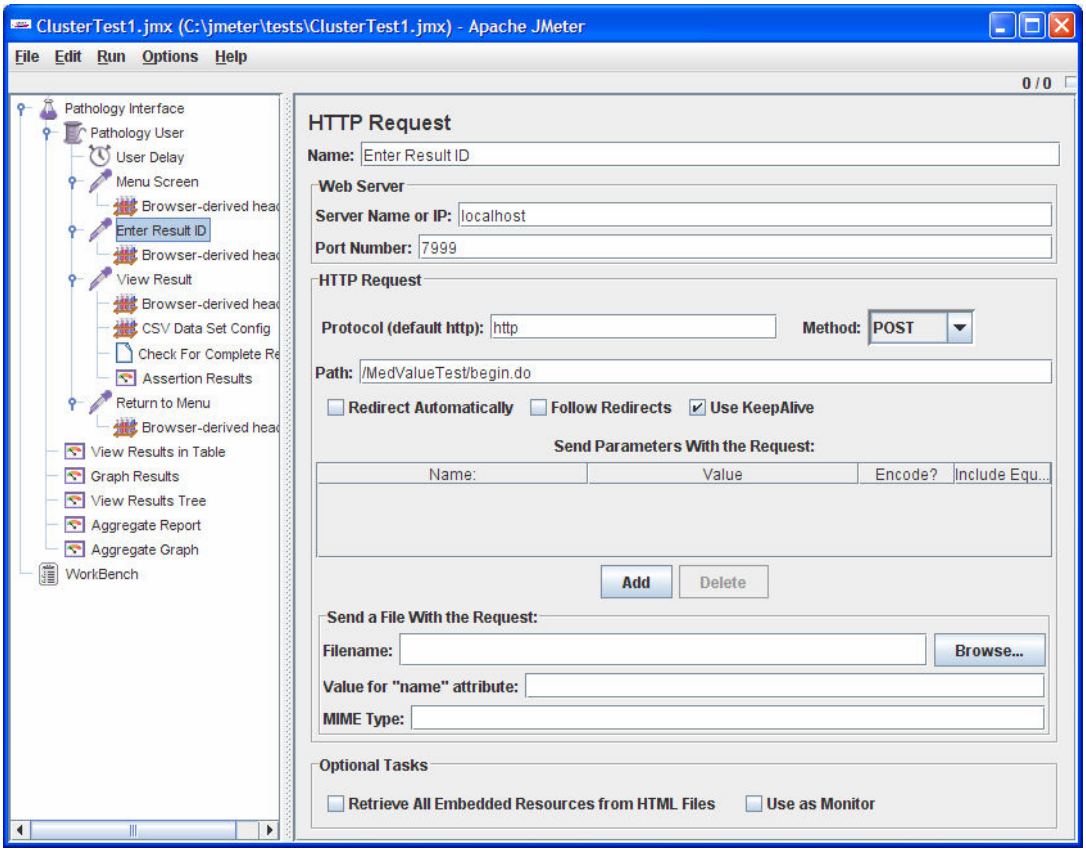
The following Jmeter test was used as the foundation for all test cases in both the single instance and clustered environment. The parameters "Number of Threads" and "Ramp-Up Period" were altered to meet individual test requirements. A copy of the Jmeter test source file can be found on the CD accompanying this dissertation.



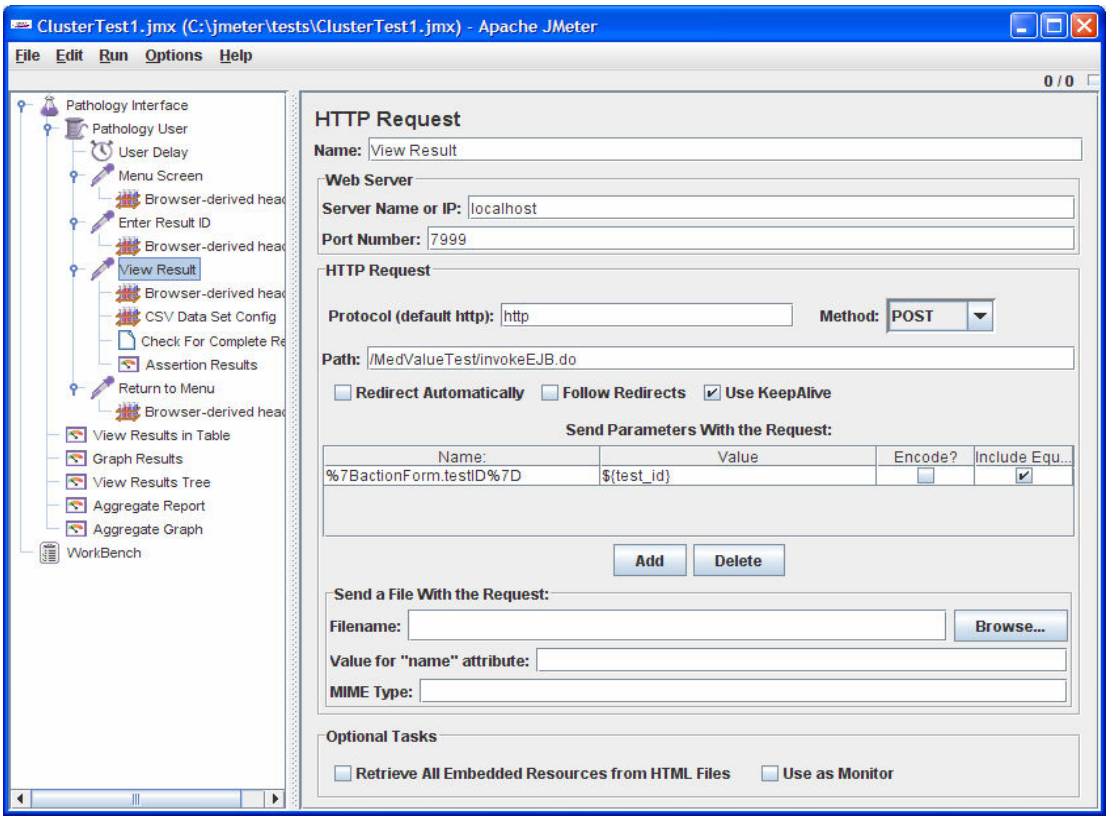
Jmeter Element K.1: Jmeter Execution Parameters



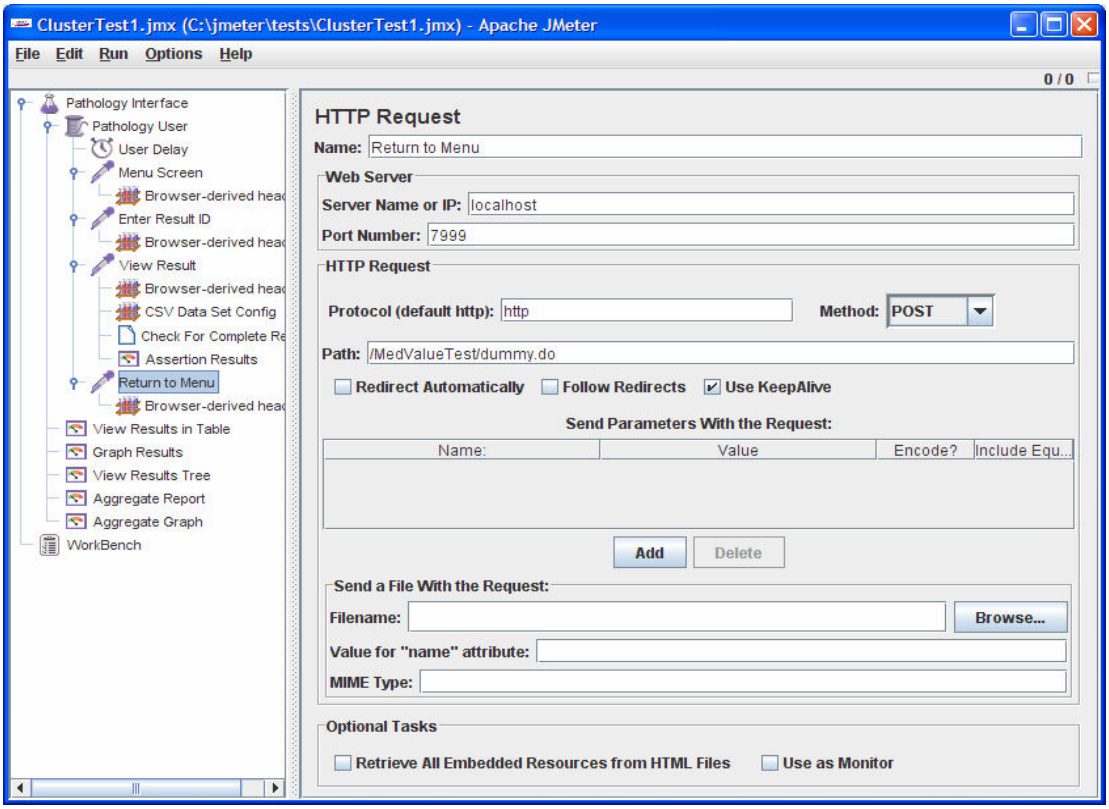
Jmeter Element K.2: Menu Screen Test



Jmeter Element K.3: Enter Result ID Screen



Jmeter Element K.4: View Results Screen



Jmeter Element K.5: Return to Menu Screen