SYSTEM ARCHITECTURE
~
ABCPROOF OF CONCEPT

SUBMITTED TO: **SUBMITTED BY:**

AUTHORIZED BY	
VERSION NO.	1.00
DATE	22-NOV-2005

Revision History

Ver.	Date	Author	Description	Reviewed By
1.00	22-Nov-2005	ABC Technolo gies	Baselined	Internal Review

Develops and installs electronic systems that eliminate the need for paper prescriptions and paper medical records and provides physicians with easy and secure online access to patient information.

Table of Contents

Table of Figures

List of Tables

1 Document Overview

1.1 Purpose and Scope

This document provides architecture overview of the EHR proof-of-concept (PoC) application. It includes various architecture views to depict different aspects of the system. The document captures and conveys the significant architecture decisions, which have been made to create the system.

1.2 Intended Audience

The intended audiences of this document are:

- The **design team** of ABC and ABCCorporation
- The **development** team of ABC
- The **deployment team** for the PoC application
- **Other stakeholders**, who would like to get a technical overview of the system

1.3 References

Following documents were referred in preparing this document:

- SRS document provided by ABCto ABC.
- high-availability(HA) framework developer documents.

2 Introduction

2.1 System overview

The set of functionality being developed under this application development effort is a narrow subset of the **EHR(Electronic Health Record) health record module of the Medical Management Platform solution of MHS.**

The following sections address the critical business considerations and goals of the system along with the details of the technical and functional components.

2.2 Goals of the Architecture

The key goals of the architecture are:

- Demonstrate the understanding of HA framework by appropriate use of prescribed HA components in the application
- Demonstrate the understanding and expertise in Spring framework and Hibernate
- Demonstrate use of AJAX technology

2.3 Notes/Assumptions

Following are the key notable points.

- HA framework components will be used for Logging, Security, Rule Engine functionality
- AJAX implementation will be done only for one application feature only

2.4 Out of Scope Activities

- The architecture won't address features which have been excluded from the scope of the PoC, i.e. Audit control and history maintenance, multi-platform testing, localization, internationalization and compliance with HIPPA and Sarbanes-Oxley regulatory standard

3 Architecture Principles

3.1 Overview

The architecture for the PoC will leverage features provided by Spring framework, Hibernate and HA components to develop a flexible, extensible and scalable architecture. The architecture principles mentioned here have been derived from the industry standard architecture best practices.

3.2 Principles

S.No.	Principles	Description	Implementation
1.	Simple multilayered architecture with loosely coupled layers	▪ Multilayer architecture separates "concerns" of an application into different layers. Layers isolate and contain the impact of changes thereby increasing maintainability ▪ Loose coupling further increases the maintainability and flexibility be reducing the dependencies between layers to a minimum	✓ Business layer hosted in Spring IoC container ✓ Use of Spring IoC and interface based access to achieve loose coupling between layers
2.	Use of (Aspect Oriented Programming)AOP for decorating functionality	▪ AOP abstracts out crosscutting concerns of the application into separate modules which can be added dynamically onto the functionality in a configurable way ▪ AOP removes clutter in the code by abstracting out code not related to core business functionality	✓ Spring AOP will be used to for logging and transaction management
3.	Multi level Security	▪ Multilevel security improves security by building in redundancy and allows business layer to be called by a different client ▪ It is assumed that the presentation layer will be the only client for the service layer which can be trusted.	✓ Form based authentication ✓ Use of Spring Handler Interceptor for authorization in presentation layer ✓ Security service based upon HA Security component

Table 1: Architecture Principles

4 System Architecture

4.1 Application Architecture

The application architecture section describes the various components used in the application. These components are logically grouped under different layers.

4.1.1 Overview

EHR PoC architecture is a multilayered architecture with the following layers:

- Presentation Layer
- Business Layer
- Data Access Object Layer
- Data Object Layer
- Framework Layer

Components depicted in the application architecture cater to all the business services and the technical services mentioned in the functional architecture. Application architecture diagram is as follows.

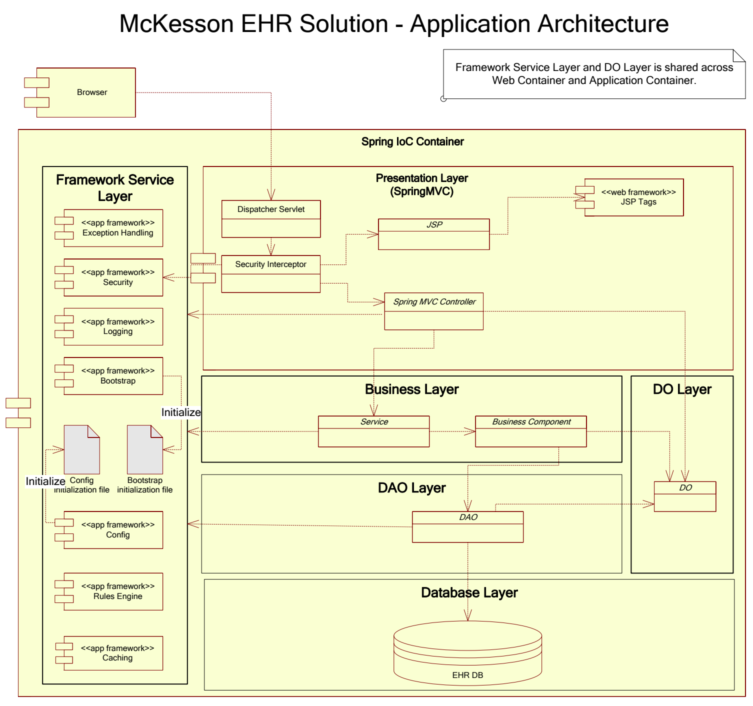
McKesson EHR Solution - Application Architecture
Framework Service Layer and DO Layer is shared across Web Container and Application Container.
Browser
Spring IoC Container
Framework Service Layer
<<app framework>> Exception Handling
<<app framework>> Security
<<app framework>> Logging
<<app framework>> Bootstrap
Initialize
Initialize
Config initialization file
Bootstrap initialization file
<<app framework>> Config
<<app framework>> Rules Engine
<<app framework>> Caching
Presentation Layer (SpringMVC)
Dispatcher Servlet
Security Interceptor
JSP
<<web framework>> JSP Tags
Spring MVC Controller
Business Layer
Service
Business Component
DO Layer
DO
DAO Layer
DAO
Database Layer
EHR DB

Figure 1: Application System Architecture

Each layer has set of components in it. All the key components of the architecture are described in the table below:

Component	Description
Html/Images	Html pages and Images represent the static content.
JSP/TagLib/ Java Scripts	The JSPs represent the user interface (screen that accepts and displays data/information) of the system.
Spring WebMVC	Provides MVC framework for the presentation layer.
Service Façade	The service façade provides coarse grained business functionality by orchestrating fine grained business service components in a transaction oriented environment.
Business Components	Business components provide fine grained business functionality by utilizing underlying DAO service functionality. They provide the basic building block for the business service.
Data Access Objects	They provide access to the underlying data store. DAO design is based upon Hibernate O/R framework.
Data Objects	DOs are utilized to carry state between different layers. They are serializable to ensure their usage in a distributed deployment environment context.
Framework Components	Framework components provide framework services like security, logging, caching etc which are required by all the components of the application.
EHR Database	Stores the EHR data namely Member data, User Profile information, Admin data and reference data required by the EHR application.

Table 2: List of Application Components

4.1.2 Presentation Layer

The Presentation layer manages the communication between user and the system.

It comprises of front-end components like static HTML/Images, Java Scripts, Tag Libraries, and JSPs. The layer also contains spring web MVC framework. The presentation layer renders application content to the user and receives inputs from the user.

4.1.2.1 Principles

S.No.	Principles	Comment
1.	Usage of composite JSP	Composite JSPs will be used to render various aspects of the screen. *<jsp:include>* tag will be used to render the various JSPs and thus reconstruct the final screen.
2.	Usage of *Spring Theme feature*	Spring MVC Themes feature will be used to manage the visual styling of the application including CSS, background images etc.
3.	Usage of JavaScript	JavaScript will be used to enable front-end validation. All validations

S.No.	Principles	Comment
		will be duplicated on the server side. AJAX will be used to implement certain select features.
4.	Usage of *Off the Shelf* component for pagination	A standard tag library DisplayTag will be used for pagination.
5.	Usage of *Spring Handler Interceptors for* authorization	Handler Interceptor will do authorization using HA security component.
6.	All static messages on HTML/JSP page will be rendered from resource bundle.	This will ensure a smooth transition to localization/internationalization even though at present it's not planned. MessageResource provided by Spring framework will be used for the purpose.
7.	Tag libraries will be used for the presentation tier.	Separation of view and logic.
8.	Spring IoC will be used to invoke business services in the Controller classes	This will ensure true loose coupling between presentation layer and business layer.

Table 3: Presentation Layer Principles

4.1.2.2 Components

4.1.2.2.1 Dispatcher Servlet

Dispatcher Servlet is the Servlet provided by Spring MVC framework. It accepts all requests coming into the application and dispatches them to appropriate controllers based upon configuration information. It is also responsible for rendering correct views on to the browser based upon the view name provided by the controller.

4.1.2.2.2 Controllers

Controller is a part of the MVC pattern which has been used in the presentation layer of this application. In a multi-layered web application controllers provide access to the application behavior i.e. the business layer. They interpret user input, invoke appropriate business operations to execute user actions and provide data for the views to be rendered to the user. Spring provides different types of controllers of which two will be used in this application:

- SimpleForm Controller: This is a concrete implementation which provides various features to assist in form processing. Controllers for all scenarios where the JSP contains a form to be processed, e.g. Create User, Edit User etc will be sub classed from SimpleForm controller.
- MultiAction Controller: MultiAction controller is capable of mapping request to method names and then invoking the right method for a request. The distinct advantage of this controller is that it can be used to effectively reduce the number of controller required in an application. For each module a subclass of MultiAction controller will be used to handle requests which do not involve a form submission e.g. Search Member, different member views etc.

4.1.2.2.3 Security Interceptor

Security Interceptor is a spring provided Handler Interceptor which will intercept all the requests being made to the any controller and perform an authorization check. If the check fails, the request is redirected to the login page.

4.1.2.2.4 AJAX Components

AJAX stands for "Asynchronous JavaScript and XML" and it uses an asynchronous request servicing model based upon XMLHttpRequest object of the browsers to provide improved user experience in web applications. AJAX implementation will involve following components:

- Client Side Components: Client side AJAX components will be set of JavaScript functions which will be used to send user request asynchronously to the server, receive the XML response based from the server and render the response on the browser.
- Server Side Components: A separate controller will be used on the server side to service AJAX requests. A utility method will be used to provide a simple XML representation of the DOs. Because AJAX is being used for demonstration purpose only and a single feature will only be AJAX enabled, all the DOs are not being designed for XML generation.

Because AJAX implementation is limited to just one feature, use of open source frameworks like DWR has not been considered.

4.1.3 Business Layer

Business layer provide coarse grained services which will be accessed by the presentation layer.

4.1.3.1 Principles

S.No.	Principles	Comment
1.	Business services will be stateless	Stateless services provide better scalability and maintainability
2.	Service Facades will provide coarse grained business functionality by orchestrating fine grained business components	
3.	Transactions will be managed at the Service Façade level. Declarative transaction management provided by Spring framework will be used for the purpose.	
4.	Business components will not initiate or commit transactions. They can only participate in a transaction.	Moving transaction specific dependencies out of business components will promote their reusability in higher layers.
5.	A business component in one domain will not call a business component in another domain directly. All functionality outside the domain will have to be accessed through the	This will ensure loose coupling between the domains which is essential for maintainability of the business layer

S.No.	Principles	Comment
	service façade for the domain	
6.	Spring IoC will be used to decouple DAO layer from the business layer. HA provided DAORegistry will be used for instantiating DAO classes.	Each business components will have a property called daoRegistry which will be injected through setter injection

Table 4: Business Layer Principles

4.1.3.2 Components

4.1.3.2.1 Business Service Interface

Business service interface will be used to publish the service contract. The presentation layer will access the service through the interface only.

4.1.3.2.2 Business Service Facade

Business services will use façade design pattern to provide coarse grained business functionality by using the business components as the basic building blocks. The service façade will implement the service interface. Service façade will have following responsibilities:

- The business service facades will be responsible for providing business operations on a certain business domain which are required by the clients (in this case the presentation layer)
- The business service facades will be responsible for the aspects like transaction management and authorization for the service
- The business service facades will be responsible for aggregating Domain Objects returned by the business component operations into DTOs which are required by the presentation layer

4.1.3.2.3 Business Components

Business components will be implemented as POJOs. They will not be transaction aware. They will interact with the DAO layer to get business specific data and will offer fine grained business logic. They will not be exposed outside the business layer. They will have following responsibilities:

- The business components will be responsible to provide operations that execute the business rules related to a certain domain/sub domains (Member, Admin, Claim) within the application
- The business component’s operations will be responsible to encapsulate the business rules like calculations, validations etc associated with business domain that it manages
- The business components are responsible to encapsulate the rules related to the constraints on the domain model that it manages
- The business components will use the HA business rule framework to externalize the business rules. The business rule framework externalizes the rules and makes easy to maintain them for any changes

4.1.4 Data Access Object Layer

4.1.4.1 Principles

S.No.	Principles	Comment
7.	Each DAO class will extend HA framework provided HibernateDaoSupport class.	
8.	Each DAO class will publish its functionality by an interface, the interface in turn extending the DAO super interface.	
9.	DAO components won't initiate any transaction. They can only be part of a transaction initiated by the business layer components.	
10.	DAO will be implemented as singleton.	DAO classes should be thread-safe

Table 5: Data Access Layer principles

4.1.4.2 Components

4.1.4.2.1 DAO

- The data access objects will be responsible to provide an interface and operations for persisting and managing the data for a certain domain e.g.: Admin domain includes User, Role and Permissions
- The data access objects will be responsible for performing the CRUD operations on behalf of its associated data objects that it manages
- The data access objects will also execute any special queries (joins, summary info) for the data required by the business components to perform the required business logic
- The data access objects will be responsible to populate the related objects of a data object as required by the business component requesting the data e.g.: The data access component will be responsible to populate the related objects like the list of permissions in a role domain object

4.1.5 Data Object Layer

4.1.5.1 Principles

S. NO.	Principles	Comment
1.	All DO classes will be serializable POJOs.	This will ensure over the wire transfer capability in a distributed deployment environment.
2.	Each layer will communicate information by means of DO	

	objects only. There won't be any other means of data transfer between different layers.	

Table 6: Data Object Layer Principles

4.1.5.2 Components

DOs are pure value objects that encapsulate the data for a business entity. These objects carry the state of the system across different layers of the system. These objects are used by other components to transfer data and to operate on the data encapsulated in them. There are two types of DOs:

- Domain Objects: These objects directly map to tables in database and their attributes map directly to columns in the corresponding database tables or to other related domain objects. These objects are used for O/R mapping.
- Data Transfer Objects: Data transfer objects are created to satisfy the needs of coarse grained service interfaces. They either aggregate multiple unrelated domain objects or represent a small subset of a large domain object which is expensive to instantiate. In the former case, Data transfer objects are assembled in the service façade.

4.1.6 Framework Service Layer

These services support the business services in satisfying the technical requirements of the application like security, logging, caching etc.

4.1.6.1 Components

4.1.6.1.1 Logging

Logging service is used in enterprise applications to enable error tracing and debugging in production environment.

As described in the Architecture Principles section, logging will be added on to the functional behaviour using Spring AOP. Logging service provided by HA framework will be used as the logging implementation.

4.1.6.1.2 Security

The security in the application will be multi level security.

In the web layer Spring Handler interceptors will be used to secure access to all the resources. Views will be rendered based upon the role of the user requesting the view. Controllers will customize the view (hide controls) for the purpose.

For the PoC implementation, the service layer will not have its own authentication; it will rely on the web layer for the same. Service layer components however will perform their own authorization.

HA security component will be used to perform authentication and authorization and to retrieve role permissions.

As EHR is an intranet application, secure channel (HTTPS) protocol is not required.

4.1.6.1.3 Data Caching

Application has very limited caching requirements which will be met with the caching features provided by Hibernate. Because all applications layers will be deployed in a single JVM, caching in DAO layer will be effective enough.

4.1.6.1.4 Rules Engine

This service is used to externalize the business rules from the application component for ease of maintenance. Externalizing the rules also makes it easy to reuse the rules within an application. The business rule framework is used by the business layer.
EHR will use the Rule Engine service provided by HA framework.

4.1.6.1.5 Configuration Management

The limited configuration management requirements of the application will be met through features provided by spring framework for the purpose. ApplicationContext interface of Spring allows reading configuration from properties files and resource bundles.

4.1.6.1.6 Exception Handling

This will provide exception handling mechanism to the application components. It will provide a common way of encapsulating the exception causing data and also a consistent way of propagating and handling the exception scenarios.
Details of exception handling framework are contained in Exception Framework document.

4.1.6.1.7 Bootstrap

This service is used to initialize the application during application server start-up. All data to be cached will also be loaded during the boot strap phase. Spring framework's eager initialization feature will be used to load all the data required by the application at the start-up.

4.2 Functional Architecture

4.2.1 Overview

Functional Architecture specifies software services (Core Business services and other support services) present in the proposed system.

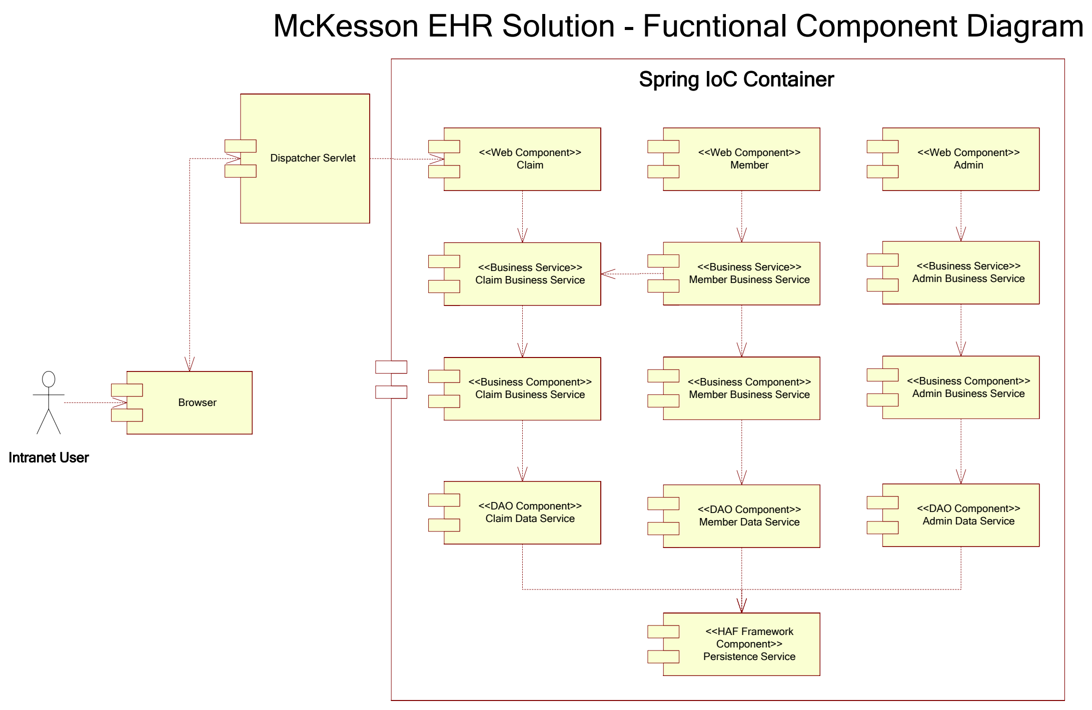
McKesson EHR Solution - Fucntional Component Diagram
Spring IoC Container
Dispatcher Servlet
<<Web Component>> Claim
<<Web Component>> Member
<<Web Component>> Admin
<<Business Service>> Claim Business Service
<<Business Service>> Member Business Service
<<Business Service>> Admin Business Service
Browser
Intranet User
<<Business Component>> Claim Business Service
<<Business Component>> Member Business Service
<<Business Component>> Admin Business Service
<<DAO Component>> Claim Data Service
<<DAO Component>> Member Data Service
<<DAO Component>> Admin Data Service
<<HAF Framework Component>> Persistence Service

Figure 2: Functional Component Diagram

4.2.2 Services

4.2.2.1 Presentation Services

Presentation services will interact with the end users through a graphical user interface hosted in a web browser. These services will be responsible for processing HTTP requests received and rendering HTML content to the user's browser.
Presentation services will be based upon Spring MVC framework.

4.2.2.2 Business Services

The business services will provide the core functionality required by the PoC application. The functionality will be provided as coarse grained interfaces. Coarse grained interfaces are easier to maintain and they provide better performance in a distributed environment. To improve reusability, the coarse grained interfaces will be implemented through collaboration of fine grained business components. This improves reusability because new composite behaviors can be easily created from the fine grained components. Following business services have been identified for the PoC application:

4.2.2.2.1 Admin Service

This service provides following operations which helps the administrator in maintaining the EHR application.

- User maintenance (Create, Delete, Update)
- Role maintenance (Create, Update)
- Assigning a role to a user
- Assigning permissions to roles

4.2.2.2.2 Member Service

This service provides following functionalities which are related to a member.

- Search members
- Provide EHR information for a member
- Provide claim information associated with a member
- Provide member compliance information

4.2.2.2.3 Claim Service

This service provides following functionality

- Provide claim information

4.2.2.3 Data Services

The EHR Data services will focus on providing domain specific data storage/retrieval services to business services. They will encapsulate the data access mechanism. Spring framework provided persistence mechanism using Hibernate as the underlying O/R framework will be used to provide data services.

5 Infrastructure Architecture

5.1 Infrastructure Architecture

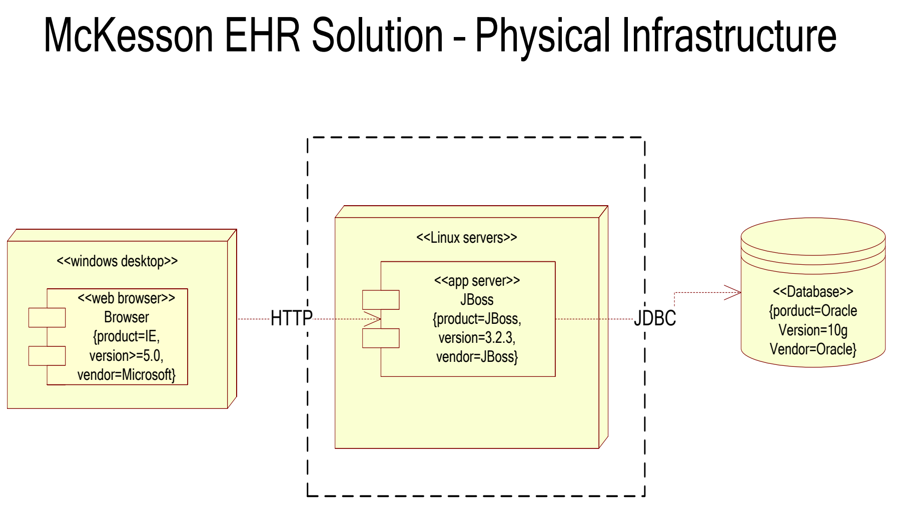
McKesson EHR Solution - Physical Infrastructure
<<windows desktop>>
<<web browser>> Browser {product=IE, version>=5.0, vendor=Microsoft}
HTTP
<<Linux servers>>
<<app server>> JBoss {product=JBoss, version=3.2.3, vendor=JBoss}
JDBC
<<Database>> {porduct=Oracle Version=10g Vendor=Oracle}

Figure 3: Deployment Diagram

The infrastructure proposed for the EHR PoC will consist of a single instance of JBOSS running on a Linux server. The JBOSS instance will provide both the Servlet container functionality and http web server function   nality.

5.2 Technology Overview

The technology stack chosen for the PoC application is described in the table below.

Application Category	**Name**	**Version/ SP/Patches**	**Comments**	**Reference Sites**
Application Server	JBOSS	3.2.3		www.jboss.org
Database	Oracle	10g		www.oracle.com
Java Virtual Machine	JRE (Run Time Environment) for Java 2	1.4.2		Java.sun.com
JDBC Driver	Oracle	10.1.0.4.0	ojdbc14.jar is the jdbc driver which will be used for database connectivity	www.oracle.com
O/R Mapping	Hibernate	3.0.5		www.hibernate.org
Application Framework	Spring	1.2.5		www.springframework.org
Tag Library	DisplayTag	1.0	Standard Pagination	displaytag.sourcefo

			Components	rge.net

Table 7: Technology Stack

In addition to the application software, the following tools are used.

Application Category	**Name**	**Comments**	**Reference Sites**
Design	Rational Rose 2000	Commercial	www.rational.com
Data Modeling	Microsoft Visio	Commercial	www.microsoft.com
Java Development	Eclipse	Open Source	www.Eclipse.org
Build	Ant	Open Source	ant.apache.org
Unit Testing	JUNIT	Open Source	www.junit.org
Version Control	CVS	Open Source	http://www.nongnu.org/cvs/

Table 8: Development Tools for PoC

6 Non-Functional Requirements

6.1 Security

Role based security will be implemented. Existing security component from HA framework will be utilized for the same.

6.2 Session Management

Session management will be done in the JBOSS application server.

6.3 Audit Trail

None

6.4 Performance

No page transition shall take longer than 10 seconds. This includes parsing the request, any queries, processing the result HTML page, as well as its display within the browser.

6.5 Scalability

None

6.6 Availability

None

6.7 Browser Compatibility

The application will support Internet Explorer 5.0 onwards.

6.8 Multilingual

None

6.9 Error Handling Requirements

The system shall support the logging of all system errors and the ability to view the error log.

6.10 Usability Requirements

None

7 Key Mechanisms and Patterns

7.1 Application Architecture Patterns

Singleton - The singleton pattern is used in framework services like caching, security implementation etc. The cache Manager and Security Manager implement the singleton pattern. All services will be configured to be Singleton in Spring IoC container.

Factory - Wherever object creational aspect needs to be addressed, factory pattern is being utilized. This will help avoiding proliferation of usage of "new" keyword across application component layers and will ensure centralization of object creation responsibility in a single method. Spring ProxyFactoryBean bean implementation has been used for the purpose.

MVC – The MVC pattern is used to decouple Model, view and Controller functionality. In the POC context, Spring WebMVC framework is used to achieve this objective.

Façade – The façade pattern is used in Business Layer to offer coarse grained functionality.

Front Controller – The Dispatch Servlet used in Spring webMVC framework implements Front Controller pattern.

7.2 Mechanisms

Inversion of Control

Inversion of Control or *Dependency Injection* is a principle used to reduce coupling between objects. In Inversion of Control, the control of code execution, creation of dependencies etc are taken away from components and given to the framework. The component declares all its dependencies to the framework. The framework is responsible for instantiating all the dependencies and wiring them together and making the component available for use. In this application spring framework provides IoC.

Aspect Oriented Programming

The aspect-oriented programming (AOP) methodology facilitates abstracting out crosscutting concerns like logging, auditing, and security into separate modules and then applying these modules dynamically where they are needed without knowledge of the objects on which they are being applied. Spring provides its own implementation of AOP which is purely JAVA based and leverages its IoC container. In this application AOP will be leveraged for providing logging and transaction management. Spring provides out-of-box support for latter through its TransactionProxyFactoryBean class.

8 Glossary

Sr.#	Abbreviation	Description
1	DAO	Data Access Object
2	DO	Data Object
3	EJB	Enterprise Java Bean
4	J2EE	Java 2 Enterprise Edition
5	JDBC	Java Data base connectivity
6	JNDI	Java Naming and Directory Interface
7	JSP	Java Server Pages
9	PoC	Proof Of Concept
10	EHR	Employee Health Record
12	UI	User Interface
14	AOP	Aspect Oriented Programming
15	IOC	Inversion Of Control
16	MVC	Model-View-Controller Pattern

Table 9: Glossary

9 Issues

Sr#	Description	Action by	Target Complet ion Date	Priority	Resolution
NONE					

Table 10: Issues

APPENDIX

This Section documents the HAF framework provided features which are going to be used in the *Proof of Concept* (Henceforth called POC) application.

Sl #	HAF Developer Guide Reference	HAF Feature	Used	Remarks
1	Core_Developer_Guide_9.8.0.doc	Configuration	No	The application has very limited configuration management requirements which can be met through Spring framework features and do not require a separate configuration management component
2	Core_Developer_Guide_9.8.0.doc	Logging	Yes	
3	Core_Developer_Guide_9.8.0.doc	JMX	No	The 'Proof of Concept' application does not have any requirement to incorporate JMX facility.
4	Business_Services_Developer_Guide_9.8.0.doc	Event based handling facility	No	Event based handling facility in business layer is not going to be used since it's specifically meant for .Net based client.
5	Business_Services_Developer_Guide_9.8.0.doc	DAO	Yes	
6	Business_Services_Developer_Guide_9.8.0.doc	Spring Inversion of Control	Yes	
7	Business_Services_Developer_Guide_9.8.0.doc	Spring persistence capability using Hibernate	Yes	
8	Business_Services_Developer_Guide_9.8.0.doc	Spring declarative transaction capability	Yes	
9	Business_Services_Developer_Guide_9.8.0.doc	Mapper Factory	No	Mapper Factory is not going to be used because it's tightly coupled with the event handling mechanism which is especially targeted for .Net based thick client development.

Sl #	HAF Developer Guide Reference	HAF Feature	Used	Remarks
10	Business_Services_Developer_Guide_9.8.0.doc	Advanced features of View class tiers	No	These features are not going to be used in the POC because these facilities are targeted for Thick client development.
11	Business_Services_Developer_Guide_9.8.0.doc	Beanchanged event framework	No	This facility is specifically targeted for thick client development and hence is not going to be used in the POC.
12	Rules_Language_Developer_Guide_9.8.0.doc	Rule Engine facility	Yes	
13	Session_Developer_Guide_9.8.0.doc	Session management facility	No	Advanced concepts offered by this framework like Event registry/listener are not going to be used since the POC does not require these facilities. Hence the POC will only use the basic HttpSession facility provided by Servlet Container.

Table 11: HA Framework Usage