Architecture Quality Attributes for Knowledge Management System

Shekar Sivasubramanian

CMU-LTI-16-004

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Dr. Eric Nyberg (Chair)
Dr. Jamie Callan
Dr. Robert Frederking
Kiran Hosakote

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Table of Content

List of figures

1 Introduction

1.1 Background

The 1990s have seen the emergence of "global software development centers" or simply Global Development Centers (GDCs) located in different parts of the world to serve the software development needs of customers. A GDC forms a viable large-scale, economic model for the remote development of software driven by cost benefits offered by the workforce in these locations. Typically, there is a concentration at the center(s) in India, with personnel located at the customer site for domain analysis and to support project implementation. There is a natural repetitiveness in the work due to the finite number of technologies used in a GDC environment. The nature of work that can be requested by multi-location, multi-business, large customers have significant opportunities to use already created technology solutions. Currently, GDCs perform each project with little or no knowledge of any similar, past work that may have been executed by them, for a customer.

Software reuse has been recognized as an approach to improve the productivity and quality of delivered software solutions, and has been applied with varying success (Basili et al: 1994). Technical and organizational impediments need to be fully addressed to harness the complete benefits of a software reuse program. Several factors in the GDC environment lend a favorable environment to overcoming these impediments. There is a need to implement a disciplined knowledge management framework to ensure that a software reuse program can be successful. GDCs are already familiar with the adoption of frameworks, since many of them use the Capability Maturity Model to optimize their software development processes. The framework can be adapted to create a similar model for knowledge management. Significant themes that will form the basis of this framework include people practices, including the creation of communities of practice, knowledge processes adapted to the project-oriented business and technologies that address the unique nature of a GDC environment. The GDC environment has become intensely competitive, and there is a need to seek a sustainable advantage in software development. Coupled with the natural economic benefit, the integrated knowledge management framework may well provide the sustained positive productivity inflexion and core competence that software organizations will need in the future.

The purpose of this document is to present the various requirements and constraints that need to considered while developing architecture for a knowledge management system. This document will not describe the implementation details if they do not affect the architecture. The goal of the document is to reflect on the architecture of the system using the twin lens of the requirements and quality attributes. This document will be used to define specific quality attribute scenarios to define architectural requirements, and elaborate on the more crucial ones that affect the architecture of the knowledge management system.

1.2 Description

A Knowledge Management System (KMS) is an enterprise-wide system that forms the foundation for the capture, storage, retrieval and usage of knowledge within an organization. The KMS described is in the context of a software development organization. This system exists

with other systems that are responsible for process management in a software organization. The following provides a high-level overview of the sub-systems of such a system:

1. Knowledge Creation. This area is responsible for the identification and capture of knowledge in an organization.
2. Knowledge Classification and Storage. This area is responsible for the classification and storage of knowledge that has been created or enriched in an organization.
3. Knowledge Enrichment. This area will perform the enrichment of knowledge in an organization.
4. Knowledge Dissemination and Sharing. This will perform the dissemination and sharing functions associated with knowledge. Once this is done, the knowledge is put to use.
5. Knowledge Communities and Expertise. This will contain the expertise in the organization and manage the communities of practice.
6. Knowledge Metrics. This sub-system provides information on the metrics associated with the Knowledge Management System.

1.3 Context

The Knowledge Management System is a critical resource for a software development organization. This ensures that knowledge gained during the course of software development projects can be stored, shared, enriched, and used across the organization use this system. Such a system must have the ability to store a large number of artifacts and permit iterative and collaborative enrichment of the artifact. In addition, the system must be able to store the assessment and classification of artifacts based on an expert's assessment of the artifact. Information related to expertise will be stored in the system and will constantly reflect the organizational repute of the expert in the organization. Communities of Practice will be able to access different areas of interest to ensure that knowledge-related activities take place in their areas of interest.

1.4 Acronyms

Abbreviation	**Description**
C&C	Component and Connector
DB	Database
EJB	Enterprise Java Beans
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
J2EE	Java 2 Enterprise Edition
JVM	Java Virtual Machine
MTTF	Mean Time to Fail
MTTR	Mean Time to Repair
MVC	Model View Controller
OA&M	Operations, Administration and Maintenance
RDBMS	Relational Database Management System
SQL	Structured Query Language
URL	Universal Resource Locator

1.5 Intended Audience

The report is intended for the following audience:

- Stakeholders (direct or indirect) of the project.
- Audience that might want to get an insight into or analyze the architecture and design of the system.
- Audience that have an interest in documentation of the architectures and design of software systems.

2 Requirements

2.1 Overview

The following sections provide the high-level requirements for the Knowledge Management System for use in the context of a software development organization. First, the functional requirements of the system are defined. Then, the architectural requirements are provided in the form of quality attributes using a set of general scenarios.

2.2 Functional Requirements

The following provides the functional requirements of the KMS. This KMS supports the needs of a software organization. The system must have the capability to co-exist within the context of a process-centric software organization. It is assumed that the system is available in today's context – over the web for use by users in the corporation. There are three types of users as stated below:

1. Users from the software development organization. Such users are the principal users of the KMS. A special type of user is an acknowledged knowledge expert.
2. Knowledge administrators who maintain the classification scheme for the KMS and perform other administrative functions.
3. Knowledge experts who are responsible for specific functions associated with knowledge classification.

The transactions for the system are as follows:

1. Create a knowledge artifact.
    - Create a new software artifact.
    - Maintain key linkages to the process used in the software organization.
2. Establish and manage a classification scheme for knowledge artifacts [by a knowledge administrator, in conjunction with knowledge experts].
    - Create the classification scheme with the help of knowledge experts.
    - Review and maintain the classification scheme.
    - Provide information on a knowledge artifact associated with its usefulness. The knowledge expert performs this.
3. Classify a knowledge artifact.
4. Establish a knowledge need based on the requirements in a software project.
5. Track fulfillment of the knowledge need.
6. Enrich a knowledge artifact. This involves making changes to the artifact to facilitate more appropriate usage in the organization.
7. Establish communities of practice and information about of knowledge experts [by a knowledge administrator].
8. Upload information on people, including repute as it relates to skill areas.
9. Get a list of knowledge artifacts based on a query. Queries can be structured, or text based searches of the classification scheme.
10. Get a list of knowledge artifacts that are currently being used by the organization.
11. Get a list of people who are most relevant to fulfill a knowledge need.
12. Provide a list of knowledge artifacts that have been created by a project.
13. Provide a list of knowledge artifacts that have been consumed in a project.

14. Provide a complete list of enrichments that have taken place to a set of artifacts.

2.3 Architectural Requirements

The following architectural requirements have been identified.

1. The system should respond to user requests in a reasonable amount of time based on the nature of the requests.
2. It is necessary that the system be highly available.
3. The system will be used in the context of a global software organization. The system must therefore be deployed in an environment that can be distributed across multiple geographies.
4. The system should enable a quick learning curve for new users so they are able to quickly understand the base functionality in a short time.
5. It is expected that new functionality will be requested from the user community. Hence, the system will be continuously developed and extended in the next few years.
6. Interaction with other systems is a likely future extension. In particular, the system must be able to coexist with automation related to software development.

2.4 Quality Attributes

Quality attributes are used to define the architectural requirements of the application. Quality attribute scenarios have been used to capture the quality attributes. The quality attributes have been defined in a tabular format to capture the stimulus and response information. This has been done for the common and important quality attributes for our system.

2.4.1 Performance

Performance refers to two principal characteristics. Since the KMS is not a transaction system and is used iteratively for support knowledge needs, this system has modest performance characteristics:

- The responsiveness of the system in an interactive mode is measured by response time – the time required to respond to a user request. The system will provide response times of less than 2 seconds for routine operations on the database.
- User requests lists of information in line with searches provided by the system. Such requests will complete in 5 seconds.
- The system will also provide a request and response capability for complex operations. The system must be able to capture requests and perform the action and report status to the users based on action performed. Classification of artifacts may take time, since the algorithm for classification may use multiple dimensions for classification, depending on the complexity of the classification scheme. This will include complex searches on the database that may require extensive processing. Such requests will be completed in less than 5 minutes for 90% of the requests in an asynchronous manner.
- The throughput of the system is measured by the number of concurrent transactions that can be processed (the maximum load on the system, or the capacity of the system) is measured by the number of transactions per second for a prescribed number of concurrent users. Concurrent users is defined as the number of users who submit a transaction for processing by the system at precisely the same time – there is often a correlation between the number of concurrent users and the total number of users using the system. Often, this is a heuristic that suggests that a specific percentage of logged on users for this type

of system are concurrent. For the Knowledge Management System with a large number of users, this percentage is expected to be low, and is pegged at 10%.

The following table provides greater information on the performance quality attribute measure using scenario-based depiction.

S No	Stimulus, Source, and Environment	Response, Response Measure, Artifact
1	User performs routine operations that update information in the repository. This includes storing an artifact, storing information on expertise, providing review feedback on an artifact, retrieving simple and storing classification information.	Response time for such transactions by the processing modules must be 2 seconds or less.
2	User initiates request for complex information.	Processing must be completed in 5 minutes or less for 90% of the cases. Classification completion must be provided to the user.
3	User tries transactions that are more search-centric, and may produce lists, under normal operating conditions.	Response time for such transactions by the processing modules must be 5 seconds or less.
4	At normal operating conditions 10 concurrent users use the system, with response times within acceptable levels.	Throughput of the number of transactions (estimated at 50) that support this level of usage by the system must be supported with acceptable response times, as defined above.

2.4.2 Availability

Availability refers to system failures and faults in the system. The users of the application measure a system failure in terms of the system not being available for use. The requirements below try and define the commitment associated with the uptime for the system, and a measure of recoverability from a specific type of error.

The following table provides greater information on the availability quality attribute measure using scenario-based depiction:

S No	Stimulus, Source, and Environment	Response, Response Measure, Artifact
1	There is a fault in the business processing, when the system is operating normally.	The system has the capacity to maintain availability of the system for the users, with no loss of transactions. The system must be available for usage for 99% or more of the time. Availability is measured as the ratio of MTTF to a sum of MTTF and MTTR. This does not include scheduled downtime as part of the calculation.
2	Users log on at 11:30 PM on Saturday, when the system is operating normally.	The system notifies logged on users of the scheduled downtime 30 minutes before the scheduled downtime. Messages are again sent at 11:44 PM and 11:54 PM. The users

		are force-logged off at 11:59 PM. The system is available for use at 4:00 AM on Sunday.

2.4.3 Usability

Usability is concerned with the ease with which a user can complete tasks. In KMS, the importance of usability cannot be undermined. The system must be able to ensure that new users can intuitively use the system with minimal external inputs. In today's world, there are certain standard processing abilities that are assumed in the manner in which systems interact. Therefore, the application must ensure capabilities that are regarded as generally accepted norms in such an environment.

The following table provides greater information on the usability quality attribute measure using scenario-based depiction:

S No	Stimulus, Source, and Environment	Response, Response Measure, Artifact
1	A user logs to the system and wishes to use the system, at normal operating conditions.	The user has the ability to perform common operations in one hour, with structured assistance from the learning module of the system.
2	A user faces a problem in usage of the system, and has a problem to resolve.	The system provides assistance to the user, so that 80% of the problems faced by the user community are arrested by the system. Only 20% of the queries require the user to call the help desk. In such cases, the system routes processing to its help module which will take control of providing assistance to the user.

2.4.4 Modifiability

Modifiability provides information on the cost of change. It is widely recognized that significant costs are expended in software, after the deployment of the initial solution. Causes of this could vary from corrective maintenance (correcting defects), performance related maintenance (as the scale increases, the need to modify the system), and preventive maintenance (adding on incremental functionality to improve the product). In addition, based on the needs, there are always points of inflexion in systems, when significant changes may be needed to address the reason for the inflexion (significant change in technology, completely different standards to adhere to, and so on). In a KMS, modifiability is an important attribute because the system will constantly undergo modification, as an organization understands various aspects of knowledge. The system must be capable of accommodating such changes with rapid turnaround times for releases.

Modifiability goes to the heart of cost containment in business applications, when the original set of staff that developed the application may no longer be at hand, several years after the system has been deployed. Therefore, this has to be addressed in the architecture through separation of concerns, and limited side effects in implementing change.

The following table provides greater information on the modifiability quality attribute measure using scenario-based depiction:

S No	Stimulus, Source, and Environment	Response, Response Measure, Artifact
1	The knowledge administrator needs to add classification to an artifact or delete classification from an artifact.	The system must provide the ability to provide flexibility in storing classification schemes, with no developer intervention.
2	The knowledge experts and communities of practice must be loosely integrated into the rest of the KMS system – significant modifications are expected in this area.	Changes in the structure of communities of practice must have minimal development impact on the rest of KMS.
3	The nature of the KMS is evolutionary – which means that the system will be constantly changed based on greater understanding of patterns in an organization associated with all aspects of knowledge.	The system must be capable of incremental improvements in functionality, without having a complete revamp of the system.
4	Most organizations will already have defined their own nomenclature associated with process management in software development. The KMS must be able to integrate with other systems.	Ability to manage interchange of critical information with already existing process management software in software organization with minimal to no code change in KMS.
5	Changes have been made in response to changing business conditions, technology, regulations, or interfaces to new systems by the development community, and have been tested.	The system must ensure that such releases can be completed in the scheduled downtime planned for the application (see Availability).
6	Critical errors must be attended to, in a timely manner.	Critical defects must be attended to in a span of less than 8 hours.

2.4.5 Security

Security is concerned with the system's ability to resist unauthorized access while providing access to authorized users. The system must be able to provide controls to ensure that authenticated users alone are permitted to use the system. In addition, there are two classes of users in the system – staff users and normal users. There are different levels of authorization for the two types of users. In addition, there may be a need to ensure that accountability for transactions can be verified, if there is a need.

The following table provides greater information on the security quality attribute measure using scenario-based depiction:

S No	Stimulus, Source, and Environment	Response, Response Measure, Artifact
1	Unauthorized user tries to access the system in normal operating environment.	The security modules in the system must ensure that unauthenticated users are not permitted access to the application. The system must maintain passwords and related user sensitive information in an appropriate

		manner to prevent tampering and access.
2	Authorized normal user accesses the application and tries to perform staff functions, in normal environment.	The security modules in the system must prevent normal users to perform knowledge administrator functions.

2.4.6 Testability

Testability measures the ability for software to demonstrate its faults. Since a significant cost is expended in testing, the system must ensure that the testability of the built system is robust.

The following table provides greater information on the testability quality attribute measure using scenario-based depiction:

S No	Stimulus, Source, and Environment	Response, Response Measure, Artifact
1	Developers have completed the development, unit testing, and integration testing of the system. They are ready to perform final business testing of the system, in preparation for a scheduled release, and would like to conduct the regression test.	The regression test must contain a test harness of over 50 test cases, and must execute in the system in under 10 hours.

2.5 Constraints

The following list provides technical constraints:

1. The system must operate on all standard browsers.
2. The system may involve users to have the need to download any software to run it. In such a case, the KMS system must manage the distribution of software.

3 Utility Tree

3.1 Overview

Quality attribute scenarios are used to specify the quality requirements. Quality attributes for the system have been defined in the prior section. The graphical view of the utility tree follows the definition presented in (Bass et al: 2003), but with the root utility node. The columns in the tree are the described below

- **Quality Attribute**: As described in the earlier section.
- **Attribute Concern**: Provides a brief overview of the concern area.
- **Scenario**: Provides a brief description of the scenario associated with the attribute.
- **Rank:** Given in a pair format, where the first member gives the importance and the second is an estimated implementation difficulty. Both values are given in discrete levels of High, Medium and Low.

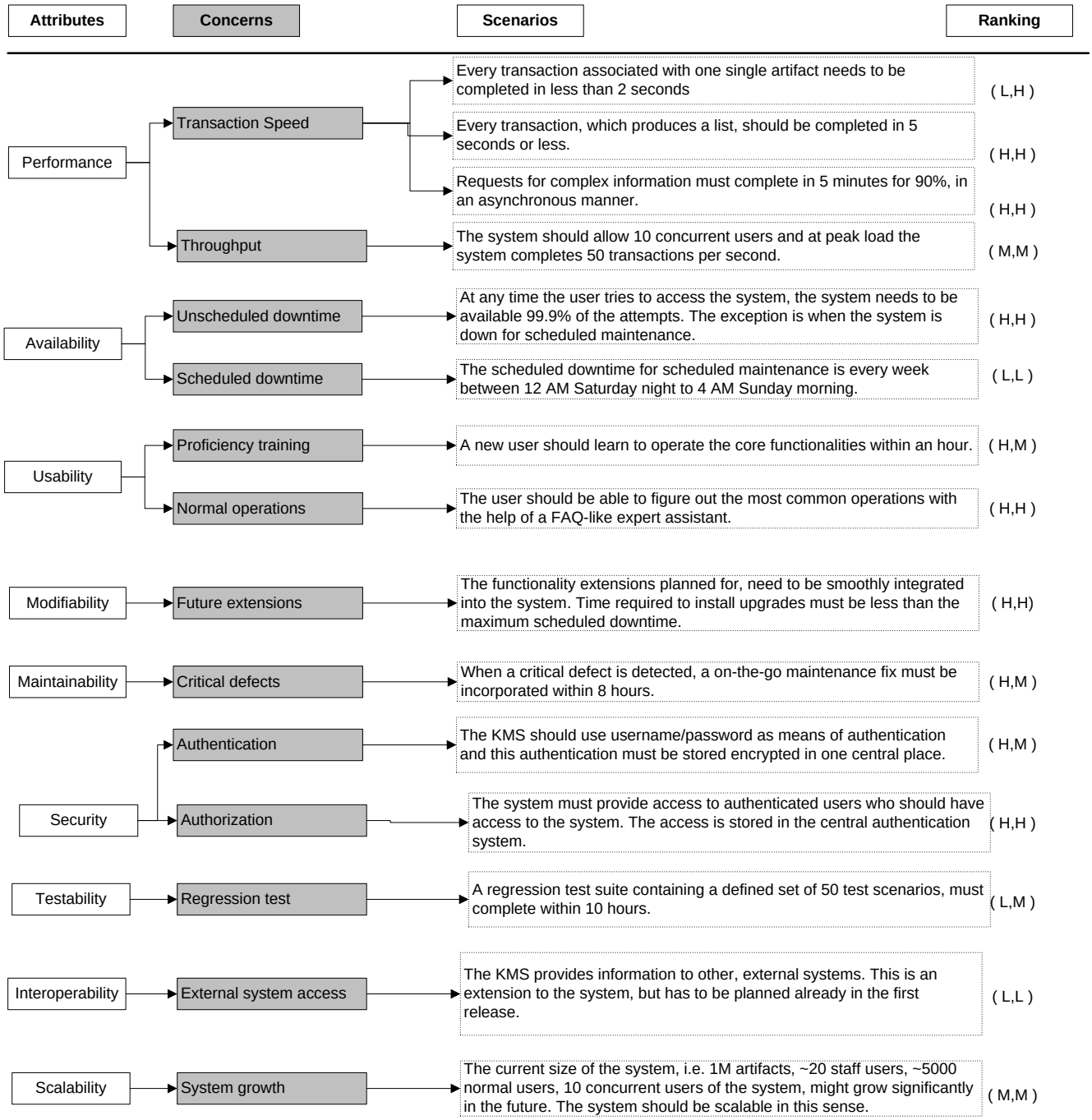

Figure 1 Utility tree for KMS

4 System Architecture

4.1 Introduction

The system can be characterized as an application that requires the user to use a standard front-end interface for interaction with it. With the need for web-based interaction, the natural fit from an architectural style is the call-return architecture using a J2EE framework. The next sections provide an overview of each tier.

4.2 Client Tier

The client tier forms the principal form of interaction for all users in the system. This contains the following.

4.2.1 Browser based interaction to the system

The system provides a standard URL, for interaction to the system. This presents a screen, which contains a user-id, and password screen used to authenticate users who wish to use the system. Once the user passes the authentication, the system provides a menu for the different functional aspects of the system. The browser-based client forms a web client and will be thin client.

4.2.2 Interaction to other systems

An interfacing system that makes a request to this system, is considered as another type of user, with different data format for information that is delivered – in general, such requests from other systems will be synchronous, and uses a set of well-defined interfaces provided by this system. Information will be delivered in a standard XML, with a well-defined DTD that will be used by the other system to interpret the data.

4.3 Business Tier

The business tier provides the functional processing of the knowledge management system. It has the following components.

4.3.1 Web Interaction Component

The web tier provides the management of the front end, in terms of provisioning the user input for the system. JSP will be used to manage this. The web tier will execute in the web container, and will also be responsible to manage the state associated with the user. This is an architectural choice that is related to the performance needs and the nature of the system. The sequence is as follows:

- User logs on, and is authenticated with the security component in the business tier.
- Authorized users have a session managed in the web tier, which provides the ability to retrieve session information, indexed on the user id. Since duplicate user-ids will not be permitted in the system, this assures unique keys to the sessions that will be managed by the web tier.
- This allows for optimal performance, since it permits the use of stateless session beans in implementing the business tier.
- The nature of the application ensures that information stored in a session is kept to a minimal amount.

- Inactive sessions will be routinely managed through an appropriate configuration at the web tier (set for the application to 15 minutes).

4.3.2 Security Component

This module forms the first module that is invoked by the user. The principal function of this module is to perform the following checks:

- Only users with an authorized user-id and password can access the system. After three authentication errors (incorrect user id or password), the account is locked out, and the user must call the help-desk to get the account reactivated.
- Ensure that subsequent business functions are provided for the two types of users – complete functionality is provided for knowledge administrators, while restricted functionality is provided for normal users.

A single security component will be used to perform all authentication and authorization functions, irrespective of the nature of the client (web based client, phone user, or another system). The choice to isolate this has been done in this manner, since this is a one-time need for any user, and it permits easier changes when any of the security roles or rules changes (improved modifiability).

4.3.3 Business Processing Component

The business session components manage the interaction for the interactive user – this will be implemented using enterprise session beans. Enterprise beans are used to meet the following requirements:

- The application must be scalable. To accommodate a growing number of users, we may need to distribute an application's components across multiple machines, with location transparency to the clients.
- Transactions must ensure data integrity. Enterprise beans support transactions, the mechanisms that manage the concurrent access of shared objects.

The business session will be managed using session beans, which connect up to the entity beans that implement the access to the database. Session beans will be stateless, since the nature of most transactions can be managed using stateless beans and performance is paramount in the system. A centralized servlet will invoke a specific stateless session bean, based on the specific choice made by the user. This session bean will invoke appropriate functions in the data access layer and fulfill the request.

4.3.4 Data Access Component

Data Access Components will manage the information and interaction with the database. The session beans to perform the needed processing with the database will invoke these. These java classes will use JDBC as the form of connection to the database. They will interact with the database and retrieve and update information as required by the business tier. Specifically, this architecture does not use entity beans to manage the interaction, once more to provide the needed performance and reduce inter-bean communication.

4.3.5 Interface Components for other systems

Interface components for other systems manage the interface to other knowledge management systems. Other knowledge management systems are mimicked as users, where the web browser is not the client. The client is another system that consumes already agreed-upon data-

interchange formats in XML. Data that is exchanged can be done using HTTPS protocol to ensure appropriate security.

4.4 Data Tier

The data tier is implemented using a set of stored procedures that are compliant to ANSI standard SQL to deliver information to the business tier. The database itself will be a set of tables that map on to the data storage and retrieval requirements of the system. Indexes will be defined on appropriate requests to ensure that the response times can be met. Specifically, indexes will be defined to optimize the access times for single transactions to ensure that the response time of 2 seconds can be bet.

4.5 Component and Connector View

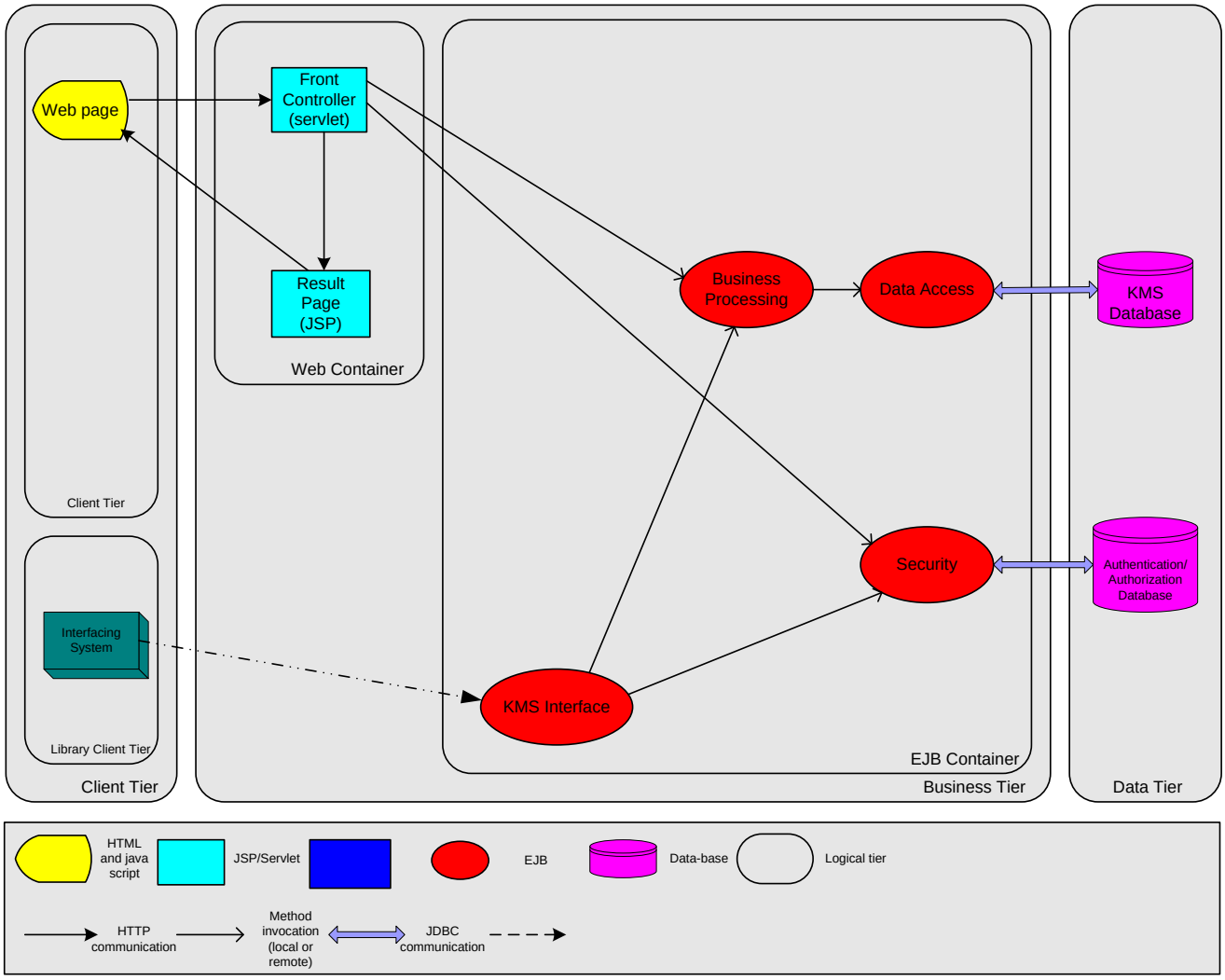

Figure 2 Component and connector view

Observe the following in Figure 2.

- The different tiers in the figure will definitely run on different machines, i.e. client, business and data tier, as shown also in Figure 5.

- More specifically, within the business tier, there can be an arbitrary number of servers implementing the tier, depending on application demand. Profiling information together with monetary budget and performance requirements will determine how the deployment will evolve.

4.6 Module View, Data Descriptor

The following provides a brief entity-attribute view of the data elements of significance in the system. This can be used to reason about the performance of the system, in terms of the database structures used to implement the functions, as well as for determining the processing elements to be designed in the data access and business processing layers.

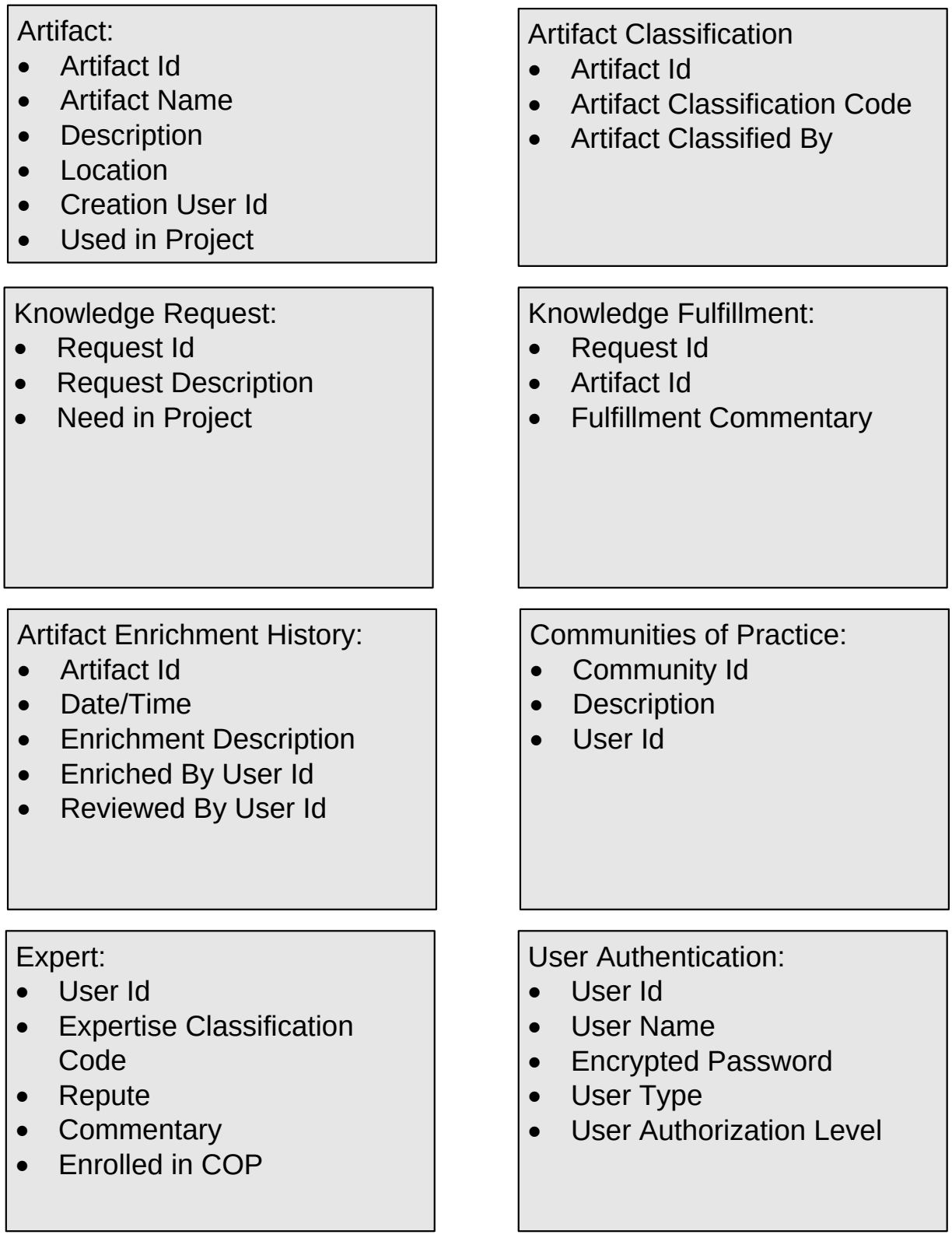

Figure 3 Considered artifacts in the system

Communities Enrollment:
- COP Id
- Classification Code
- Description
- Nature of Enrollment

Ontology Master:
- Classification Code
- Description
- Classified By

Project Master:
- Project Id
- Project Name
- Project Manager User Id

Repute Master:
- Repute Code
- Description

Information:
- Ontology Master has been simplified. This will be extended in implementation.
- Project Master is a representation of an interface to external systems.
- Representation of entities has left out implementation-centric details.
- This model will be constantly extended to include additional functions.

4.7 Module View, decomposition style

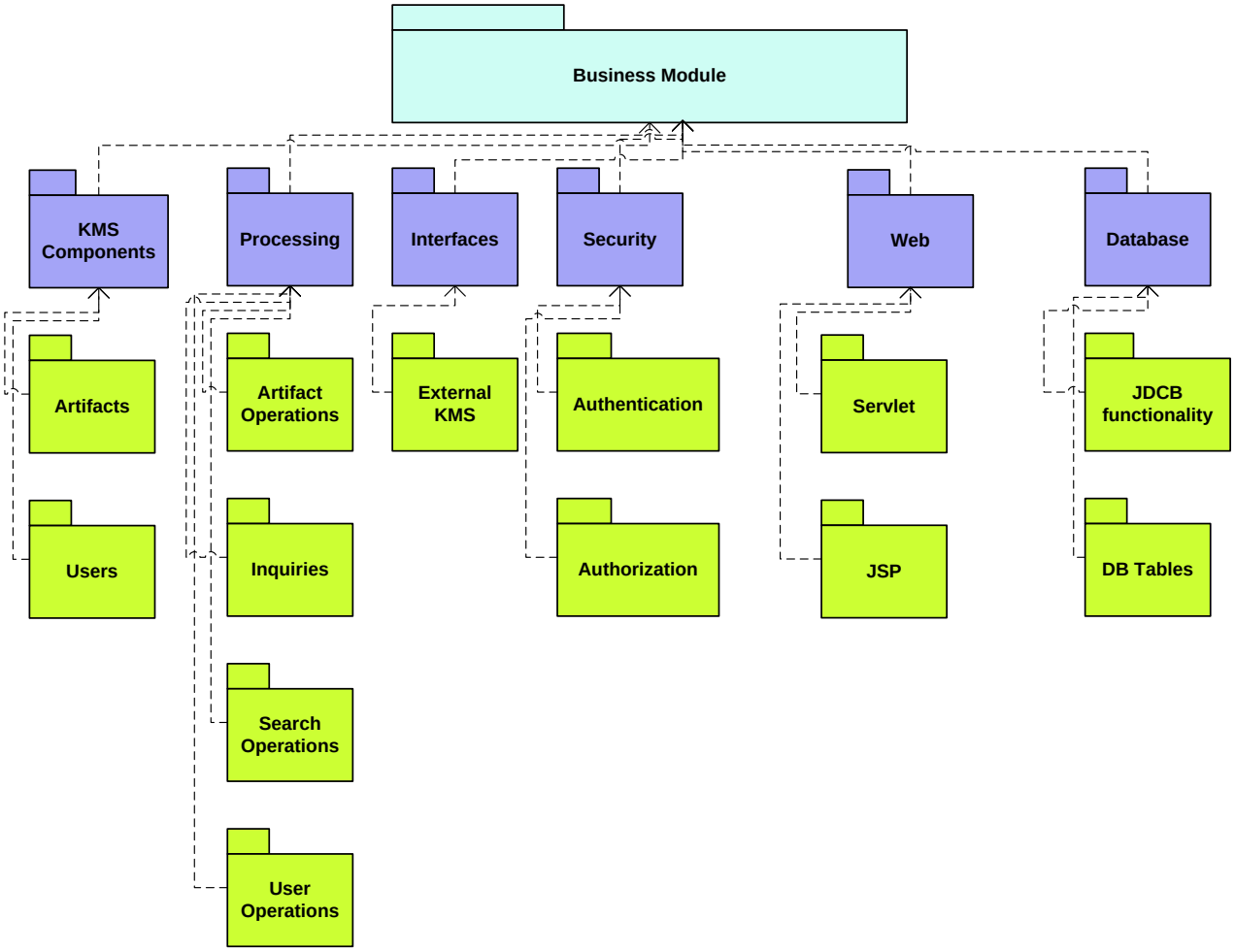

Figure 4 Module Decomposition

As for all pure decomposition style diagrams, the goal is to present the functionality of the system in intellectually manageable pieces.

4.8 Allocation View, deployment style

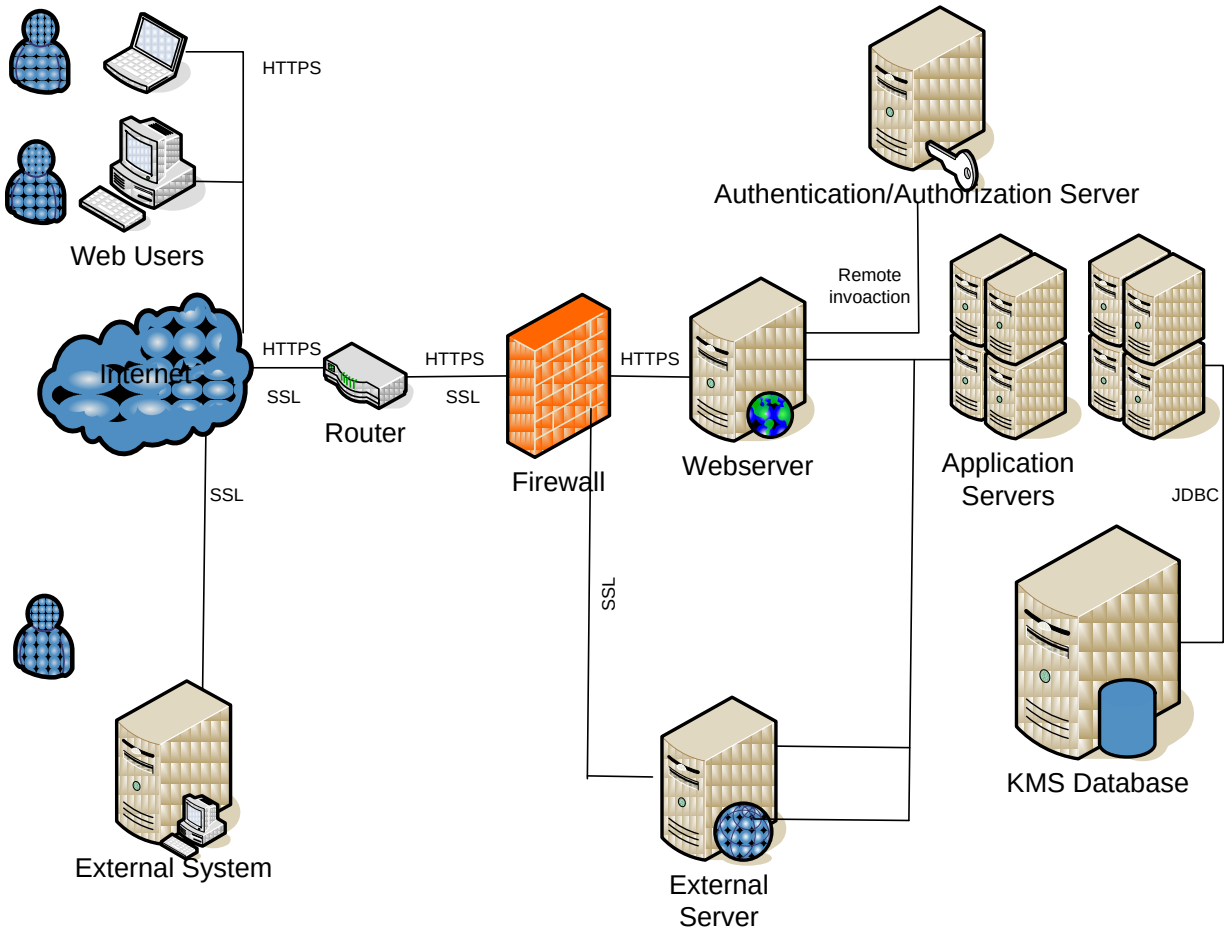

Figure 5 Allocation view, deployment style

The business tier is very scalable in terms of the number of servers used. The view above essentially shows the most generic implementation, when there is a farm of application servers, on which the EJBs run. The security component is also running on a separate server, since one would probably keep this machine in a safe location in order to avoid physical hacker attacks. Most likely the authorization/authentication database runs on the same machine as the security bean.

4.9 Sequence chart

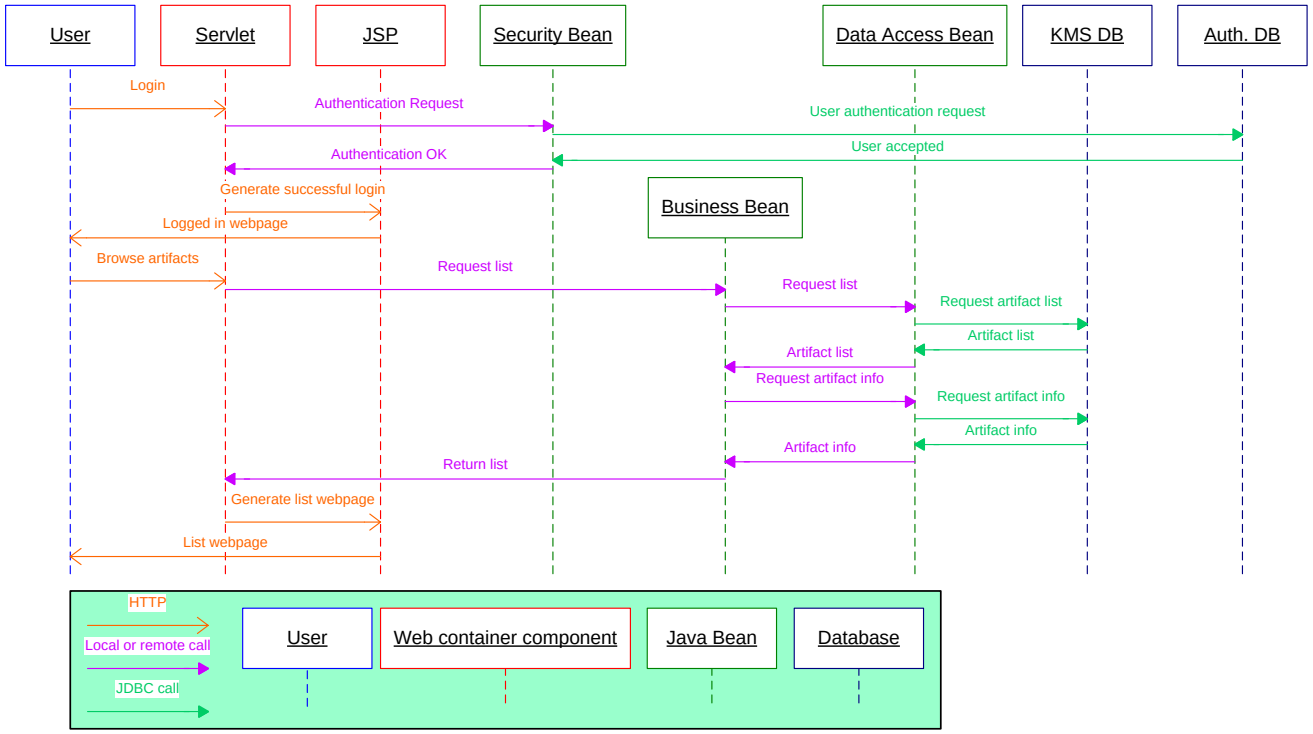

Figure 6 Sequence chart

4.10 Summary

The basic system is premised on call-return architecture – this establishes standard 3-tier architecture for the system. The three-tier architecture (also referred to as the multi-tier architecture) emerged to overcome the limitations of the two-tier architecture. In the three-tier architecture, a middle tier was added between the user system interface client environment and the database management server environment. The middle tier performs queuing, application execution, and database staging. For example, if the middle tier provides queuing, the client can deliver its request to the middle layer and disengage because the middle tier will access the data and return the answer to the client. The three-tier client/server architecture has been shown to improve performance for groups with a large number of users (in the thousands) and improves flexibility when compared to the two-tier approach.

The three-tier architecture with application server (J2EE architecture) has been chosen as the architecture for the knowledge management system. For adopting the technology and how long it takes to implement the technology, programmers can learn easily and quickly the Java standard library, which contains objects and methods for opening sockets, implementing the HTTP protocol, creating threads, writing to the display, and building a user interface.

5 Scenario Analysis

5.1 Introduction

This section provides the list of scenarios that are relevant to the system. This is presented in the form of a table which includes the quality attribute, the stimulus / environment, and the response / response measure. Where relevant, the artifact that is responsible to provide the response is identified. The analysis uses the terminology and framework of ATAM. Key concepts and interpretations are (Kazman et al: 2000) :

- *Risks* are architecturally important decisions that have not been made, or decisions that have been made but whose consequences are not fully understood.
- *Sensitivity points* are parameters in the architecture to which some measurable quality attribute response is highly correlated.
- *A tradeoff point* is found in the architecture when a parameter of an architectural construct is host to greater than one sensitivity point where the measurable quality attributes are affected differently by changing that parameter.

Risks, sensitivity points, and tradeoff points are areas of potential future concern with the architecture.

5.2 Transaction Speed

Stimulus:	User makes a single artifact related transaction or asks for lists.
Source of Stimulus:	User makes menu selection.
Environment:	Normal operating conditions of the system.
Artifact:	Appropriate processing components, including the web tier servlet, the stateless session bean, and appropriate data access components.
Responses:	User must get a response in 2 seconds or less in interactive response for single-artifact transactions, and 5 seconds or less for list transactions.
Response Measure:	2 seconds or less for single-artifact, and 5 seconds or less for list.

Architectural Alternatives	Risk	Sensitivity	Trade Off
Use of web-tier to manage the state, and stateless session beans to deliver business functions. Use of specific indexes in specific tables to support access times.	1, 2	1, 2, 3, 4	1
Use of stateful session beans. Use of specific indexes in specific tables to support access times.	1, 2, 3	2, 3	2, 3

5.2.1 Architectural Alternatives

1. Use of web-tier to manage the state, and stateless session beans. Use of specific indexes in specific tables to support access times. Web tier will be used to maintain the conversation state for the user.

2. Use of stateful session beans. Use of specific indexes in specific tables to support access times. Session beans will maintain the state of conversation for the user.

5.2.2 Risks

1. The need for JVM may increase the execution time for the transactions. Improvement in compiler associated with JVM is the mitigation for this risk.
2. Dependency on Internet connection speeds for external users has an impact on the response time. There is limited control over this factor. Internal users will not be affected since they will access the system in a LAN environment.
3. Memory management for stateful beans may need good understanding of underlying implementation of the J2EE architecture. Any changes in the architecture may have an impact on the implementation.

5.2.3 Sensitivity Points

1. The configuration and setup of the application server is important – in terms of managing the timeouts associated with when to release the bean pool back for new users.
2. The sizing of the application server is important to get an understanding of the number of concurrent users who can access the system and the memory footprint for each of the users.
3. Index performance and optimization is very important to provide access. Clustering of the tables must be done, periodically to the most-used index to provide optimal performance.
4. An understanding of session management in the web tier is important – specifically, the configuration associated the release of resources with a timeout is important.

5.2.4 Tradeoffs

1. Resource and processing of specific transactions will require distribution between the database and the application server.
2. Higher resources may be needed for stateful session beans at the application server side.
3. Availability may be more difficult to implement with stateful session beans.
4. May have an OA&M impact in managing the database periodically to manage the clusters in the tables.

5.2.5 Reasoning and Conclusion

The advantage of the web tier managing the session is that it permits minimal resources at the application server to do the processing, which has a performance impact in the completion of transactions. The application server tier is left, along with the data tier to fulfill the transactions – and therefore this offers the best opportunity for managing resources to deliver the performance in the J2EE architecture.

5.3 Security

Stimulus:	Unauthenticated user tries to access the system
Source of Stimulus:	User comes in from the web, the phone system, or another system
Environment:	Normal operating conditions of the system
Artifact:	Unauthenticated access is denied and appropriate levels of authorization are granted based on the user.
Responses:	Permission for inaccurate user id and password is denied with appropriate message for the front end, appropriate IVR response for the phone user, and appropriate XML return type for an interfacing system.
Response Measure:	No unauthenticated users can access the system.

Architectural Alternatives	Risk	Sensitivity	Trade Off
Centralized security module	1, 2	1	1
Decentralized security module	1, 2, 3	1, 2	1, 2

5.3.1 Architectural Alternatives

1. Centralized security module for all authentication and authorization.
2. Decentralized security module for different forms of authentication and authorization.

5.3.2 Risks

1. User sources are different and have different characteristics – this may have an impact on the security of the data being transferred between the user source and the system.
2. Changes in data interchange interfaces may have a side-effect in the security module.
3. If there are manual synchronization procedures in user management, there may be times when the security need may be compromised.

5.3.3 Sensitivity Points

1. System and user management is important to manage the currency of active users, along with attributes such as password, telephone-password, and overall authorization levels for the system.
2. Synchronization of user management is important and requires additional effort in user management. In a decentralized system, this must either be designed into the

5.3.4 Tradeoffs

1. Security transactions may take time to process, and have an impact on a performance. In a centralize system, there may be a higher impact on performance
2. Decentralized user management may have higher OA&M involvement in user synchronization.

5.3.5 Reasoning and Conclusion

The advantage of a centralized user authentication system is in consistency of implementation, and reduced issues associated with compromising the access and authentication for the system. This comes at a cost of reduced modifiability (since a decentralized module may be easier to

manage in terms of the code) and performance (since a smaller code set will probably execute for each of the types of accesses. However, in-spite of this, the advantage associated with reduced O&M and minimal manual intervention offers the needed assurance of a stronger security framework. In addition, most users will be willing to wait a longer time for the first access to a system, since there is wide-spread recognition of additional processing needs in such cases.

5.4 Modifiability/Maintainability

Stimulus:	User requires to change classification, or information on expertise classification and communities
Source of Stimulus:	User requests the system for the desired change.
Environment:	Normal operating environment.
Artifact:	The function of the system, its platform, another system with which it interoperates.
Responses:	Modification must cascade through the system.
Response Measure:	No programming for change desired.

Architectural Alternatives	Risk	Sensitivity	Trade Off
Database Centric Definition	1	1, 2	2
XML Centric Definition	2, 3	2	1
Hybrid Definition	1, 2, 3, 4		

5.4.1 Architectural Alternatives

1. The database centric definition uses structures in a relational database to capture information on ontology and expertise.
2. The XML centric definition uses structures in XML to capture information on ontology and expertise.
3. The hybrid definition uses both the database and structures in XML to capture information on ontology and expertise.

5.4.2 Risks

1. The database centric definition may impose restrictions of a number of fixed columns in definitions.
2. XML definitions may pose issues in performance of the system, since many of the retrieval functions may need to interpret the XML definition for retrieval
3. XML definitions have the greatest flexibility in interpretation.
4. Hybrid definitions may permit the opportunity to move the definitions between the database (as the definitions mature) or retain them in XML (for definitions that require great extensibility).

5.4.3 Sensitivity Points

1. The definitions need to be managed in a consistent manner.

5.4.4 Tradeoffs

1. If flexibility is a consideration, then XML based definition offer a faster initial solution. The database centric definition of artifacts and information storage may not permit a good

structure for flexibility.

2. If consistency is important, a database centric solution may permit greater consistency – this is because the definitions in the database for classification and expertise will be more hard-wired compared to an open-ended XML classification.

5.4.5 Reasoning and Conclusion

There may be a need to adopt the hybrid approach, since there will be trade-offs between performance needs and consistency on one hand and flexibility on the other. There will be certain cases when speed is of essence and there will be a need to encode the information into the database. In such cases, there will be some loss of flexibility. The goal will be to review the flexible XML definitions periodically to see if the definitions are settling down and "move" these into the database.

6 ATAM Process Evaluation

6.1 Overview

The team followed an architecture evaluation process along with architecture development. The requirements and quality attributes were identified before going ahead with the architecture. The quality attributes used the common and important system quality attributes to define specific scenarios. The document provided a very brief overview of the stakeholders of the system. The definition also touched upon the need to have external consensus in terms of data interchange – this may need to isolate certain components that may need to have greater flexibility due to external considerations. The project also posed the unique challenge of a distributed workforce, with the need to ensure that a natural, greater emphasis placed on documentation as the means to communicate the architecture thought process.

6.2 Evaluation

Evaluation Area	Comments
Concise statement of architecture	• Description and figures were provided with appropriate details. • Reference was consistently made to the call-return architecture with a J2EE solution. This served as a useful abstraction.
Articulation of business goals	• Business goals were described. A high-level overview of the system has been provided.
Quality requirements in terms of a collection of scenarios	• Quality requirements were derived from the requirements. • The common scenarios were described with precision. • A utility tree was used to rank the scenarios
Mapping of architectural decisions to quality requirements	• The basic chosen architecture met the principal need for distributed and wide usage of the system with no barrier to access. • A web-based choice was natural for the nature of the system, and was used. • Integration of different types of interfaces was folded in. • A standard and well-proven J2EE architecture was used. • Considerations for security were addressed with the help of a detailed sequence diagram. • Performance needs were addressed through the use of stateless session beans with the web tier left to manage the state, and a backend database for transaction management and access speeds. • Flexibility was addressed through a hybrid approach for approaching artifact classification and storing static information.

A set of identified sensitivity and tradeoff points	• In the identified scenarios, there is an identification of the sensitivity and tradeoff points. These reflect the key elements in the architecture and the side effect of the decision on other quality attributes.
A set of risks and non-risks	• Risks have been identified. Non-risks have not been identified.

(Bass et al: 2003)

6.3 Conclusion

The business requirements definition, the quality attribute definition, architecture definition and scenario analysis provides a basis for decision making for the Knowledge Management System. This analysis has been carried out in a few of the more-important scenarios. In terms of this study, it is clear that the architecture will meet the performance and scalability needs of the quality attributes, while providing a secure method of accessing the application.

7 References

L. Bass, P. Clements, R Kazman (2003). "Software Architecture in Practice"

Kazman, Rick, Klein, Mark, & Clements, Paul. ATAM: Methods for Architecture Evaluation (CMU/SEI-00-TR-004). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.

Basili, V. R., Caldiera, G., Rombach, H. D., "Experience Factory", Marciniak, J. J. (Hrsg.), Encyclopedia of Software Engineering – Vol .I, Wiley, 1994, pp. 469-476, 1994.