SOFTWARE ARCHITECTURE

[TRACEABILITY TOOL]

**Team TechA**

Aishwarya A

Ajay Krishna A

Arul Dhana Saam Prakash A

**Team RelyCorp**

Aparrnaa R

Dinesh P

Sriram J

Thejesvini R

**Mentor**

Dr. S. Sivakumar

2016

Contents

[1. Business Context 2](#_Toc465278061)

[2. Business Goals 2](#_Toc465278062)

[3. Raw scenarios 2](#_Toc465278063)

[4. Key Architectural Drivers 2](#_Toc465278064)

[4.1. Functional Requirements 2](#_Toc465278065)

[4.2. Quality Attributes 3](#_Toc465278066)

[4.3. Business Constraints 3](#_Toc465278067)

[4.4. Technical Constraints 3](#_Toc465278068)

[5. Initial Architecture 3](#_Toc465278069)

# Business Context

SSK Process Systems Services Private Limited develops software systems for process industries. Some of the typical products produced by this company consist of unit operation simulators (such as pumps, batch and continuous reactors, absorbers, heat exchangers, distillation columns, etc.), thermodynamic packages (properties calculations, VLE calculations), process engineering calculation packages, flow sheet simulators for utilities plant, etc. This company sells customized versions to various customers and maintains the customized versions of their product.

This company now has nearly 50 customers who are traditional small scale chemical industries and captive power plants. It wishes to capture the customers in bio-tech and pharmaceutical domains. This company has an in-house software development team which uses an iterative lifecycle model for development purposes.

# Business Goals

* **Goal:** Satisfy client Service level agreements
* **Goal subject:** Project Manager of SSK Private Limited
* **Goal object:** Organization wide change in **SSK Private Limited**
* **Environment:** Competitive and Customer environment
* **Goal measure:** By the middle of April 2017, the teams of RelyCorp and TechA will provide a solution that will aid SSK Private Limited in satisfying its service level agreements. Within a month of delivering the solution, the service level agreements that are satisfied by SSK will increase by 25%, by 40% by the end of three months and 75% at the end of five months. The solution provided will enable SSK to identify the right artifacts for a change request to less than a minute.
* **Justification:** On an average SSK private limited receives 50 service request per day. Among that 40 calls are change request calls from clients. Currently, SSK private limited is able to satisfy only 15 change request calls. SSK private limited is not able to satisfy around 60% of the change request calls because they are spending around 2 hours to 2 days to find the artifacts that will be affected for a particular change request.

# Raw scenarios

The scenarios for the subsystem A can be viewed in this document: <<to append>>

The scenarios for the subsystem B can be viewed in this document: <<to append>>

# Key Architectural Drivers

The key architectural drivers for the system can be enlisted as follows:

# Functional Requirements

Please refer to the Requirements Document which contains the functional requirements for the system.

# Quality Attributes

We derived the corresponding quality attributes from the business needs of the system. Hence, by addressing these quality attribute concerns we can satisfy the needs and achieve the goal.

The prioritized list of quality attributes related to the system is given in the below table:

|  |  |  |  |
| --- | --- | --- | --- |
| Quality attribute | Description according to context | Business Needs | Priority |
| Reliability | The system must retrieve the right artifacts related to a change request at any point of time. This quality attribute cannot be achieved to the fullest. However, compared to the system-as-is, reliability of the system will be improved. | Deliver right product versions to customers | HIGH |
| Usability | The users of the system must be able to use the system easily so that the number of wrong linkages between artifacts can be reduced and the productive time of the users is increased. | Reduce Service Time for Change Request | HIGH |
| Performance | The artifacts related to a change request should be retrieved in less than a minute. | Reduce Service Time for Change Request | HIGH |
| Security | Authorize the users who use the system and make sure that no person can deny the fact that he/she hasn’t made a linkage. | Deliver right product versions to customers | HIGH |

# Business Constraints

* The product must be delivered to the customer by April 15th, 2017.
* The costs involved for the development of the product will be approximately $ 9000.
* The system must use only open source tools for development and maintenance. No other licensed products shall be purchased.

# Technical Constraints

* Java must be chosen as the programming language to enhance portability (through platform independence).
* MongoDB must be chosen as a data repository.
* HTML5/CSS3 must be chosen to develop the front end.
* Apache Tomcat must be chosen as a web and application server.

# Initial Architecture

* 1. Description of architecture

C:\Users\Aish\AppData\Local\Microsoft\Windows\INetCacheContent.Word\Untitled Diagram (2).png

Figure 1: Architecture for Model Based Traceability Tool

The overall notional system architecture is designed based on 3 tier architectural pattern.

* Application tier
* Service tier
* Data tier

The users are going to interact with the system through the application tier. It composes of model manager and traceability tool. Model manager is going to be a desktop based application and traceability tool will be a web based application.

The .wars files pertaining to traceability manager are hosted in the traceability server. The users of the traceability managers connect to this server using standard HTTP/HTTPS protocol. The service layer also contains the service providers which is going to host those specific services pertaining to model manager and traceability manager. All the service level interactions are going to be handled using RESTful web services.

The data tier is going to hold the MongoDB which will be our data source. The service provider has web services connecting to this MongoDB using J Connector.

# 5.2 Design Decisions for splitting the subsystems

We are splitting the system in two parts. The first subsystem is model manager and second system is traceability. We have identified the responsibilities of the entire system, classified them and categorized them into two major subsystems:

1. Model manager- a model based definition created by the project manager.
2. Traceability portal – a browser based application where the users log artifact instance information and their linkages to the respective artifacts. They can also pull traceability from this system. This traceability portal is used by Users (Developers, Testers, BA’s, Architect, Designer).

We have split this on the basis of following reasons:

* Both of these subsystems do not share responsibilities.
* If the teams get stuck at some point in development, an even distribution to both of this subsystems won’t occur.
* It is always better to split the responsibilities among the systems to keep the project morale up.
* Addressing different users.

# 5.3 How the design choices made support the business goals in terms of quality attributes?

In order to retrieve the right project artifacts, the proposed model manager - a model based system which performs model definitions ensures reliability by providing role based access for artifact linking, thereby mitigating the probability of wrong artifacts getting linked.

We have narrowed down to MongoDB as our design choice for data repository because there is a need to retrieve artifacts as soon as possible. Non-relational database reduces latency (performance) as we have document structures instead of schema definitions in relational database. Moreover, there is not going to a fixed schema. The schema is going to change depending on how model is designed by the model manager. Document based NoSQL database can accommodate to these changes more easily than relational databases.

We decided to go with a web service based middleware service because it improves interoperability between the subsystems. Also, the client had mentioned that there is going to be growth of business in future. When the business is growing, there will be different systems which might be integrated with the existing system and these systems might be in .net or VB. In that case they can reuse these services rather than writing again. Also, the client had mentioned that the stakeholder had ideas of moving the components to cloud in future. If it’s going to be web services, it is going to cause less manual changes. If it is going to be RPC or some socket or simple FTP over shared repository, there is going to be portability issues and availability issues if the server on which it is operating goes down.

Currently, in the system-as-is the artifacts are retrieved manually and it takes a long time to retrieve. Hence by having an automated artifact retrieval, we are ensuring that every related artifact is retrieved within 1 minute. For this, services are exposed to the traceability manager using RESTful web services.

# 5.3 Tradeoff decisions

However, inconsistency is a serious issue to be taken care. But, in our system we have only one database server and it is not going to be scaled horizontally in the near future, the problem of inconsistency becomes negligible. Considering this, we gave more priority to the changing schema concern of reliability over inconsistency concern.

RESTful web service was chosen over SOAP because, REST web services allow us to send requests with JSON/XML/HTML payloads and can receive responses over the same whereas SOAP restricts the usage of XML. We have JSON data coming from our Mongo DB hence REST offers more flexibility to operate.