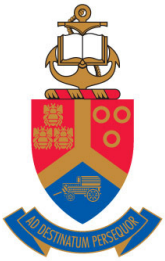

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

COS301
Mini Project
Architectural Requirements Specification

Group 1B

David Breetzke	*u12056503*
Tienie Pritchard	*u12056741*
Ephiphania Munava	*u10624610*
Paul Engelke	*u13093500*
Ryno Pierce	*u12003922*
Aluwani Simetsi	*u11322935*
Mkhabela Phethile	*u12097561*
New Member	*uxxxxxxxx*

GitHub: Visit

Version 1
March 9, 2015

Contents

1 Access and Integration Requirements 3

2 Access Channels 4

3 Integration Channels 5

4 Architectural Responsibilities 6

5 Quality Requirements 8

5.1 Scalability . . . 8
5.2 Performance Requirements . . . 8
5.3 Maintainability . . . 8
5.4 Reliability and Availability . . . 8
5.5 Security . . . 8
5.6 Monitorability and Auditability . . . 8
5.7 Scalability . . . 8
5.8 Testability . . . 8
5.8.1 Type of Quality: . . . 8
5.8.2 Priority: . . . 8
5.8.3 Description: . . . 9
5.8.4 Stake Holder: . . . 9
5.8.5 Context: . . . 9
5.8.6 Measurable Specification: . . . 9
5.9 Usability . . . 10
5.9.1 Type of Quality: . . . 10
5.9.2 Priority: . . . 10
5.9.3 Description: . . . 10
5.9.4 Stake Holder: . . . 10
5.9.5 Context: . . . 10
5.9.6 Measurable Specification: . . . 11
5.10 Integrability . . . 11
5.10.1 Type of Quality: . . . 11
5.10.2 Priority: . . . 11
5.10.3 Description: . . . 11
5.10.4 Stake Holder: . . . 12
5.10.5 Context: . . . 12

5.10.6 Measurable Specification: 12
5.10.7 Required Integration Channels: 12
5.10.8 Protocol Requirements on Integration Channels: . . . 13
5.10.9 Quality requirements on integration channels: 13

6 Architecture Constraints **14**
6.0.10 Description: 14
6.0.11 Technical constraints: 14
6.0.12 Business constraints: 14

7 Technical Constraints: **14**
7.0.13 Programming technologies: 14
7.0.14 Platforms supported and operating system: 15
7.0.15 Hardware: 15
7.0.16 Use of a specific library or framework: 15

8 Business Constraints: **15**
8.0.17 Schedule: 15
8.0.18 Budget: 16
8.0.19 Software licensing restrictions: 16

1 Access and Integration Requirements

[insert text]

2 Access Channels

[insert text]

3 Integration Channels

[insert text]

4 Architectural Responsibilities

Responsibilities which need to be addressed by the software architecture are:

- To be able to host and provide an environment for the execution of the system
- To provide an Infrastructure that provides a web access channel
- To provide an Infrastructure that provides a mobile access channel
- To integrate with the LDAP repository
- To provide an infrastructure that allows module plug-ability
- Provide infrastructure that allows in-dependency from core modules and add on modules
- To provide an infrastructure to integrate the system into the department's website
- To provide an infrastructure to integrate the Hamster marking system
- To provide a flexible reporting framework that allows users to define and generate their own reports
- To provide an Infrastructure that provides access to system resources (Communication resources and memory)
- Providing a storage environment for media (images, podcast, and video) as well as uploaded text files (pdf, odf, and doc)
- File streaming
- Persistent domain data
- Providing an event infrastructure
- Providing an execution environment for processes
- To provide an Infrastructure that integrates with an email/sms server
- Providing a moderated and controlled environment

- To provide an Infrastructure that integrates with the computer science data source adapter
- Executes processes
- To provide an Infrastructure that enforces security including authentication, confidentiality and non-repudiation

5 Quality Requirements

5.1 Scalability

[insert text]

5.2 Performance Requirements

[insert text]

5.3 Maintainability

[insert text]

5.4 Reliability and Availability

[insert text]

5.5 Security

[insert text]

5.6 Monitorability and Auditability

[insert text]

5.7 Scalability

[insert text]

5.8 Testability

5.8.1 Type of Quality:

System Quality

5.8.2 Priority:

Critical

5.8.3 Description:

Testability measures how easy it is to create testing standards for a system and its individual components, theses standards are tested to evaluate if a criteria has been met. Thus software testability is the point to which the software system supports testing in some context. Hence if the software testability is high finding faults in the system is easier.

5.8.4 Stake Holder:

- Persons who operates the system : Administrator, Maintenance Operator and Tech-team.
- Persons who benefits from the system : Lectures, Teaching Assistance, Tutors, Students and Guest.

5.8.5 Context:

- Stimulus : The testing is performed by tester (these might be system testers, integration testers and even the end user).
- Artifact : The target of the attack can be the system or the data in the system.
- Environment This attack can come from the user of the system or an outsider like a hacker.
- Response : The system has to authorize certain actions and responses for each of the given tasks.
- Response Measure : The measure of the system and it functionality before, during and after the attack.

5.8.6 Measurable Specification:

- Understand-ability : The point at which the component of the system that being tested is self-explanatory.
- Separation of concerns : The point, at which the component of the system that's being tested has a well-defined responsibility.
- Observe-ability : The point, at which the component of the system thats being tested become possible to discern the test results.

Component Under Test
Is a test that restrictions the scope of the used software to a ration of the system that is being tested.

- Controllability : The point, at which the system thats being tested becomes possible to control the state of the component under test as required.
- Isolate-ability : The point, at which the system thats being tested becomes possible for the component under test to be tested in isolation.

5.9 Usability

5.9.1 Type of Quality:

User Quality

5.9.2 Priority:

Critical

5.9.3 Description:

Usability describes how the system meets the requirements of the stake holders by being instinctive on condition that good access for incapacitated users is provided, and resulting overall great user experience. Thus software usability refers to the ease of use and learn-ability of the system. In other words how user-friendly is it.

5.9.4 Stake Holder:

- Persons who benefits from the system : Lectures, Teaching Assistance, Tutors, Students and Guest.

5.9.5 Context:

- Stimulus : The stake holder wants to use the system efficiently.
- Artifact : The target of use which is the system.
- Environment : This stake holders action with which the usability quality is concerned.

- Response : The system provides the stake holder with features that the stake holder will or might need.
- Response Measure : The response of the system and it functionality is measured by the number of errors, number of problems encountered, user satisfaction and time taken per task.

5.9.6 Measurable Specification:

Cognitive Modelling Methods
Cognitive Modelling Methods involves creating computational method in order to estimate the time it will take people to perform given tasks.

- Human Processor Model : This model was developed to calculate how long it takes an individual to perform a task. A table is given with amount of times a user would take to execute an action i.e. move eye to look at the screen 230ms.
- Keystroke level modelling : Very much like the GOMS version but simplifies assumptions so that calculation time and complexity is reduced.
- Heuristic Evaluation : This measurable method involves bringing in a set of experts that will evaluate the usability of your system based on their prior knowledge and research.

5.10 Integrability

5.10.1 Type of Quality:

User Quality

5.10.2 Priority:

Critical

5.10.3 Description:

The capability of making components of a single system that is isolated and developed separately work together correctly.

5.10.4 Stake Holder:

- Persons who operates the system : Administrator, Maintenance Operator and Tech-team.

5.10.5 Context:

- Stimulus : The stake holder wants to use to configure, maintain, update and use the system.
- Artifact : The target of use which is the system.
- Environment : This stake holders action with which the system has to conform to.
- Response : The system allows the stake holders to interact with it.
- Response Measure : The response of the system and it functionality after the system has been refactored.

5.10.6 Measurable Specification:

- Spread load across time : This can be achieved by making use of Queuing as a tactic.
- Reduce communication load : This can be accomplished by using strategies like compression, batching and course grained services.
- Fault prevention : This can be addressed by making use of persistent messaging.
- Component Application : We encounter naming service, trader service and interface/ contract repository which is addressed with integerability.
- Security : The use of encryption and restricting accessibility will address the security insures we might come across.

5.10.7 Required Integration Channels:

- The Required Integration Channels are those that the system will require to make it accessible by the stake holders.
  - Graphical User Interface.

- The Required Integration Channels are those that the server will host the system.

5.10.8 Protocol Requirements on Integration Channels:

- HTTP (REQUEST/GET/POST requests)
- SOAP (Messaging Protocol)

5.10.9 Quality requirements on integration channels:

- ISDN (Simultaneous digital transmission of data, video and voice)

6 Architecture Constraints

6.0.10 Description:

Below are system constraints that have a significant bearing on the Buzz system architecture. These constraints are divided into two categories.

6.0.11 Technical constraints:

- These are fixed technical design decisions that absolutely cannot be changed.

6.0.12 Business constraints:

- These are unchangeable business decisions that in some way restrict the software architecture.

7 Technical Constraints:

7.0.13 Programming technologies:

Buzz system will primarily be implemented using various Java view technologies and frameworks. These technologies offer important functionality for distributed web applications and should provide strong foundation for Buzz.

Java technologies to be used include:

- Java EE
- JPA(Java Persistence API)
- JPQL(Java Persistence Query Language)
- JSF(Java Server Faces)

Other technologies include: HTML, AJAX and CSS

- SOAP (Simple Object Access Protocol) will be used as interface for exchanging information on the web services and across the network.
- UTF-8 must be used for encoding to keep information safe and secure.
- GitHub will be used for a web based repository to store all of the information regarding the project.

7.0.14 Platforms supported and operating system:

The CS (Computer science) systems operate mainly on Linux. Buzz should also support Linux as it will need to be integrated with other existing systems such as:

- Computer Science LDAP repository
- Computer Science Portal
- World Wide Web

Buzz should support major browser clients that include:

- Mozilla Firefox
- Google Chrome
- Safari
- Internet Explorer

Buzz will not support the Android platform because of the limited time allocated for the development.

7.0.15 Hardware:

Buzz will be hosted by the computer science server and therefore should support the servers hardware capabilities while providing the recommended functionality to the users.

7.0.16 Use of a specific library or framework:

Any specific framework or library can be used/should be used for Buzz provided that its open source.

8 Business Constraints:

8.0.17 Schedule:

Buzz System must be implemented within the stipulated time period and should be functional at the end of the deadline.

8.0.18 Budget:

A budget has not been allocated for buzz system and therefore its recommended that any technology or product that requires capital should not be used.

8.0.19 Software licensing restrictions:

Buzz should adhere to all software licensing restrictions that are stipulated by the software owners.