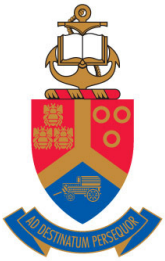

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

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*
James Murray	*u12030733*

GitHub: Visit

Version 1
March 10, 2015

Contents

1 Access and Integration Requirements

1.1 Access Channels

1.1.1 Human Access Channels

- **Students** will be able to access Buzz from the Computer Science website (or the equivalent URL). It is therefore required that students have access to an internet connection and any of the major web browser clients available, some of which include Mozilla Firefox, Google Chrome, Safari, Internet Explorer and Opera.

- **Lecturers** will be able to access Buzz in the same way that students can, i.e. via a web browser. However, unlike students, lecturers will have access to more services that are not available to students.

- **Administrators**, being classified as the group of individuals that monitor the system's operation, as well as manage and maintain the system, may have access to the system through a web browser (as before with students and lecturers) with far less restrictions on available services, if any at all. Administrators will also have access to the back-end of the system through the host servers, locally.

- **Auditors** may require access to service logs. These services logs may be accessed via the host servers or other server on which the service logs for Buzz are located. Access to these logs may be provided remotely, although this may depend on the security risks of allowing such access.

1.1.2 Access Channel Protocols

- **HTTP** is one option as a protocol for server-client communication (between the web browser and the server). Due to its statelessness, it provides for speed and lower bandwidth usage, which is ideal for users that have slow connections or limited bandwidth, or both. However, HTTP is not secure.

- **HTTPS** is the other option for server-client communication. It works in conjunction with SSL to provide a secure connection. This will be most valuable in providing secure authentication when a user attempts to log in to Buzz.

1.1.3 Other Access Channels

- **Android Client:** An optional means of access to Buzz for users is the Android client application. However, this system is out of scope for the mini-project.

1.2 Integration Channels

1.2.1 Systems with which Buzz must integrate

- LDAP (Lightweight Directory Access Protocol) Server
  - Each user will need to have an LDAP account (i.e. username and password) in order to connect to the LDAP server which they will use to log into the buzz system
  - University of Pretoria Computer Science students automatically have these accounts as theyre registered on the CS web.
  - However, this is a requirement if other Universities want to integrate the buzz system into their systems.
- CS Web
  - The buzz system will be integrated into the CS Web (http://www.cs.up.ac.za/) in order to allow users to access the functionality.
  - Most likely, the buzz system will replace the current CS webs forum system, which will allow modules to link their personal buzz space from the module site.
  - Thus, it is necessary for the CS web to have access to the buzz system.
- WWW (World Wide Web)
  - The buzz system will need to be hosted to a server and needs to have access to that servers individual functionality services (i.e. Apache, PHP, etc.)
  - It is necessary to integrate the buzz system with the WWW in order to allow people to actually access it (i.e. via a URL).
- DBMS (Database Management System)
  - The buzz system will need to connect to some database which will store all user information such as profiles, passwords, usernames, etc.

  - This storing of information will be done via some DBMS (Such as MySQL).

1.2.2 Integration Channels and Protocols

- GUI (Graphical User Interface)
  - This will allow users to actually interact with the Buzz System so that they can post, comment, etc.
    - Protocols which will be used here are: HTTP
- The server which the buzz system is hosted on.
  - The server will provide a lot of the functionality needed for the buzz system to actually be able to handle and respond to requests.
  - It will also provide the core access channel for users.
  - Various protocols need to be used here, such as
    - HTTP
      - For processing and handling GET/POST requests from users (i.e. conformity to the RFC (Request for Comments standard).
    - SOAP (Simple Object Access Protocol).
      - This is a messaging protocol that allows programs which run on disparate operating systems to communicate using HTTP and its XML.
      - This is necessary to ensure that users between different operating systems can communicate using a standard interface, so that there are no technological gaps which will hamper communication among different users.
    - TCP/IP (Transmission Control Protocol/Internet Protocol)
      - TCP is needed to connect to the host server's socket so that data may be delivered in streams of octets to users connected to the Buzz System's server.
      - IP is needed to deliver packets from the Buzz System server to the clients based on the relevant IP addresses in the packet headers.
    - HTTPS (HTTP Secure)
      - This is needed for secure communication between the server and the clients.

- It is very important in terms of security, especially in terms of communication of passwords/user names/personal information.

2 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

3 Quality Requirements

3.1 Scalability

3.1.1 Type of Quality:

Maintenance,System

3.1.2 Priority:

Important

3.1.3 Description:

As soon as an instance of a server reaches the maximum amount of connection that it can handle, a new instance of a server should be spawned. When there are multiple instances of servers running, and the amount of connections to the system can be handled with fewer instances, one or more should be terminated with its connections migrating to other instances.

3.1.4 Stake Holder:

Server Maintenance Staff

3.1.5 Context:

When new connections are made to the Buzz system or existing ones are terminated, it should scale.

3.2 Performance Requirements

3.2.1 Type of Quality:

User Quality

3.2.2 Priority:

Critical

3.2.3 Details:

Any response time longer than 1 second would break the user's though flow. Therefore if a server is overloaded, and the amount of requests in its request queue is higher than the amount that can be handled per second, a new server should be spawned to reduce stress on the current one. When more than one instance of a server is running, load balancers have to implemented to mitigate the workload. Jobs that do not need immediate responses, can be handled at times that the system is under little strain. Since the persistent data in the system will be accessed on almost every request to the server, optimal structures and indexes are very important.

3.2.4 Stake Holder:

Users

3.2.5 Context:

When a user makes a request to the server via the Buzz interface.

3.3 Maintainability

3.3.1 Type of Quality:

Maintenance, System

3.3.2 Priority:

Critical

3.3.3 Context:

Any module developed for the Buzz system has to be maintainable, and easily replaceable.

3.3.4 Details:

By using service contracts for all modules, with coupling ratios being kept as low as possible, complete pluggability must be achieved. Any module has to be swappable or rewritable, without making any changes to any other module. All code has to adhere to strict coding standards.

3.3.5 Stake Holder:

Maintenance Staff

3.4 Reliability and Availability

3.4.1 Type of Quality:

User Quality

3.4.2 Priority:

Critical

3.4.3 Context:

Reliability in providing a service to the users that will react predictably is key. The system needs to be online and functional.

3.4.4 Details:

The system should not crash, freeze or hang. The system must be available 99

3.4.5 Stake Holder:

The users and administrators/moderators

3.5 Security

3.5.1 Type of Quality:

System

3.5.2 Priority:

Critical

3.5.3 Context:

Buzz must not put other systems with which it integrates at risk. Users privacy must be protected at all times.

3.5.4 Details:

User log-in should be secure. Encryption should be used wherever possible. It should be assumed that any attackers would know the internal structure of the system, and implementation should be done accordingly. Administrators accounts and details should be further protected with passwords of certain length and strength.

3.5.5 Stake Holder:

The systems with which Buzz integrates as well as Buzz users.

3.6 Monitorability and Auditability

3.6.1 Type of Quality:

Maintenance, System

3.6.2 Priority:

Important

3.6.3 Context:

History should be kept on user activity so that system abuse, suspicious patterns and other unwanted behaviour can be identified. Logs must be detailed so that they are useful in bug-finding.

3.6.4 Details:

User history should be logged. Administrator actions should be logged. System maintenance and downtime should be logged. A separate back-end account type should be available for auditing. Logs should be accessible in read-only mode. Logs should be accessible via secure login only.

3.6.5 Stake Holder:

Administrators, Maintenance staff and The Buzz System.

3.7 Scalability

3.7.1 Type of Quality:

3.7.2 Priority:

3.7.3 Context:

3.7.4 Details:

3.7.5 Stake Holder:

3.8 Testability

3.8.1 Type of Quality:

System Quality

3.8.2 Priority:

Critical

3.8.3 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.

3.8.4 Details:

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.

3.8.5 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.

3.8.6 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.

3.9 Usability

3.9.1 Type of Quality:

User Quality

3.9.2 Priority:

Critical

3.9.3 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.

3.9.4 Details:

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.

3.9.5 Stake Holder:

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

3.9.6 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.

3.10 Integrability

3.10.1 Type of Quality:

User Quality

3.10.2 Priority:

Critical

3.10.3 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.

3.10.4 Description:

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

3.10.5 Stake Holder:

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

3.10.6 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.

3.10.7 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.

3.10.8 Protocol Requirements on Integration Channels:

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

3.10.9 Quality requirements on integration channels:

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

4 Architecture Constraints

4.0.10 Description:

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

4.0.11 Technical constraints:

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

4.0.12 Business constraints:

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

5 Technical Constraints:

5.0.13 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.

5.0.14 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.

5.0.15 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.

5.0.16 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.

6 Business Constraints:

6.0.17 Schedule:

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

6.0.18 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.

6.0.19 Software licensing restrictions:

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