Gynaecological Patient Information management System:

Architectural Requirements

Team Pentec:

Ruth Ojo 12042804
Liz Joseph 10075268
Trevor Austin 11310856
Maria Qumayo 29461775
Lindelo Mapumulo 12002862

Final Version

July 30, 2015

Contents

1 Introduction

2 Architectural requirements

2.1 Architectural Scope

This section discusses the boundaries and extent or range of view, outlook, applications, operations or effectiveness the system software architectur needs to address. More spesificaly we will be looking at the following three topics in depth. Persistence, Reporting and Process execution.

2.1.1 Persistence

Buzz Space needs to be percistant in order for states to be stored and outlive the many processes done. It will do so by making use of the following:

- Databases, to store and retrieve user accounts data and information.
- Java Data Objects (JDO), a specification of Java object persistence. With it's great transparency feature of the persistence services to the domain model.
- System prevalence, a technique that joins system images and transaction journals to achieve percistancey.

2.1.2 Reporting

It is crucial that the Buzz Space gives the user some sort of reporting and feed back after activities such as successful posts, voting(either up or down), deleted or hidden posts, read and unread posts or any other change of state. This will be achieved by sending the user an email to notify him of any changes.

2.1.3 Process execution

Process infrastructure and execution deals with the essential operation components, such as policies, processes, equipment, data and internal operations, for overall effectiveness. In providing a good Process infrastructure for the Buzz Space system, we aim to:

- Reduce duplication of effort
- Ensure adherence to standards (both coding and design)

- Enhance the flow of information throughout an the system
- Promote adaptability necessary for a changeable environment (Including the different human access channels.)
- Ensure interoperability among organizational and external entities
- Maintain effectiveness

2.2 Access and integration requirements

2.2.1 Human access channels

2.2.2 System access channels

2.2.3 Integration channels

2.3 Quality requirements

2.3.1 Critical Architectural Requirements

2.3.2 Description

This is the most critical requirement for the patient information management system. Patient's information should not only be confidential, but levels of user authorisation must exist. The existing information should not be modifiable by users that have no access to the information; this also applies to outside intruders.

2.3.3 Justification

The whole system should not be penetrable by an intruder; the general public should not have access to any of the information about the patients. The system should also make sure that each user can only access the attributes that applies to them.

2.3.4 Mechanism

1. Strategy:

    - Encryption: Patient data is confidential and should be encrypted. Not every patient has to be encrypted, but the data should not imply to which patient it belongs to. The patient's personal information could be encrypted instead of all the attributes.
    - Authorization: A user's access level and identity needs to be determined before they can access the system.
    - Store log information: Although this applies to audibility, it helps to know the nature of a security threat. If new threats are noted, more security features can be implemented.

2.3.5 Description

This ensures that a user will be able to use the system, with ease. The system should provide support to the user.

2.3.6 Justification

Patient information management system is user-oriented. How the users interact with the system is a critical, and this should be done with little to no effort. The system should appear easy to use and should not, at any point, baffle the users.

2.3.7 Mechanism

1. Strategy:

    - A tutorial on how the patient information management system works. A user can be initiated into the system, the first time they use it. Or they can enable the tutorial until they're familiar with the functionality.
    - Enable the user to troubleshoot their problems. Frequently asked questions or frequent problems could assist with this aspect. A user will be provided with predefined help options such that they will not need to contact the system's administrator, for assistance.
    - Provide descriptive headings that make navigation easier. Headings should not be ambiguous. A user should know what to expect when they select a certain heading.
    - Error signals should be displayed to the user, if some user-inflicted error occurs. The necessary steps to rectify this problem must be provided.
    - A user should be able to undo their action, should they be aware of their mistake.

2. Architectural Pattern(s):

    - Model-View-Controller: This separates the user interface from the rest of the system (Bass and John). A user should only interact with a simple interface that was designed for them. This is describable for patient information management system because the users don't necessary have an adept understanding of the lower levels of the system.

Description

Scalability is an essential aspect of a system and is the ability of a system to be easily enlarged in order to accommodate a growing amount of work.

Justification

The PIMS should allow for hundreds of concurrent users, as such the system must be able to handle such a number without breaking down or reducing performance.

Mechanism

1. Strategy:

    - Clustering: using more resources by running many instances of the application over a cluster of servers or instances, to ensure system resources are not strained by a high workload.
    - Efficient use of storage: data storage can be efficiently used through compression of the data (reducing data size to make room for more) paging (ensuring that primary storage is used only for more crucial data) as well as de-fragmentation (organizing the data into continuous fragments and free more storage space). by ensuring that no data duplications occur, storage space can be conserved, thus the load on system resources will be reduced.
    - Efficient persistence: through indexing and query optimization, the amount of system power used to persist a database will be reduced, as data retrieval will be quicker and costly queries will be done without, thus also reducing system load. In addition, connections can be grouped and accessed via a central channel in order to aid persistent storage to the database.
    - Load Balancing: by spreading the systems load across time or across resources the load on the system can be distributed, therefore no system resource will be heavily strained. In the case that the limit for a server has been reached, a new instance or so will have to be created in order to handle the number of increasing requests. On the other hand, if the usage of a server is way below the capacity, the number of instances will have to be reduced.
    - Caching: to ensure no duplication or repeated retrieval of frequent objects or queries; a separate module can facilitate caching; thus system resources will not be used up unnecessarily.

Description

The PIMS should be accessible at almost all times, in particular during the peak operational hours of the hospital. This accessibility will be limited to

the hospital network only, so as to ensure no information can be changed without approval.

Justification

The reliability and availability requirements are very important seeing as the information to be kept on the system is highly valuable to the medical staff, as the need up-to-date information concerning the patients they are dealing with (lives could be at risk). With this in mind, only a downtime of less than 2 hours, at most twice a month will be allowed, so as too allow for the medical staff to maximally use the system. A high reliability rate is recommended to ensure that users do not encounter any errors and/or data corruption in their use of the system. The only leeway that will be given for errors, is to have at most one.

Mechanism

1. Strategy:

    - Clustering: using more resources by running many instances of the application over a cluster of servers or instances; therefore if any server should fail, the reliability and availability of the system will not be compromised.
    - For reading from and writing to the database, we will ensure that no parallel updates are possible through enforcing the use of a single object to stream all database transactions; thus reducing inaccuracy that would be a result of data redundancy.
    - Use of more resources: This would heavily reduce system downtime, as a temporary server can be run while the other is maintained.

2.3.8 Important Quality Requirements

2.3.9 Maintainability

2.3.10 Testability

2.3.11 Reliability and Availability

2.3.12 Integrability

2.3.13 Nice to have quality requirements

2.3.14 Description

This ensures that a user will be able to use the system, with ease. The system should provide support to the user.

2.3.15 Justification

Patient information management system is user-oriented. How the users interact with the system is a critical, and this should be done with little to no effort. The system should appear easy to use and should not, at any point, baffle the users.

2.3.16 Mechanism

1. Strategy:

    - A tutorial on how the patient information management system works. A user can be initiated into the system, the first time they use it. Or they can enable the tutorial until they're familiar with the functionality.
    - Enable the user to troubleshoot their problems. Frequently asked questions or frequent problems could assist with this aspect. A user will be provided with predefined help options such that they will not need to contact the system's administrator, for assistance.
    - Provide descriptive headings that make navigation easier. Headings should not be ambiguous. A user should know what to expect when they select a certain heading.
    - Error signals should be displayed to the user, if some user-inflicted error occurs. The necessary steps to rectify this problem must be provided.
    - A user should be able to undo their action, should they be aware of their mistake.

2. Architectural Pattern(s):

    - Model-View-Controller: This separates the user interface from the rest of the system (Bass and John). A user should only interact with a simple interface that was designed for them. This is describable for patient information management system because the users don't necessary have an adept understanding of the lower levels of the system.

Description

Scalability is an essential aspect of a system and is the ability of a system to be easily enlarged in order to accommodate a growing amount of work.

Justification

The PIMS should allow for hundreds of concurrent users, as such the system must be able to handle such a number without breaking down or reducing performance.

Mechanism

1. Strategy:

    - Clustering: using more resources by running many instances of the application over a cluster of servers or instances, to ensure system resources are not strained by a high workload.
    - Efficient use of storage: data storage can be efficiently used through compression of the data (reducing data size to make room for more) paging (ensuring that primary storage is used only for more crucial data) as well as de-fragmentation (organizing the data into continuous fragments and free more storage space). by ensuring that no data duplications occur, storage space can be conserved, thus the load on system resources will be reduced.
    - Efficient persistence: through indexing and query optimization, the amount of system power used to persist a database will be reduced, as data retrieval will be quicker and costly queries will be done without, thus also reducing system load. In addition, connections can be grouped and accessed via a central channel in order to aid persistent storage to the database.

- Load Balancing: by spreading the systems load across time or across resources the load on the system can be distributed, therefore no system resource will be heavily strained. In the case that the limit for a server has been reached, a new instance or so will have to be created in order to handle the number of increasing requests. On the other hand, if the usage of a server is way below the capacity, the number of instances will have to be reduced.
- Caching: to ensure no duplication or repeated retrieval of frequent objects or queries; a separate module can facilitate caching; thus system resources will not be used up unnecessarily.

3 Architectural Patterns and Styles

3.1 Model-View-Controller (MVC)

Model-View-Controller is an architectural pattern that divides software applications into three interconnected parts.

The controller is responsible for translating requests in to responses (Nadel, 2012). The controller 'inserts' the response back into the view, which could be html or it's equivalents.

The view translates the response, from the controller, into a visual format for the client (Nadel, 2012). The view also sends requests to the controller.

The model's task is to maintain state and give methods for changing this state. Typically, this layer can be decomposed to:

- Service layer - provides high-level logic for dependent parts of an application.
- Data access layer - services objects invoke this layer, which provides provides access to the persistence layer.
- Value object layer - provides data-oriented representations of terminal nodes in the model hierarchy.

3.2 Technologies

input./Technologies.tex

3.3 Tactics and strategies

4 Architectural responsibilities

5 Architecture constraints