System Design Document - Yellow Project

William Diedrichsen Marstrand, Dennis Thinh Tan Nguyen,
Jacob Mullit Moeiniche, Thor Valentin Olesen

October 25, 2015

1 Introduction

1.1 Purpose of the System

In **AUTOSYS**, the main client is the **SystemReviewClient** who needs to set up a **STUDY** consisting of one or several **PHASES**. A **PHASE** will consist of one or more **TASKS** which need to be completed before moving on to the next **PHASE**. The **StudyParticipants** can work in different **ROLES** which we anticipate will be either as **REVIEWER** or **VALIDATOR**. Also the **StudyParticipants** will be working together on a **STUDY** in **TEAMS**, but multiple teams will not be working on the same **STUDY**. Furthermore **TEAMS** working on the same **STUDY** will be considered as one big **TEAM**.

1.2 Design Goals

- **Ease of use**:
  Goal: It should be relatively easy to setup a study configuration because it makes up the foundation that all study work processes rely on.

  The end user may have a low level of computer expertise potentially resulting in the wrong setup of a study. This can happen because the user cannot find or access the resources required for setting up a study or they become frustrated if they have to go through many windows. However, these usability traits should not compromise with the system functionalities. The system must still be sufficiently complex in order to provide a variety of ways to setup the configuration. A primary focus on usability traits has been chosen due to the scope and time span of the project. This design goal is a refinement of the non-functional requirement usability in Requirement Analysis Document (section 3.3.1).

- **High Reliability**:
  Goal: The server and study configuration ui must be reliable, handle system failures and wrong user input.

  The server should handle system failures (e.g. exceptions or network failures). Due to the client-server architecture, it is important that the server can automatically reboot upon system failures. As a result, other clients can continuously access the server. Also, the server should ensure that incomplete data transfers are resumed in case of network failures. Further, the Study Configuration UI should handle invalid user input and ensure the input is only sent to the server when correct. Finally, the server should have a backup of the database in case of corruption. These decisions ensure that the work of the end user is handled properly and the server is available when needed. This design goal is a refinement of the non-functional requirement "high reliability" in Requirement Analysis Document (section 3.3.2).

- **Scalability**:
  Goal: The response time of the system must not degrade dramatically with the number of users and concurrent studies.

  The system is used by multiple teams with several users working on different studies. The work carried out by these users could be done concurrently and so the system will have a big workload when multiple requests are sent to the server. Since the users need study data quickly to conduct their research, it will be cumbersome if the users have to wait for a long time to get the data. Thus, the server should support quick data access for users. However, the system must do this in a way which takes memory into account to avoid an extensive resource consumption. This design goal is a refinement of the non-functional requirement "performance" in

Requirement Analysis Document (section 3.3.3).

- **High performance**:
  Goal: to ensure a quick and responsive user interface by using a database, which facilitates fast data processing.

  To achieve this, we will implement an entity framework database, which utilizes Microsoft's .NET framework to optimize database queries. This takes advantage of optimized queries and features implemented by Microsoft database experts. Furthermore, a future feature may be implemented to cache the result of large queries for quick access. This allows the server to respond hastily to large queries, which have previously been used. This will only be implemented for queries on data that is less likely to change. As a result, a memory trade-off may occur, which can be negated by using limits on how much memory and how many queries are cached. By way of example, the last x number of query results to the server could be cached for repeated use. The design goal is a refinement of the non-functional requirement "high performance" in the Requirement Analysis Document (section 3.3.3)

2 Proposed Software Architecture

2.1 Subsystem Decomposition

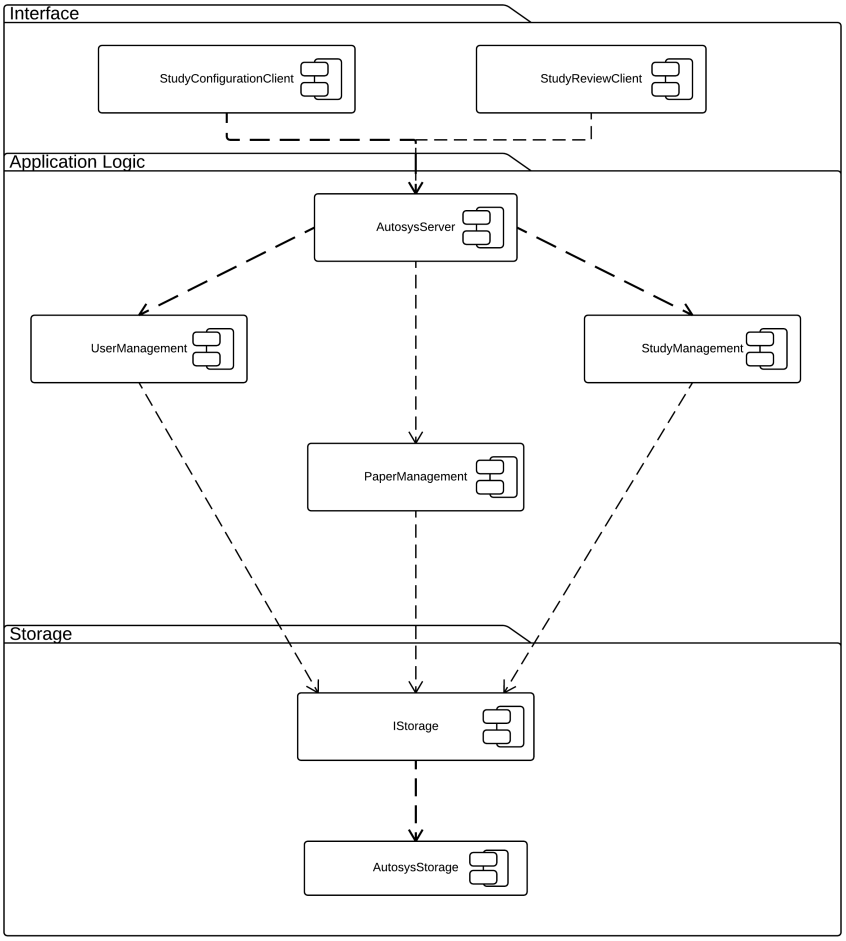


Figure 1: Autosys subsystem decomposition (UML Component Diagram, layers shown as UML packages)

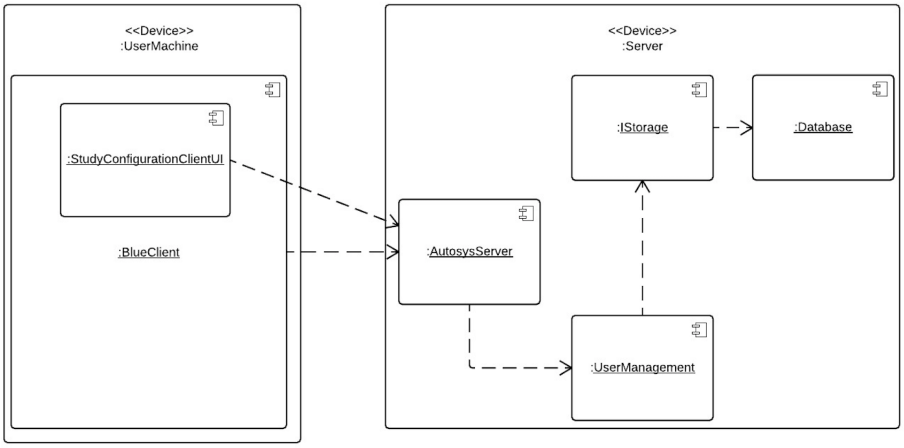
The subsystems are identified from the functional requirements of the Autosys RAD document. A three-tier architectural style has been used for the decom-

position of the system where a **StudyConfigurationClient** provides a front end for users to initiate all use cases related to setting up or configuring a **Study**. Also a **StudyReviewClient** will provide the front end for users to manage teams, and work with review tasks. The **AutosysServer** will be managing access control, concurrency control, and delegates to nested subsystems for the application logic. The **UserManagement** component is holding the responsibility for handling all CRUD operations regarding teams and individual users. The processing of all Study related CRUD operations are carried out by the **StudyManagement** component. The **StudyManagement** also manages Study Tasks, Study Criteria and Classifications, Study Phases and he other parts included in a study. The **PaperManagement** runs the filtering mechanisms on the research papers and does import/export of research papers according to the Study Criteria and Classifications. The **IStorage** interface is used for a Bridge Pattern to decouple the storage abstraction from the application logic so that the two can vary independently. At the bottom tier the **AutosysStorage** represents the subsystem for storing the user data, study data, and research papers.

2.2 Hardware/Software mapping

Systems
This program is inherently a documentation tool which map research articles goals and conclusions in a searchable context. This requires the program to be stable and reliable, especially in the context of data preservation.
The program is split in two components, a client and a external database server. The user will run a client on his computer which will communicate with the server which runs the application logic and storage. The user client will feature data preservation, to enable offline functionality, but only to a limited extend. This report primarily focus on the server, if any questions to the user client should occur, please refer to the according Blue SDD.

All user clients will contact the a single server. To solve this, the server will be multi threaded, which allows for multiple users interacting with the server at the same time, but At the current scope of the program, this is possible due to low amount of users, but should this system expand a new design should be conceived.



Program components
The server will be implemented in C Sharp as requested by the the client. By using C Sharp, we can utilise to Microsoft expansive database systems and interfaces especially in regards to the database. Additionally this also make the program easier to maintain, since C Sharp is a well known language and can utilise the .NET framework. As communication between the server and the clients, we plan to communicate with HTTPS request containing JSON object. This will make it easier to implement future changes and even completely replace the clients user interface with a more modern solution like a web page

2.3 Identifying and Storing Persistent Data

Identifying persistent objects
Autosys deals with three sets of objects that must be stored. The first set consists of research data that are accessed during conduction of secondary studies. The second set consists of objects that are created and accessed by the blue part of the system(eg. Users, roles, tasks). It need to be persistent to track the progress of the study, who is involved. The third set consists of data for a study configuration that are created by the study configuration UI.

The first set of objects is well defined and will rarely change during the lifetime of Autosys. These changes may occur whenever new research data has been created or removed and thus create a need for an update of the set. The second set of objects are managed and defined by the blue part of the system. Hence, we decide to let the blue part decide how to manage and access these persistent object through a generic interface. The third set of objects is also well defined and will not change once the study configuration has been made.

In this scope of the Autosys system, the main focus are set on the first and third set of persistent objects.

Selecting a storage strategy
By selecting and defining a persistent storage strategy enables us to deal with issues related to storage management. The main design goals of the yellow part of Autosys are to be reliable, scalable while also have high performance. It is therefore decided to implement a database management system because it allows concurrent queries and provide transaction mechanisms to ensure data integrity.

Compared to a flat file storage method, a database management system will also scale to large installations with many researchers that are to conduct studies simultaneously. However, to allow future upgrades or changes it has been decided that the storage strategy is not solely dependent on a database management system. Thus, the Istorage subsystem will provide an abstract interface that enables other kinds of storage to be coupled. The users of Autosys are not able to change the storage, however if an update is ever needed for the storage strategy, a system administrator are able to do so.

2.4 Boundaries

Configuration Use Cases
The handling of the previously described persistent objects is described in the use cases in the Requirement Analysis Document. By way of example, a ResearchManager can create, modify, remove Users and assign roles for them. However, the handling of the User objects have not yet been described in the use case model so far, as these objects have been refined during system design. A ResearchManager can always create new users and set up teams. Teams and roles are only assigned when a new study is set up in the Study Configuration UI. Hence, two additional use cases are invoked by the **ResearchManager**, CreateUser and SetupTeam.

CreateUser
The ResearchManager creates a user, gives it a name and configures the role of the user also determining which resources are accessible. Optionally, one may choose between selecting a persistent storage subsystem of a database or a bibtex file.

SetupTeam
The ResearchManager chooses a set of users as part of a team based on their name and role. This is then depicted in the Study Configuration UI where one can choose a given team predefined by the ResearchManager.

Startup and shutdown use cases
The start-up and shutdown of the StudyConfigurationServer is currently not described in the use case model. Hence, the two following use cases have been created.

StartStudyConfigurationServer
The **SystemAdministrator** start the StudyConfigurationServer. If the server was not shut down properly, this use case invokes a 'Check Data Integrity' use case, which validates the consistency and lack of corruption in last saved data. As soon as the initialization of the server is complete, ResearchManagers and Users may initiate any of their use cases.

ShutdownStudyConfigurationServer
The SystemAdministrator stops the StudyConfigurationServer. The server terminates any ongoing studies and stores any cached data from the blue client server side or from the setup of a new study in the StudyConfigurationUI. The connection between the clients and the server is shut down. Once this use case is completed, ResearchManagers and Users cannot access or modify a study.

Exception use cases

AutoSys can experience three major classes of system failures:

- Network failure: one or more connections between the Systematic Review Client and the Study Configuration Server are interrupted.
- Host failure: one or more clients are unexpectedly terminated.
- Server failure: StudyConfigurationServer is unexpectedly terminated.

The first two class of exceptions are handled in the AutoSysServer component. The component class will provide generic methods for clients to re-establish connection after a network failure or to restore the state of Studies after a crash. The handling of the exceptions depend on whether it happens real-time or just occur due to short interruptions/delays.
We handle network failures by notifying the client user of the network failure. We expect the client to retry later which the possibility of resuming from the last saved data in the backup. Consequently, we will design studies and their configuration in such a way that data is continuously saved to minimize data loss upon failures.
The server failure is handled by the use case for checking integrity of persistent data after an unexpected termination of the StudyConfigurationServer. This use case can be invoked automatically by the system upon start-up or manually by the SystemAdministrator.

CheckDataIntegrity AutoSys checks the integrity of the persistent data. This may include invoking tools provided by the database management system to re-index tables. Also, the system may check if the last logged studies and their configuration was saved to disk. The last saved data may be reused in the configuration of a new study.