Emergent Architectural Design

Willem Vaandrager 4175115 Elgar de Groot 4091108
Johnny Verhoeff 4137175 Hugo Reinbergen 4161173
Koos van der Linden 4133145

May 31, 2013
third draft

Abstract

The emergent architectural design is an important document in de design of software. Design elements such as how the system will be divided into subsystems, how these subsystems will be mapped to processes and computers, how these processes communicate with each other and what shared resources they will have and how the data will be stored (database design). These are all of major importance and will definitely changed throughout the design of the system.

Our system will be divided into three main subsystems, namely the model-layer, the view-layer and the controller-layer. There will be software running on the computer of the therapist, on the computer of the patient, and there will be some kind of server used to facilitate the communication between the former two. An SQL database will be used to handle the data.

Contents

1 Introduction **1**
1.1 Design goals 2

2 Subsystem decomposition **2**

3 Hardware/software mapping **2**

4 Persistent data mapping **3**

5 Concurrency **3**
Glossary3

1 Introduction

A good software product requires not only good code, it needs to run on the right hardware and that hardware can be spread among different platforms. This needs to be managed in a way that is maintainable and efficient. The data will be stored in a relational database for easy access and management the information about users and their files.

This document will describe the implementations of such systems for this project. It will start with evaluating the various sub-systems and how these are dependent of each other. Then, the way those sub-systems are mapped to computers, processes and how the communication between multiple systems is handled is covered. Following the database will be described, the design of the data and how it will be stored. The types of files used by the systems will also be specified. As last item, the concurrency between processes

1.1 Design goals

The goals behind designing these systems is to get a clear idea what is required to be made to get the eventual product to function properly. It has to be clear how much time will have to be spent on making systems such as databases and sub-systems so that the creation and implementation can be spread equally among the sprints. This document offers a form of assurance where there will be fewer surprises during the development process when it comes to accepting certain data structures and systems. This way, problems where finished work ends up being a waste of time due to changing systems half way through, will not occur as often, or hopefully, not at all. Finally, this architectural design presents another opportunity to show our plans for the development process to the product owner to create more transparency and options for communicating about the project.

2 Subsystem decomposition

In order to build the eCoach program, this task has to be divided in smaller sub problems. So the system is divided into subsystems. To increase the cohesion of the different subsystems, a strong division is needed. Another important issue is reuse, will parts of the program be reusable. To ensure the quality of the program the system also has to be easy to test. To achieve these goals, the choice has been made to design he system by use of a MVC-model (MVC). Cohesion is achieved with the strong layer-cohesion. The model-layer will be easy to reuse, as well as parts of the control-layer. And the system will be easy to test because the application can be tested separately from the user interface. One more reason for this approach is the need of two different representations of the data for the therapist and the patient: two different *views* on the same data.

So for the view-layer at least two different views will be created, one for the therapist and one for the patient. The patient view-layer will use an avatar as an eCoach and in this project the avatars display is realized by remote procedure calls to an engine called vizard. To ease the replacement of this engine, communication from the view subsystem with this system has to be done through an interface. This interface is a sub system of the view layer.

The controller-layer can be divided into the different tasks the system has to accomplish. A communication system to transfer data from the patient to the therapist and back is one of the subsystems. Another subsystem is the behaviour logic of the coach avatar. The eCoach has to behave as a real coach would do.

The model-layer consists of the database system to store all the information. For now, the model-layer is not sub divided. In section 4 a further description of this layer is given.

3 Hardware/software mapping

The different layers explained in the previous session are the divisions that can be made inside the code. But how is the system organized in term of processes and hardware?

For instance for hardware there will be at least three different computers/types involved:

- The computer of the therapist, used to display the progress of all patients and to communicate with them
- Numerous computers of patients, used to work with the eCoach and to communicate with the therapist
- Some kind of server used to facilitate the communication

You can also see this in figure 1. On the computer of the therapist and on the computer of a patient a database system will run to store all the permanent user data (more about this in paragraph 2.3). No decision has yet been made about what kind of database system is going to be used.

On the patient system a vizard process will run to display the avatars. Most of the view-layer (without the avatar displaying) and the control-layer is run from the same main process.

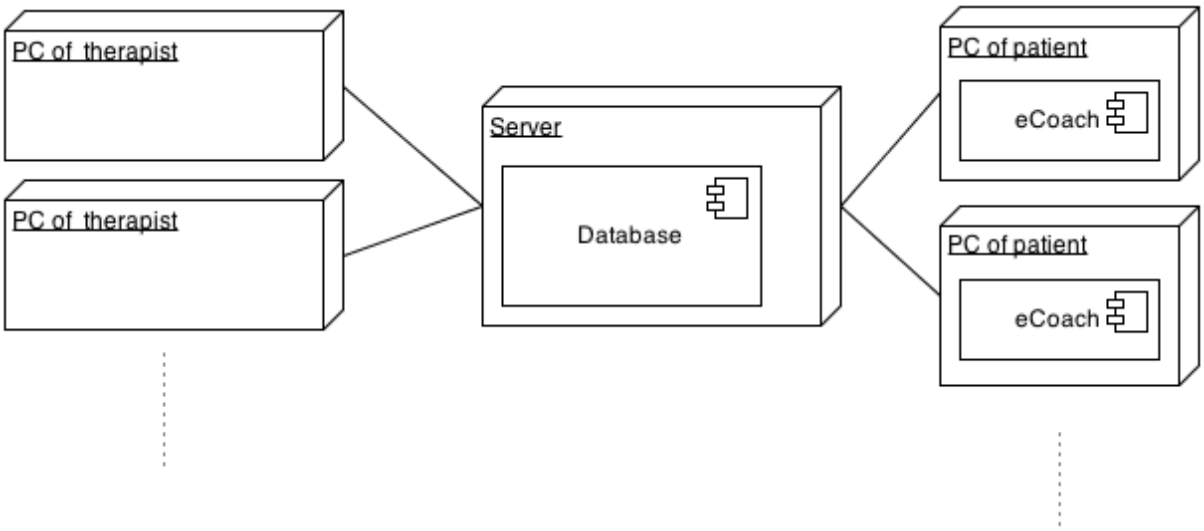

Figure 1: Deployment diagram of the system

Because the patient and the therapist do not share a computer and because the use of the program is very different for the patient and the therapist, separate programs are made for both users. This decreases the chance of security leaks and also decreases the amount of files to be installed localy.

4 Persistent data mapping

For this project we will have to use a database of some sort, because there is a lot of data that has to stored and accessed later on. There are many ways to store data for example a file system, but the team has chosen for a database because of the easy built-in functions offered by a database, such as rollbacks and faster searching with the use of indexes. This data may also have to be accessed at an other location for example, the patients progress has to be viewed by the therapist and the therapist needn't be at the same location. The team's preference for a database type is a SQL database that is "in the cloud" for when we deploy the system. When we're developing the system we'll use local databases on our laptops. We have chosen for this type because we all have experience with SQL. We have chosen for "in the cloud" because then it will be accessible everywhere and we have no maintenance for the server and we don't even have to buy a server or domain etc. In the database all the therapists and patients are stored, each therapist has a reference to which patient he/she has. Also all login information is stored for everyone (login names and passwords). This is done, so that the therapists and patients can login everywhere, for example to see an overview or their progress. The progress is very important for the therapist, because all exposure session are now digital so the patient and therapist don't see each other on a regular basis. The social events that the patient talks about with his/her eCoach will also be saved , so the avatar can 'remember' it and talk about it with the patient some time in the future.

Because the server or the patients internet connection can be offline, the patients progress and everything else that can be stored in the database, is first saved locally at the patients hard drive. And then it is synced with the server.

5 Concurrency

In order for the program to work as good as possible some processes have to run at the same time to assure faster and better use of the program. For better use and in case of a crash the database should be synchronized while the program is running. While the user is using the program and for instance talking to the avatar the system should be able to store info it has gained from the user and could also retrieve information from the database when the program has to display or use information. Also the program and the vizard should be threading. While the animation is shown the program should still run to interpret the user and should control the avatar.

Glossary

MVC A desing patern which seperates the data (model) from the representation (view) and the programlogic (controller). 3