ARCHITECTURE NOTEBOOK

LINKÖPINGS TEKNISKA HÖGSKOLA | IDA | TDDD09 | GRUPP 3

REVIEWED	Sebastian Wihlborg	2013-02-24
APPROVED	Josef Gustafsson	2013-02-25

TABLE OF CONTENTS

DOCUMENT HISTORY

VERSION	DATE	CHANGES	REVIEWED
1.0	2013-02-06	First version	Sebastian Wihlborg
2.0	2013-02-24	Completed the document	Sebastian Wihlborg

1. PURPOSE

This document describes the philosophy, decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

2. ARCHITECTURAL GOALS AND PHILOSOPHY

The main goal of the project is to replicate the original system and from there be able to expand and add new functions. Since the original system is on whiteboards and everything is done manually, there is a high priority on the system being very flexible. The original system uses several whiteboards and several people interact with each whiteboard. This means that this system must be able to handle several people on one screen at a time, as well as several touchscreens being connected to the server at the same time. The scalability of the system is a high priority.

The original system includes a large database of different patients, people and scenarios. This is something the new system must be able to handle and use. It must also be easy to enter new information into the database. Every relevant action that is performed during a game must be able to be logged and later reviewed in a user friendly way.

One of the major differences of going from whiteboards to touchscreens is the ability to remove something from the whiteboard, walk over to another whiteboard and place it there. This is something that needs to be handled in a different way on the touchscreens since it's something you can't do with them.

3. ASSUMPTIONS AND DEPENDENCIES

- We assume that several clients shall not be able to share a view.
- We assume that new views do not have to be able to be created during runtime.
- We assume that the server will have all the data necessary to start a scenario.

4. ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

4.1 System shall:

- It must be possible to switch between different views in order to make the system executable with few screens.
- Manage multi touch and parallel work.
    - Interaction with multiple fingers on multiple items simultaneously on the same screen
- Operate against a database
    - Contains all patients with its information
        - Visible information, which is written on the front.
        - Information from examination, ABCDE.
        - Patient category, what measures are demanded within what time.
        - Graphics.
    - Information about scenarios, what patients, or what types of patients - patient sets, that will be included.
- Be able to run with the clients and screens distributed across several rooms.
- Be able to run on several clients over a network
    - One client controls one screen

4.2 System should:

- Be able to run with the clients and screens distributed over several geographically separated locations.
- Contain a database table for the current scenario, which contains information about where all patients, resources and vehicles are. It will also reflect which actions have been made. This is actually the log - the question is whether it should be in database form or be handled internally by the program.

5. DECISIONS, CONSTRAINTS, AND JUSTIFICATIONS

- We have decided that Actor class will have 4 subclasses, patient, vehicle, cover and worker. We have done this because currently we see no actor type that doesn't fit into one of these subclasses. This decision means that it becomes a lot easier to manage the database since every actor can be put into one of four tables.

- We have decided that the client will work with a local copy of a model and notify the server of any significant changes so that it can change the "real" model. We have decided to do it this way since it means that we do not have to worry about any lag between input and output. It also means that even if you lose connection you can still work on your local model.

We have decided to use MVC (*Model-View-Controller*) to handle the interaction between the user and the model. This made it easy for us to split the team into a front-end and a back-end group and we felt that the pattern worked very well with how we envisioned the architecture of the system.

6. ARCHITECTURAL MECHANISMS

6.1 Create scenario

This mechanism is run by the server and extracts a scenario from the database and creates the data structures necessary to run the scenario.

6.2 Create local copy of a model

This mechanism is initiated by the client. It creates a copy of the model held by the server for the client to work against.

6.3 Update the model

This mechanism is initiated by the client when something is significantly changed in the local model. It updates the model held by the server.

6.4 The interface between view and model

This mechanism is what handles the communication between the view and the model.

6.5 The log

This mechanism is what keeps track of all the significant things that have been done in the scenario.

7. KEY ABSTRACTIONS

Actor: Everything that can be moved and interacted with on the board, such as patients, ambulances or covers. Does not include the menus.
Tags: The items that you can attach to actors, such as priority markers and medical treatment markers.
Log: Where every relevant action preformed during the game is stored.
Scenario: The starting point of a game. Contains all the information needed to start a game.
View: What can be seen on the screen excluding user interface elements that are the same in every view. It is comparable to a whiteboard in the original system.
Model: The non-visual representation of a view. Contains all information related to what is on the view.
Type: Defines what kind of actor an actor is: Patient, vehicle, cover or worker.

8. LAYERS OR ARCHITECTURAL FRAMEWORK

The system revolves around the server. At the start of an exercise the server creates a set of models taken from the database. Then when the clients connect to the server they can create a local copy of a model to work with. When something significant is changed in the local model, the server is notified and changes the “real” model. The log is also notified of this. The client uses MVC to handle the interaction between the user and the model.

9. ARCHITECTURAL VIEWS

9.1 Logical view

The logical view describes the structure and behavior of architecturally significant portions of the system. It displays the relationship between the major architectural parts.

9.2 Deployment view

The deployment view illustrates the system from a programmer’s perspective. It contains classes, attributes, data types and the like.

9.3 Implementation view

The implementation view shows the physical parts of the systems and the physical connection between these components.