A Patient Post-operative Function Survey System for the Tablet-PC

Jing Yao
Abstract

A Patient Post-operative Function Survey System for the Tablet-PC

Jing Yao

This project aims to develop a system for evaluating patient satisfaction with foot surgery. It is in collaboration with the Uppsala Academic hospital, the main user of it will be the patients and researchers. That means a smooth running system is not enough. We need to consider the human-computer interaction factors associated with helping patients to express their situation as easily and accurately as possible, and help the researchers to obtain intuitive and meaningful results and the main goal of this project is to develop a survey system on the tablet PC which uses the touch screen to implement survey instruments that have previously been impossible to deliver in a computerized manner. The system provides a way for patients to give their feedback to their doctor easily, and also for researchers to measure, keep and compare results. We will use a slide tool bar in the GUI to help the patient to express their status accurately and intuitively. Feedback to researchers includes numerical representations of patient feedback measure the effects as well. The survey results will be stored in a Database system, compare to the traditional way, it will perform better in results management and querying.

This report describes how we design and develop this survey system, and how we make it as usable as possible. The project is based on Windows. The main program and GUI for the patient is developed in Java. The Database for storing the result is based on the MySQL system. The interface for the researchers to query and compare the results is developed in J2EE. The project was divided into several development phases (called "sprints" in the Agile development terminology). Each sprint lasted 1-3 weeks (depending on the subtask to be completed). Each sprint focuses on a specific use case, and the documentation is also included in the sprint.
Contents
1. Introduction ............................................................................................................. 3
   1.1 Background ................................................................................................... 3
   1.2 Tools ............................................................................................................. 5
   1.3 Planning ........................................................................................................ 5
2. Pre-analysis ............................................................................................................ 6
   2.1 Technology .................................................................................................. 6
      2.1.1 Java .................................................................................................... 6
      2.1.2 MySQL ............................................................................................... 6
3. Analysis .................................................................................................................. 8
   3.1 Requirement ................................................................................................. 8
   3.2 Use case ........................................................................................................ 8
   3.3 Service .......................................................................................................... 10
4. Design & Architecture ......................................................................................... 12
   4.1 Survey front-end ......................................................................................... 12
   4.2 Administration front-end ........................................................................... 13
   4.3 Database Design ......................................................................................... 16
5. Implementation .................................................................................................... 17
6. Conclusion & Future work .................................................................................... 20
Reference ..................................................................................................................... 21
Appendix ...................................................................................................................... 22
   Interface module .................................................................................................. 22
   Data module ......................................................................................................... 23
   Controller module ............................................................................................... 26
1. Introduction

This report describes how we design and implement the survey system, what problems we suffered and how we solved them. The report consists of the following parts. First we introduce the project background and our plan in chapter 1, and then we will make a preliminary analysis of the project requirements, including the technologies which are used in the project in chapter 2. We also analyze the project system requirements, use case and services we should provide in chapter 3. In chapter 4, we present and discuss the system chapter 4 design system architecture. Chapter 5 describes the implementation. The final chapter presents the conclusion and makes some recommendations for future work.

1.1 Background

Survey methods have been widely used in medical research in different ways for a long time, and the number of methods for survey increased as the time went on. The simplest way to survey patient satisfaction with treatments of operations might be to just ask the patient “How do feel today?” or “Are you feeling better?”. However, there are a number of other methods including using a paper questionnaire. In many studies researchers distribute questionnaires to the patients, collect them back and analyze the data gathered from the survey. Patient function surveys include a number of scales developed for a pen and paper survey environment. These scales are difficult to implement in traditional computer based survey systems. For instance, the figure below shows a sliding ruler scale which is usually used in an oral survey.
As we can see from the picture, one side of the ruler is marked with numbers as degree, and the other side is marked with emotions. When doing the survey, the patient does not need to answer doctor’s question any more, for a question like “How do you feel today?”, what the patient should do is just point out which degree they are on the emotion side, the doctor will be clear that which degree the patient is on the degree side using a numeric scale.

But what we should notice is that all those methods are not perfect. The oral survey is simple, but hard to measure. The patient can just answer the stuff “I feel much better now.”, but it is hard to know how much better he is. The unclear answer may mislead the doctor. The questionnaire and sliding ruler perform better in this field, but here another problem emerges. The doctor will always need a place to store the latest survey results, and there is a significant overhead associated with the time needed to sort, compare and analysis them. Comparisons with previous results are also extremely difficult. In addition to these activities, the doctors or researchers in the hospital also need to record patient feedback and personal details in a secure manner.

The project is in collaboration with the Uppsala Academic hospital, the main user of it will be the patients and researchers. The project aims to develop a system for evaluating patient satisfaction with foot surgery. That means a smooth running system is not enough. We need to consider the human-computer interaction factors to help the patients to express their situation as intuitively and accurately as possible, and help the researchers to get a good overview of the results and to be able compare them easily. The main goal of this project is to develop a survey system on
the tablet PC which uses the touch screen to mimic techniques used in traditional oral and written survey methods. The system will provide a way for the patients to give their feedback to their doctor easily, and liberate the doctors from the recording, keeping, sorting and checking work associated with paper surveys. We will use a sliding tool bar which is similar to the sliding ruler in the GUI to help the patient to express their statues accurately and easily, and that will feed back a number to the researchers to measure the effects as well. The survey results will be stored in a Database system, comparison to the traditional way, a Database will perform better in terms of results storage, management and querying. [1][2][3]

1.2 Tools

- IDE
  - Netbeans 6.5
- MySQL 5.1
- Java Development Kit 6 (JDK6)

1.3 Planning

The project commenced on 2009/03/27 and was finished at 2009/09/07, we hope the beta version will be released on first of July 2009. The following is the detailed time schedule:

- 2 weeks background reading
  - familiarity with surveys
  - familiarity with Tablet-PC
- 2-3 weeks test programming
  - develop simple Java applications for Tablet-PC & test functionality
  - access viability of slider scale
  - Implementation prototype
  - draft Database structure/schema
- 8-10 weeks implementation of survey instrument & Database programming
  - patient survey application development
  - Database implementation & connection to survey application
  - results analysis/management system implementation
- 5-7 weeks testing and report writing

Total 17-22 weeks
2. Pre-analysis
From the previous chapter, we can get an outline about the system. In this chapter, we describe the technologies which are used in implementation and the problems we encountered.

2.1 Technology

2.1.1 Java
Java is a programming language originally developed by James Gosling at Sun Microsystems and released in 1995 as a core component of Sun Microsystems’ Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. That means that the same Java program can execute on different platforms, what we need to do is to just provide different Java Virtual Machines (JVM) for different platforms. For instance, the same Java application can be run on both Windows and Linux platforms. So the Java language is more suited for internet applications and enterprise network and applications. The Java platform includes JVM and Java APIs for different platforms.

The original and reference implementation Java compilers, virtual machines, and class libraries were developed by Sun from 1995. As of May 2007, in compliance with the specifications of the Java Community Process, Sun made available most of their Java technologies as free software under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java and GNU Classpath. [4]

2.1.1.1 Swing
Swing is a widget toolkit for Java. It is part of Sun Microsystems' Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. [5]

2.1.2 MySQL
MySQL is a relational Database management system (RDBMS) which has more than 6 million installations. MySQL stands for "My Structured Query Language". The program runs as a server providing multi-user access to a number of Databases.

The project's source code is available under terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL is owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now
a subsidiary of Sun Microsystems, which holds the copyright to most of the codebase.

MySQL is commonly used by free software projects which require a full-featured Database management system, such as WordPress, phpBB and other software built on the LAMP software stack. It is also used in very high-scale World Wide Web products including Google and Facebook. [6]
3. Analysis
We have some hard requirements from Uppsala Academic hospital when the project starts up. We divided those requirements into two parts, functional requirement and non-functional requirement.

3.1 Requirement

3.1.1 Functional requirement
- The system should be able to emulate usual survey methods
  o Support choice question like the paper questionnaire
  o Support drawing question like the VAS (sliding ruler) pain scale
- The system should have a management system to manage related data
  o Users of the system
  o Method types
  o Problem types
  o Question Database
  o Questionnaire Database
- The system should provide result and patient information data analysis functions for the users supporting
  o Searching for specified data
  o Comparison of data
  o Exporting certain survey result data in .xls format

3.1.2 Non-functional requirement
- The system is implemented in the Windows environment
- The system is designed for use on the Tablet-PC with touch screen
- The System should also support PC’s without touch screen

3.2 Use case

Use cases describe the system behaviors which originate from outside the system. According to our system requirements, we divided the use case into two parts, interactions for management and survey front-ends.

The Management front-end manages with user data, system properties, and data analysis.

- Log in: it is the security entrance for users to access into management system.
- Log out: it is the security exit for the management system.
- User management: to manage the users of the system.
- Data management: it should include multiple functions to manage related data such as method type, problem type, etc.
• Survey material management: it is used to manage related materials of survey, including questions, questionnaires.
• Data analysis: the users can do the data analysis work and export the data into an .xls file.

Figure 3.1 Management GUI use case

The survey font-end is the main view of the system, and is designed to interact with both researchers and patients.

• Questionnaire browser: the users can check questionnaire details and choose questions to compose a survey.
• Patient information generator: collects related patient information for data analysis.
• Survey executor: executes the survey using the questions selected by the user.
3.3 Service

From the system requirements and use cases, we can summarize the services which should be provided by our system as follows:

- Log in service: user information verification
- Log out service: log out current user
- User data management service: to manage system users
- Data management Service: to manage related data including method type and problem type in system
- System material service: to manage survey related materials including question and questionnaire in system
- Data analysis service: supports data search, comparison, comment, and export function for users
- Record service: store information which need to be kept into Database
• Survey service: read related information of every question in the questionnaire which is picked out by the user and generate question paper according to different type of questions. Record patient’s answer, calculate sub-score and total score of this patient.
4. Design & Architecture
In this chapter, we describe the system architecture and its implementation. The system is into three parts, the Survey front-end, Administration front-end and database.

![Figure 4.1 Structure of the Survey System](image)

4.1 Survey front-end

The Survey front-end is responsible for generating patients’ information, results of the survey questions and storing the latest results into database. The model-view-controller (MVC) design pattern is used in our system, because it helps the program to be more extensible and readable than programs that do not use MVC. In this design pattern, there are three layers:

- **Model layer**: consist of object that represent the data in the application, for example represent user’s name, patient’s name, problem of the patient, method for the patient, etc.

- **View layer**: it presents format and appearance of the system user interface, for example application windows, views, buttons and so on, in addition, the view objects need to have notification of response to some events.

- **Controller layer**: provides a bridge between model and view layer, it receives the notification from view layer, the action depends on the data from the model layer.
The structure of the Survey front-end is not so complicated, as we can see from figure 3.1, it is divided into three parts.

**Patient Information Generator:** is responsible for picking a questionnaire from our data base and generating patient information including name, person number, problem, and method type of the patient.

**Main Survey Application:** is the main part of Survey front-end. It generates different type of questionnaires according to different type of questions, and records the answer from patient.

**Database Connector:** is the bridge between our application and Database, and responsible for storing gathered patient information, survey result, and the other related information into Database.

### 4.2 Administration front-end

The Administration front-end is a bit more complicated than the Survey front-end. We also applied the MVC design pattern in this part of system. The high level structure can be seen as figure 4.2.

![Figure 4.2 Architecture of the Administration front-end](image-url)
The Log in Verifier module is used to verify whether the user’s user name and password matches or not. If the result is yes, the system will log the user into administration front-end and allow him to manage part or the whole system according to his authority level. In our system, there are two kinds of authorities: normal user and super user. The normal user can only manage the content which is related to that particular user. For instance, in Result Analysis module, a normal user can only see and operate the result records which are from surveys created and managed by that user. On the other hand, a normal user will not be allowed to do any operation to the other users. The super user plays as a super manager in the system. A super user can see and operate everything in the system, not only the content related to the user’s self, but also the ones related to the other users. And a super user will be able to add, remove users from the system or even reset the password for existing users.

The Administrator Management module is more like a module allows a super user to add or remove users in the system or reset the password for existing users in this module. A normal user will not be able to do anything in this module except check for existing users.

Question Database Management is one of the key modules in our system. The users (doctors) will need to add their questions to a database and specify the details of each question. The details includ the question body of the question, choices for the question if it is a choice question, the type of the question (is it a single choice question, a multiple choice question, or a drawing question.), the scores for each choice if it is a choice question or the total score for the question if it is a drawing question, etc. Existing questions can also be modified or removed from Database using this module. The details of the question in the system can be seen in the following table.

<table>
<thead>
<tr>
<th>Details of a question</th>
<th>Question body</th>
<th>Question type</th>
<th>Choices for question</th>
<th>Score for every choice</th>
<th>Score for the question</th>
</tr>
</thead>
<tbody>
<tr>
<td>The question body for every question</td>
<td>Simple choice, Multiple choice or Drawing question</td>
<td>Only available for simple choice or multiple choice questions</td>
<td>Only available for simple choice or multiple choice questions</td>
<td>Only for drawing questions</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1 Details of a question in the system

The Questionnaire Management module is another key module in our system. The users will be able to pick existing questions up from Database and form a questionnaire for survey. While doing that, they will need to specify the classification for the questionnaire according to the problem types. As the same as the Question
Database Management, the existing questionnaires can also be modified or removed from Database.

Compared to the Question Database Management and Questionnaire Management modules, the Problem Type Management and Method Type Management modules are a bit simpler. Those two modules provide standard to classify questionnaires and patient. Our system uses method type to classify patients and problem type to classify questionnaires.

The data analysis part in our system consists of two modules, Result Analysis and Patient Information Checker. As we can see from their name, Result Analysis is used for analyzing the survey results which are recorded by our system. According to the demand from Uppsala Academic hospital, we offered several alternatives for analyzing the results.

![Figure 4.3 Functions of result analysis](image)

Main functions offered by the Result Analysis modules are shown in figure 4.3.

The key point of Patient Information Checker is search and sort tool embedded in our system. The search function we offered supports fuzzy query and multiple conditions query. By using this tool, the users can extract system Database on questions like “How many male persons who are more than 50 years old and have Jack in their name”. The search function which is provided by the Result Analysis module is also implemented by this tool.

The Database Connector plays the same role as it does in the survey front-end. The services which are offered by this module include query and updating Database.
4.3 Database Design

The data in our Database also present as objects in data model in MVC design pattern. The follow figure 4.4 shows the full Database diagram of the system.

Figure 4.4 Design of Database
5. Implementation

This chapter describes the system implementation. To implement the MVC pattern, we need at least three parts. They are the user interface module, the data module, and the controller module.

The interface module defines a new window. All components which are used in our project are extended on this window (like what is shown by figure 5.1).

![Figure 5.1 different modules based on a same window](image)

The following pictures show some modules of the administration front-end we implemented. To enhance the intuitive performance of our system, we rewrote the `JTable` component which is provided by Swing.

![Welcome to management system, please choose the following to manage:](image)
We implemented two types of questions in our system. They are choice question and drawing question. The drawing question is a simulator for the sliding ruler which is showed in figure 1. The drawing question provides a solution for the researchers to ‘measure’ the situation of the patients, and also, it is a intuitive way for the patients to present their state to the doctors and researchers.
Figure 5.5 two types of questions
6. Conclusion & Future work

The project taught us a lot during the project duration. What we learned is not only the coding skill, but also some other skills which are useful for software development. We learned how to communicate with our coordinators, how to develop software for people who has some special demands, since the program we developed will face to the patients in Uppsala Academic hospital.

The project we did is surely not perfect. There are some future works which can be completed.

- The project can be extended to support more types of questions.
- The data analysis part can be developed with more functions.
- The program can be updated into a remote application. This will make it be more convenient for the patients who are resting in their home.
Reference


Appendix

Interface module

Definition of the main window

```java
public static JFrame mainWindow = new JWindow();

public static GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();

public static GraphicsDevice gd = ge.getDefaultScreenDevice();

try{
    UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
}
```

```java
catch(Exception e){}

gd.setFullScreenWindow(Main.mainWindow);

mainWindow.setLayout(new BorderLayout());

mainWindow.setVisible(true);
```

To add components on this window, we can just extend it as following.

```java
Back_Front.Main.mainWindow.remove(oldPane);

Back_Front.Main.mainWindow.invalidate();

......

backPane = new JPanel();

backPane.setLayout(new BorderLayout());

backPane.setBorder(BorderFactory.createEmptyBorder(200, 200, 200, 200));

backPane.add(questionBodyTxt,BorderLayout.NORTH);

backPane.add(choicePane,BorderLayout.CENTER);
```
for different data, we defined several data modules in our system. For instance, for the survey related information and questionnaire which is chosen for survey, we defined its data module as following

```java
private String typeOfProblem;
public void setTypeOfProblem(String problem){
    typeOfProblem = problem;
}
public String getTypeOfProblem(){
    return typeOfProblem;
}
private String typeOfOperation;
public void setTypeOfOperation(String operation){
    typeOfOperation = operation;
}
public String getTypeOfOperation(){
    return typeOfOperation;
}
private int surveyTaker;
public void setSurveyTaker(int takerID){
    surveyTaker = takerID;
}
```
public int getSurveyTaker()
{
    return surveyTaker;
}

private int surveyHolder;
public void setSurveyHolder(int holderID)
{
    surveyHolder = holderID;
}

public int getSurveyHolder()
{
    return surveyHolder;
}

private String questionnaireName;
public void setQuestionaireName(String name)
{
    questionnaireName = name;
}

public String getQuestionaireName()
{
    return questionnaireName;
}

private int classificationID;
public void setClassificationID(int clsf)
{
    classificationID = clsf;
}

public int getClassificationID()
{
    return classificationID;
}

private int[] queID;
public void setQuelDElement(int i,int qID)
```java
queID[i] = qID;
}
public int getQueIDElement(int i){
    return queID[i];
}
public int getQueIDLength(){
    return queID.length;
}
private int numberOfQuestion;
public void setNumberOfQuestion(int num){
    numberOfQuestion = num;
}
public int getNumberOfQuestion(){
    return numberOfQuestion;
}
private int questionnaireID;
public void setQuestionaireID(int id){
    questionnaireID = id;
}
public int getQuestionnaireID(){
    return questionnaireID;
}
private int[] score;
public void setScoreElement(int i,int tmpScore){
    score[i] = tmpScore;
}
```
public int getScoreElement(int i){
    return score[i];
}

Controller module

The follow code shows how to set the data which is collected from choice question interface into data module.

    int questionIndex = GeneratePatientDate.sv.getQuestionIndex();
    for (int i=0;i<choiceCollection.length;i++){
        if (choiceRB[i].isSelected()){
            questionnaireView.init.setScoreElement(questionIndex, Integer.valueOf(scoreCollection[i]));
            GeneratePatientDate.sv.setQuestionIndex(questionIndex+1);
            GeneratePatientDate.sv.go(backPane);
        }
    }

And the following shows how the Database connector writes into or updates our Database.

    public GetConnection(){
        try {
            Class.forName( "com.mysql.jdbc.Driver" );
            connection = DriverManager.getConnection(url, username, password );
            statement = connection.createStatement();
        } catch(Exception e){
            System.out.print(e);
        }
    }
To meet the need of our project, we rewrote some of the modules which are provided by Java as our controller module. The following is the JTable component which is rewritten by us.

```java
public class MyTableModel extends AbstractTableModel{
    private Vector columnNames = new Vector();
    private Vector data = new Vector();

    public MyTableModel(Vector Rows, Vector columns){
        columnNames = columns;
        data = Rows;
    }

    public int getColumnCount() {
        return columnNames.size();
    }
}
```
return columnNames.size();
}

public int getRowCount() {
    return data.size();
}

public String getColumnName(int col) {
    return columnNames.get(col).toString();
}

public Object getValueAt(int row, int col) {
    Vector tmprow = (Vector)data.elementAt(row);
    return tmprow.elementAt(col);
}

public Class getColumnClass(int c) {
    return getValueAt(0, c).getClass();
}

public boolean isCellEditable(int row, int col) {
    if (getValueAt(row, col).toString().equals("true") || getValueAt(row, col).toString().equals("false"))
        return true;
    else
        return false;
}
public void setValueAt(Object value, int row, int col) {
    Vector tmprow = (Vector)data.get(row);
    tmprow.set(col, value);
    data.set(row, tmprow);
}