Software Design Document

TheraWii

Tim Chagnon
Joe Kokinda
Andrew Meinert
Don Naegely

April 7, 2009
Revision 7

SRS Revisions

Date	Description	Revision	Editor
2/1/2009	Created the document	0	Tim Chagnon
2/1/2009	Added Profile & UI UML	1	Andrew Meinert
2/2/2009	Added Storage UML & Text	2	Joe Kokinda
2/2/2009	Added Architecture Diagram	3	Tim Chagnon
2/3/2009	Added wiiuse, Therapy UML	4	Tim Chagnon
2/3/2009	Added Controller, State, Game	5	Don Nagely
2/3/2009	Data Storage, Model Methods	6	Joe Kokinda
2/3/2009	Formatting	7	Tim Chagnon
4/6/2009	GUI	8	Andrew Meinert
4/6/2009	Task Editing	9	Tim Chagnon
4/6/2009	Sequence Diagram	10	Joe Kokinda

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe the implementation of the TheraWii Software described in the TheraWii Business Requirments. The TheraWii Software is designed to create and perform physical therapy activities.

1.2 Scope

This document describes the implementation details of the TheraWii Software. The software will consist of a two major functions. First to design therapies that are made up of tasks, and the second to perform the therapies. This document will not specify any actual therapies or the testing of the software.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Physical Therapy

Posture The orientation of any body segment relative to the gravitational vector. It is an angular measure from the vertical [1].

Balance The dynamics of body posture that prevents falling. It is related to the inertial forces acting on the body and the inertial characteristics of body segments [1].

Center of Mass (COM) A specific point at which the system's mass behaves as if it were concentrated [1].

Center of Pressure (COP) The point location of the vertical ground reaction force vector. It represents a weighted average of all the pressures over the surface of the area that is in contact with the ground. It is also called the Center of Balance (COB) [1].

1.3.2 Nintendo Wii

Wii Remote Device that communicates through Bluetooth wireless protocol to the Nintendo Wii Gaming System. Data communicated includes button press and releases, accelerometer readings, and an Infrared (IR) LED pointing system.

Wii Balance Board Device that communicates the COP through Bluetooth wireless protocol to the Nintendo Wii Gaming System.

1.3.3 Software

Therapy A series of tasks that is completed in one session.

Session A given time in which a user completes a therapy.

Task A subunit of a therapy that has an objective with success and fail criteria.

2 Design Overview

2.1 Description of Problem

Traditional physical therapy techniques are often limited to activities that provide little to no statical feedback and technologies that provide this type of feedback are normally expensive. The TheraWii Software will use Nintendo Wii technology to provide physical therapists with data in a economical and efficient approach.

2.2 Technologies Used

The TheraWii Software will communicate with input devices designed for the Nintendo Wii. The required devices include the Wii Balance Board, the Wii Remote, and the Wii Nunchuck. These devices will communicate with the software through Bluetooth wireless protocol.

The target platform will be Microsoft Windows, and the development environment is Microsoft Visual Studio 2008. The therapies will be implemented in the Microsoft XNA game development framework.

2.3 System Architecture

Figure 1 depicts the high-level system architecture. The system will be constructed from multiple distinct components:

- **Therapist Interface** — The windowed interface for constructing and editing Therapies, and viewing or exporting profiles.
- **Game Interface** — The simple game-like environment for executing Therapies and collecting data.
- **Wii Library** — The interface for maintaining Bluetooth connections to Wii input devices and generating input events.
- **Data Model** — The classes needed to organize Therapies, Tasks, Profiles, Sessions, etc.
- **Data Storage** — The interface for storing, importing and exporting the data model and raw collected data.

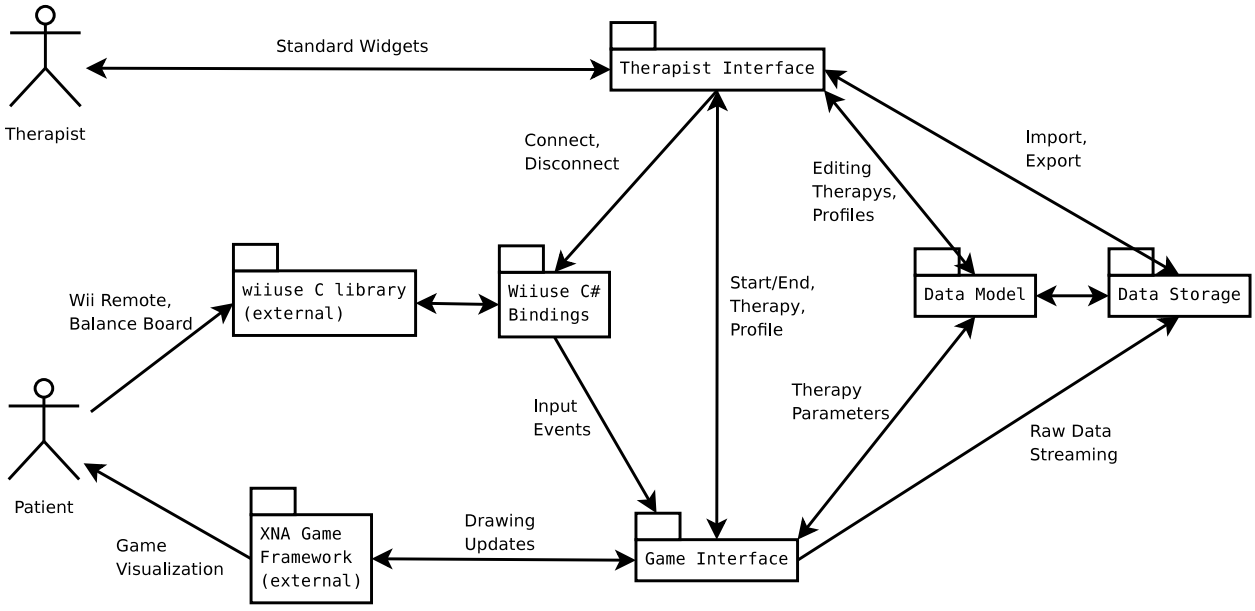

Figure 1: TheraWii Architecture

2.4 System Operation

Figure 2 is the typical sequence of events that occur during a TheraWii session.

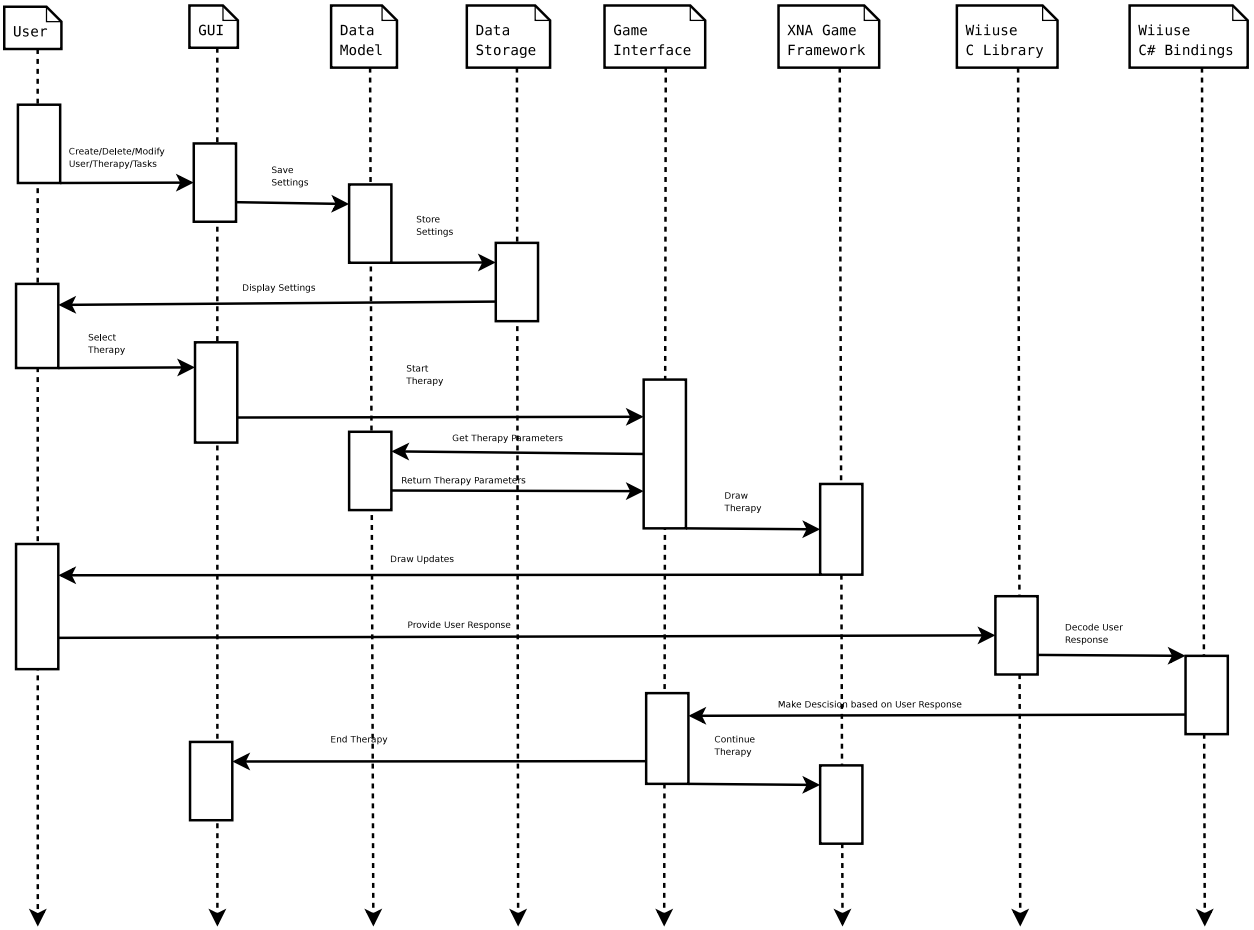

Figure 2: TheraWii Sequence Diagram

3 Requirements Traceability

Requirement	Description	Design Reference
[r1.*]	Inputs	§6
[r3.*]	Menus	§4, Fig. 3
[r4.*]	Therapy Editing	§4, Fig. 3
[r5.*]	Task Editing	§4, Fig. 4
[r6.*]	Profiles	§4, Fig. 5
[r7.*]	Exporting	§7
[r8.*]	Session Details	§4, Fig. 5
[r9.*]	Therapy Execution	§5

4 Therapist Interface

Figure 3 depicts the UML model for the Therapist Interface. Figure 4 depicts the UML model for each of the task editing forms. Figure 5 depicts the UML model for the Profile interface. Most of the methods in these class diagrams represent callback functions for user input events.

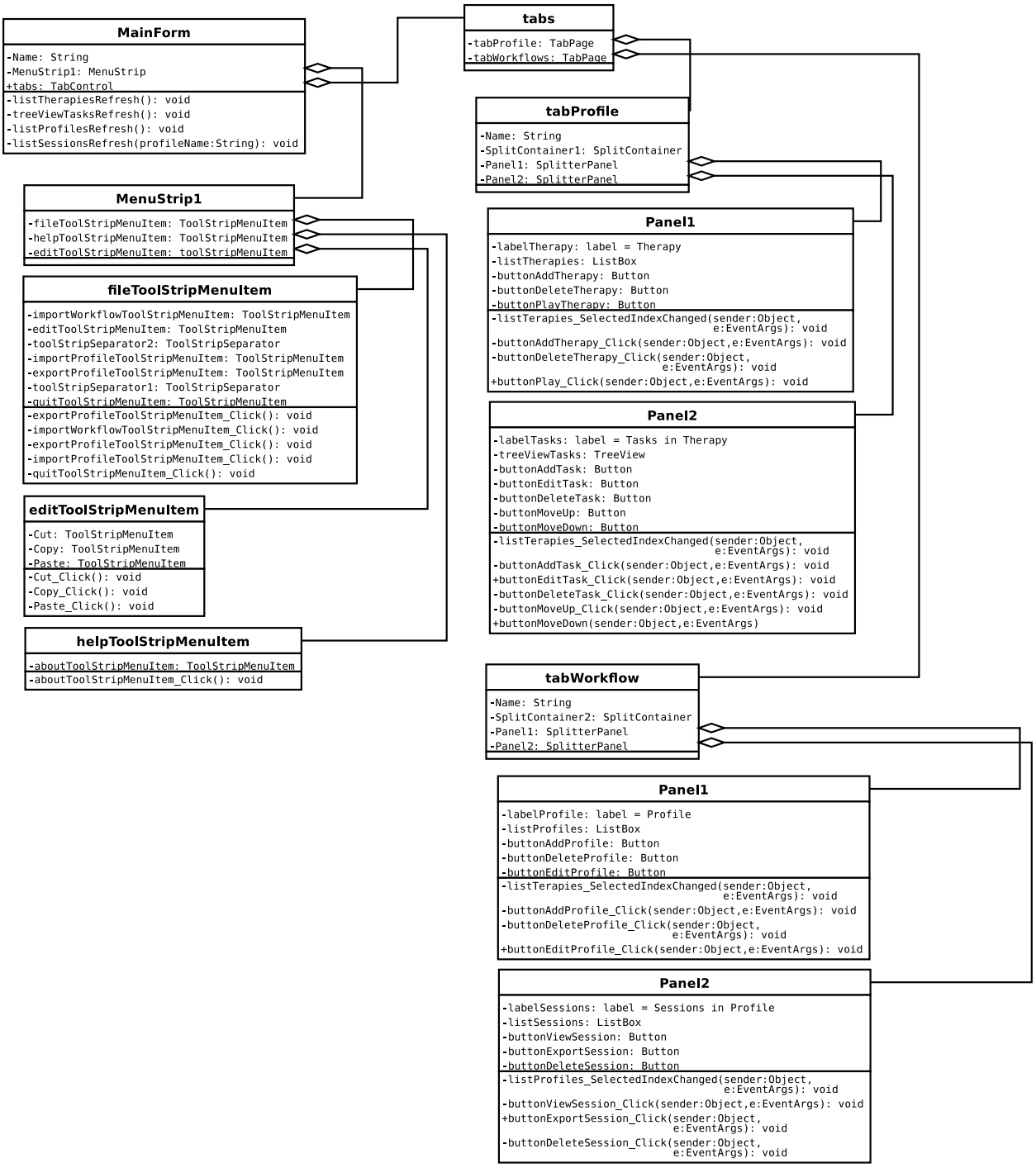

Figure 3: Main Therapist Interface

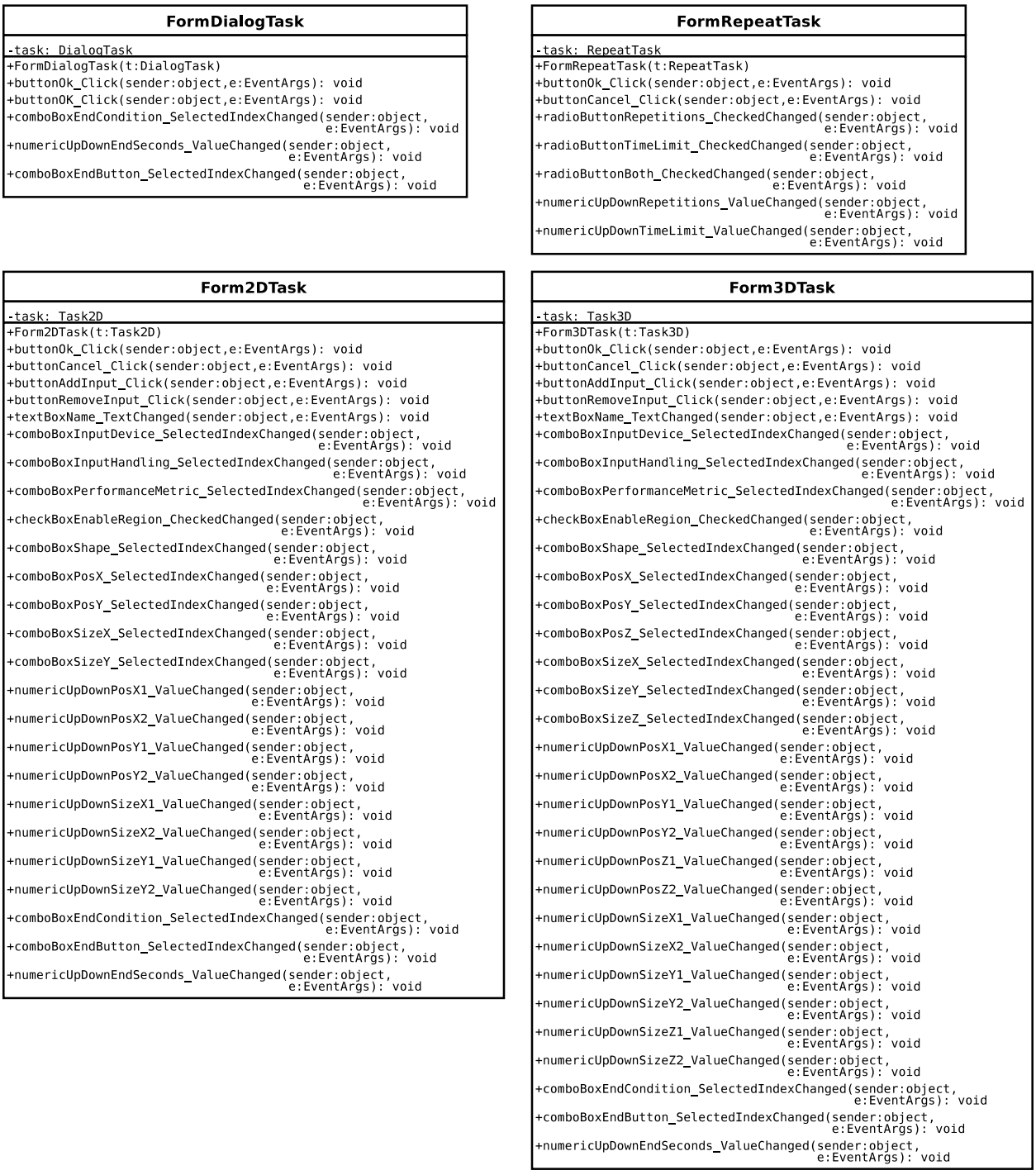

Figure 4: Task Editing Forms

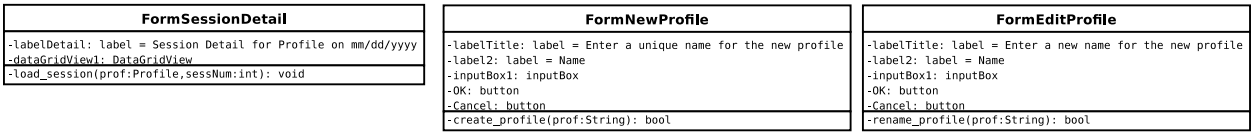

Figure 5: Profile Forms

5 Game Interface

Figure 6 depicts the UML model for the Game class.

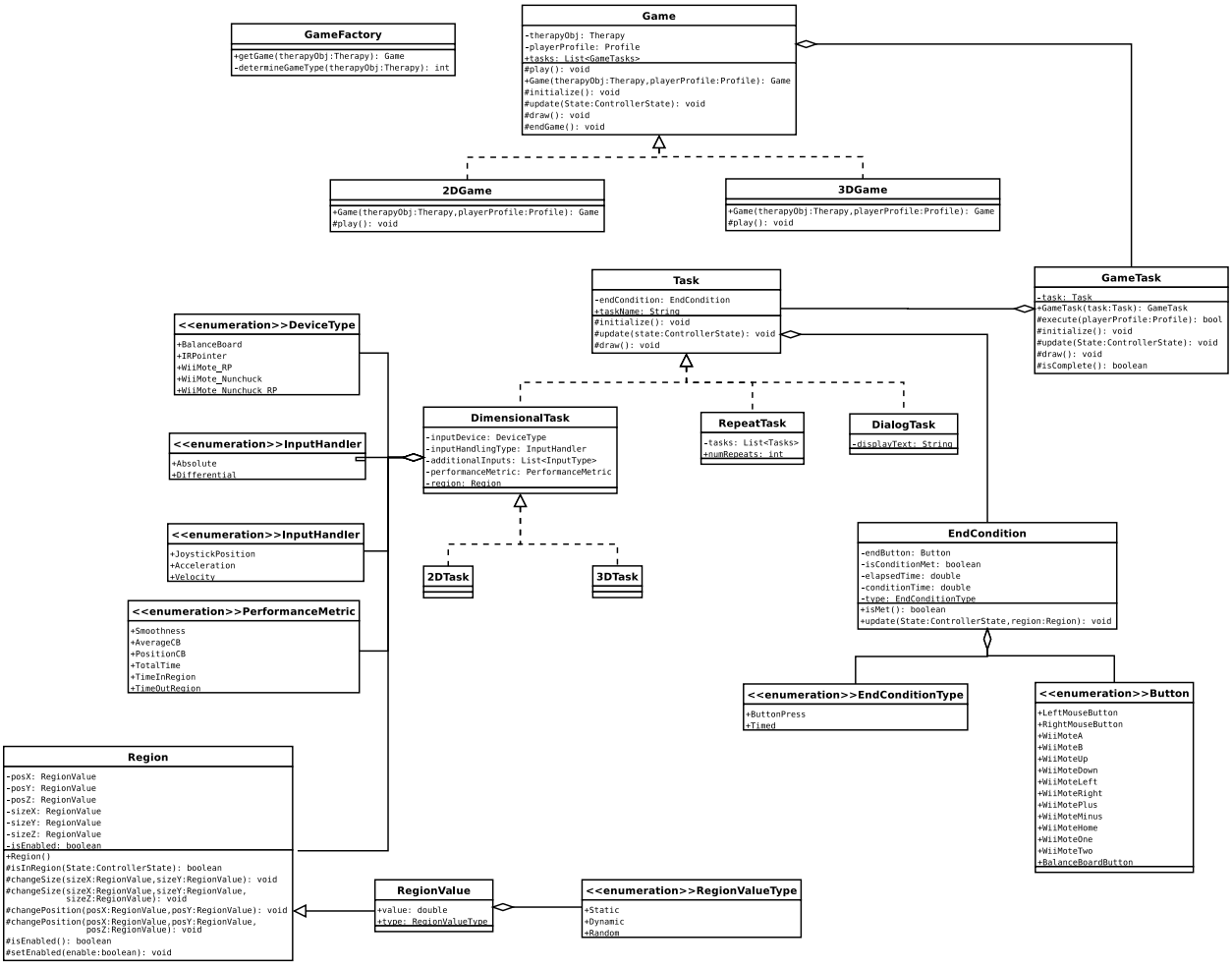

Figure 6: Game UML

6 Input Devices Subsystem

Figure 7 depicts the UML model for the WiiController Interface.

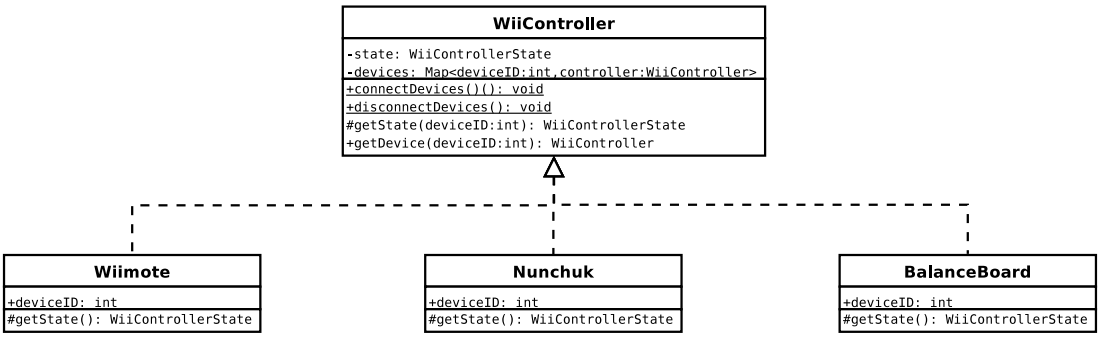

Figure 7: WiiController UML

Figure 8 depicts the UML model for the WiiControllerState Interface.

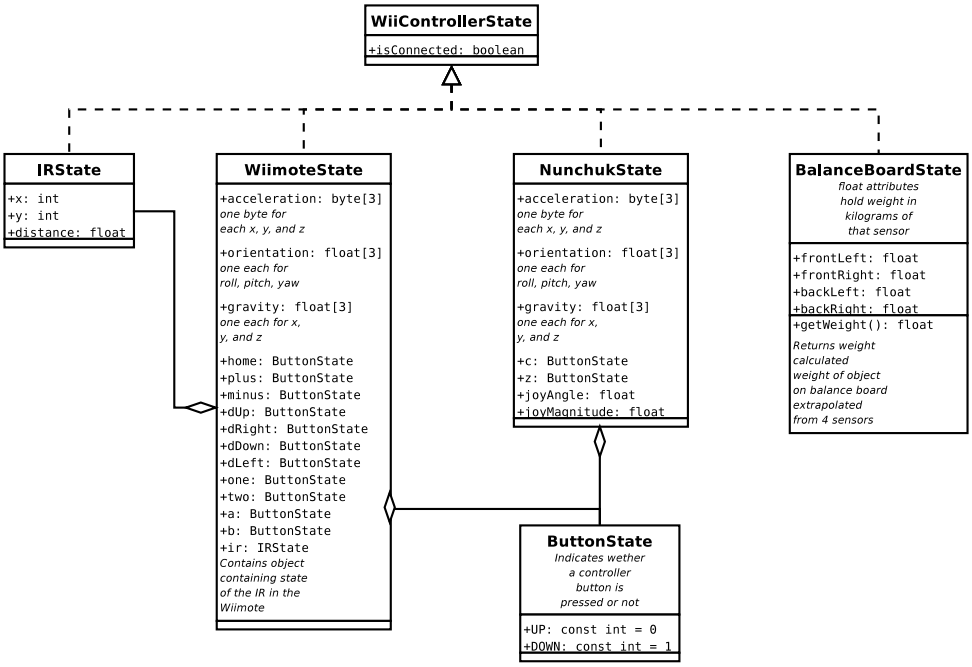

Figure 8: WiiControllerState UML

7 Data Model and Storage

The Data Model and Storage classes control loading and saving the profile and therapy information. The Data Model holds the in-memory data while the Data_Storage class makes it possible to save the information in XML and CSV formats.

Figure 9 depicts the UML model for the Profiles/Sessions Data Model.

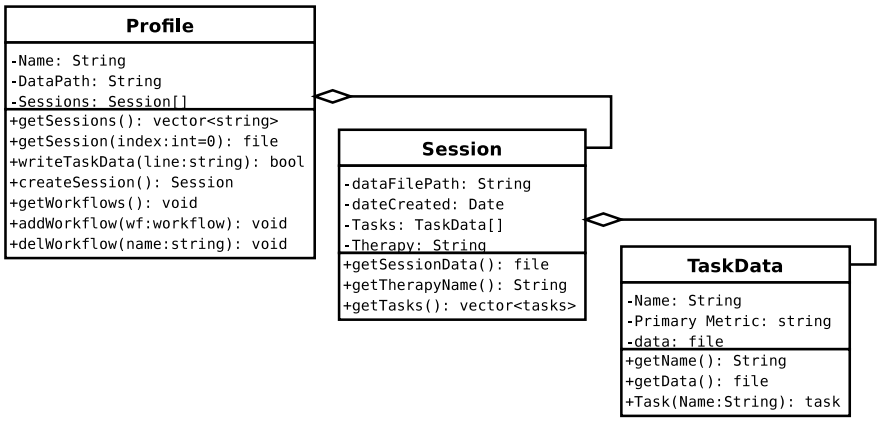

Figure 9: Profile/Session Data Model

Figure 10 depicts the UML model for the Therapy/Task Data Model.

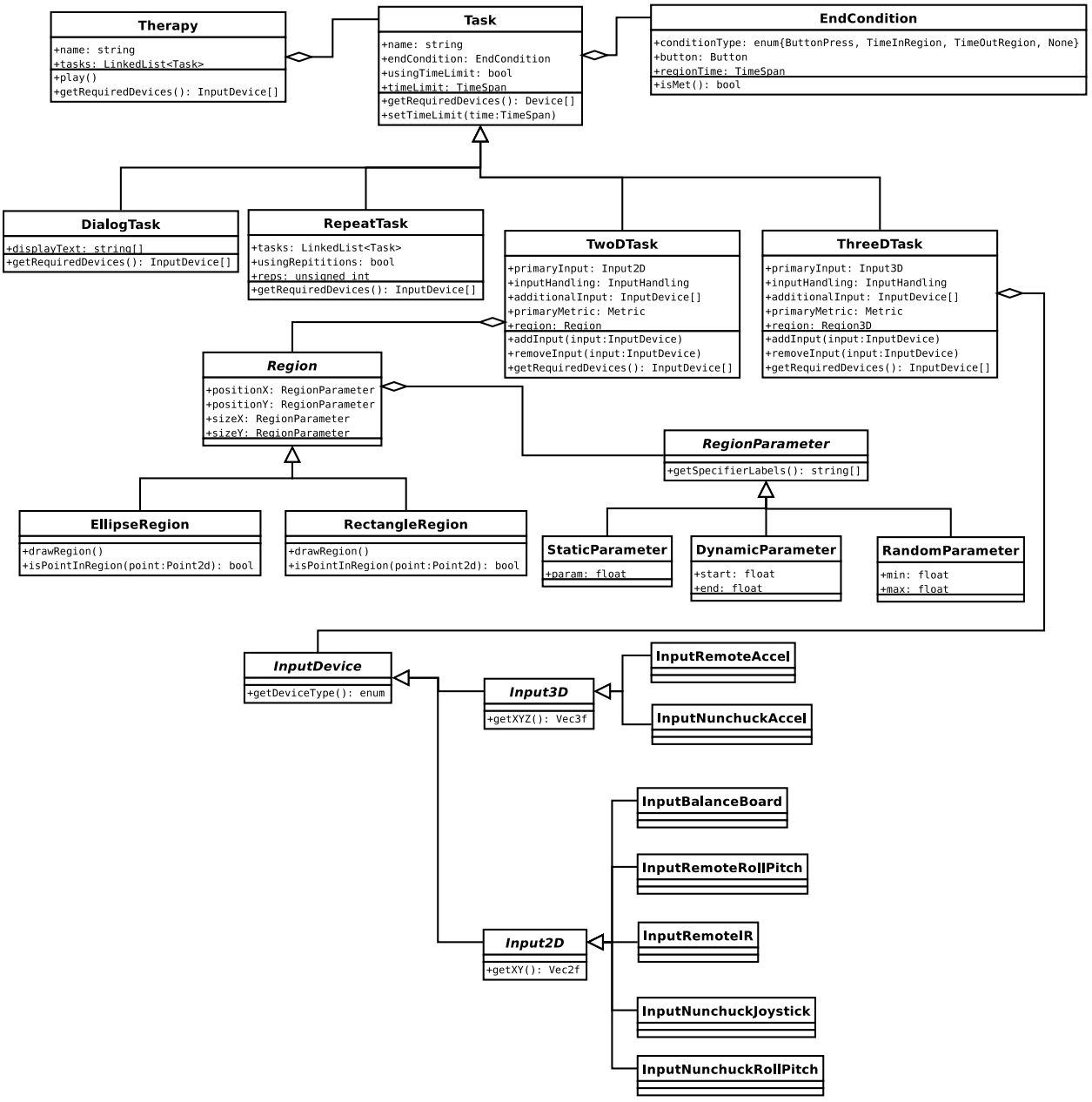

Figure 10: Therapy/Task Data Model

Figure 11 depicts the UML model for the Data Storage.

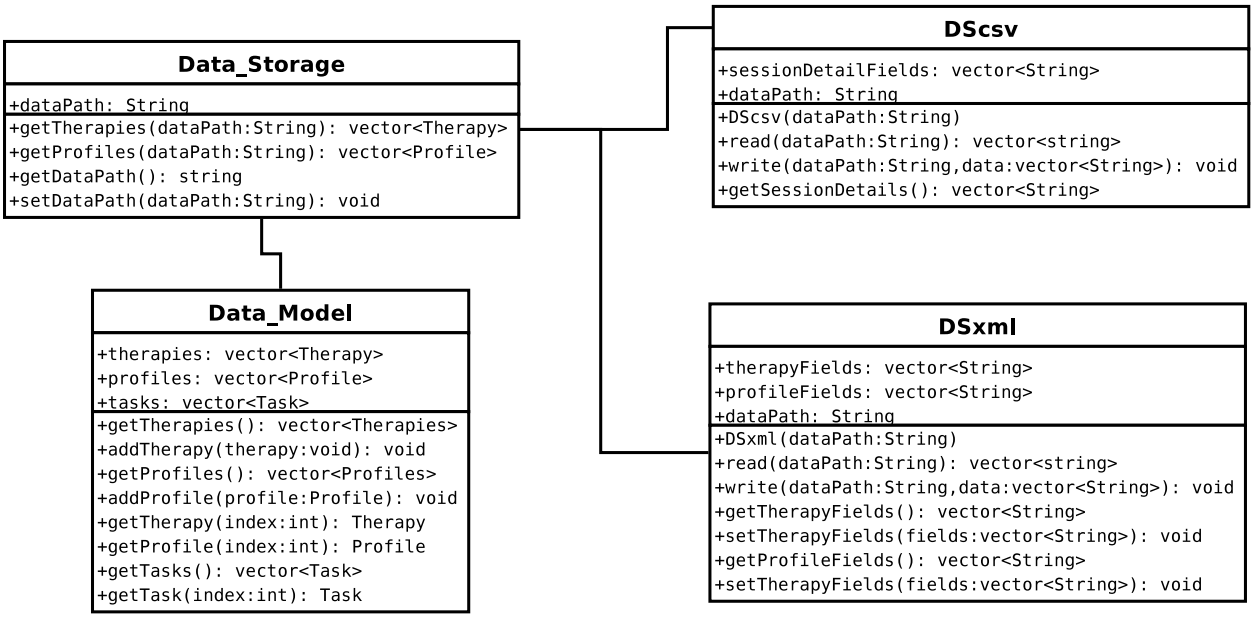

Figure 11: Data Storage

7.1 Data_Storage

7.1.1 Attributes

Name	Type	Description
dataPath	String	Contains the main directory for the storage of Profiles and Therapies

7.1.2 Methods

Vector<Therapy> getTherapies(String dataPath)	
Input:	A String that contains a file or directory of Therapies
Output:	A vector of objects of the Therapy Class loaded from the given directory or file
Description:	Method returns a vector of Therapy objects from the given directory or file

Vector<Profiles> getProfiles(String dataPath)	
Input:	A String that contains a file or directory of Profiles
Output:	A vector of objects of the Profile Class loaded from the given directory or file
Description:	Method returns a vector of Profiles objects from the given directory or file

String getDataPath()	
Input:	Void
Output:	A String that contains the main directory for the storage of Profiles and Therapies
Description:	Method returns the data path of the main storage directory

Void getDataPath(String dataPath)	
Input:	A String that represents the main directory that contains Profiles and Therapies
Output:	Void
Description:	Method sets the main directory that contains Profiles and Therapies

7.2 Data_Model

7.2.1 Attributes

Name	Type	Description
therapies	Vector<Therapy>	Contains a vector of Therapies that can be performed
profiles	Vector<Profile>	Contains a vector of Profiles of users that can perform Therapies
tasks	Vector<Task>	Contains a vector of Tasks that can be used to create Therapies

7.2.2 Methods

Vector<Therapy> getTherapies()	
Input:	Void
Output:	Returns a vector of Therapies that are stored in memory
Description:	Method returns a vector of Therapy objects that are stored in memory

Void addTherapy(Therapy therapy)	
Input:	A Therapy Object that has been created or loaded
Output:	Void
Description:	Method adds a new Therapy to the vector of therapies stored in memory

Vector<Profile> getProfiles()	
Input:	Void
Output:	Returns a vector of Profiles that are stored in memory
Description:	Method returns a vector of Profiles objects that are stored in memory

Void addProfile(Profile profile)	
Input:	A Profile Object that has been created or loaded
Output:	Void
Description:	Method adds a new Profile to the vector of profiles stored in memory

Therapy getTherapy(int index)	
Input:	The number of the Therapy requested from the vector of Therapies
Output:	Returns the Therapy that is in the index position of the vector of Therapies
Description:	Method returns the Therapy that is in the index position of the vector of Therapies

Profile getProfile(int index)	
Input:	The number of the Profile requested from the vector of Profiles
Output:	Returns the Profile that is in the index position of the vector of Profiles
Description:	Method returns the Profile that is in the index position of the vector of Profiles

Vector<Task> getTasks()	
Input:	Void
Output:	Returns a vector of Tasks that are available to create new Therapies
Description:	Method returns a vector of Tasks that are available to create new Therapies

Task getTask(int index)	
Input:	The number of the Task requested from the vector of Tasks
Output:	Returns the Task that is in the index position of the vector of Tasks Description: Method returns the Task that is in the index position of the vector of Tasks

7.3 DSxml

7.3.1 Attributes

Name	Type	Description
sessionDetailsFields	Vector<String>	Contains a vector of Strings indicating the names of the fields for a session
dataPath	String	Contains the main directory for the storage of sessions

7.3.2 Methods

Void DScsv(String dataPath)	
Input:	String that contains the main directory for the storage of sessions
Output:	Void
Description:	Method sets the main directory for the storage of sessions

Vector<String> read(String dataPath)	
Input:	String that contains the main directory for the storage of sessions
Output:	Vector of Strings representing the data in sessions
Description:	Method returns a vector of session details from the given Directory or File in csv format

Void write(String dataPath)	
Input:	String that contains the main directory for the storage of sessions
Output:	Void
Description:	Method writes a vector of session details to the given Directory or File in csv format

Vector<String> getSessionDetails()	
Input:	Void
Output:	Vector of strings that represent the session details
Description:	Method returns a vector of strings that represent the session details

7.4 DScsv

7.4.1 Attributes

Name	Type	Description
therapyFields	Vector<String>	Contains a vector of Strings indicating the names of the fields for a Therapy
profileFields	Vector<String>	Contains a vector of Strings indicating the names of the fields for a Profile
dataPath	String	Contains the main directory for the storage of Profiles and Therapies

7.4.2 Methods

Void DSxml(String dataPath)	
Input:	String that contains the main directory for storage
Output:	Void
Description:	Method sets the main directory for storage

Vector<String> read(String dataPath)	
Input:	String that contains the main directory for the storage of Profiles or Therapies
Output:	Vector of Strings representing the data in Profiles or Therapies
Description:	Method returns a vector of strings representing Therapies or Profiles from the given Directory or File in xml format

Void write(String dataPath)	
Input:	String that contains the main directory for the storage of profiles or therapies
Output:	Void
Description:	Method writes a vector of Profiles or Therapies to the given Directory or File in xml format

Vector<String> getTasksFields()	
Input:	Void
Output:	Vector of strings that represent the fields that create a Task
Description:	Method returns a vector of strings that represent the fields that create a Task

Void setTasksFields(vector<String> fields)	
Input:	a vector of Strings representing the fields names that make up a Task
Output:	Void
Description:	Method sets the vector of strings that represent the fields that create a Task

8 References

[1] Winter, D. A. *A. B. C. (Anatomy, Biomechanics, Control) of Balance during Standing and Walking.* Waterloo Biomechanics, Waterloo, Ontario, Canada, 1995.