

Project: Sapien 190
Date: 15.05.2010
Version: 0.03

Product architecture document for Sapien 190

Embedded Real-Time Systems (TI-IRTS)
Spring 2010

Peter Høgh Mikkelsen (20087291)
Anders Block Arnfast (20085515)
Kim Bjerge (20097553)

PAsien

Document history

Date	Version	Description	Author
24.04.2010	0.00	Initial Version	KBE
10.05.2010	0.01	Updated with Use Case View and initial UML diagrams for first delivery UC#1	KBE
15.05.2010	0.02	Updated diagrams with UC#1 and added IPump protocol in comm package	KBE
15.05.2010	0.03	Added command pattern for setting parameters	KBE

Template User guide:

- To be deleted in the final document

This Template is a highly modified version of a RUP template (Rational Unified Process) for a software architecture document. The modifications includes elements from the Danish Structured Development Handbooks (SPU) SW documentation guide. The template is based on the famous “4+1” View Model described by Philippe Kruchten, “The 4+1 View Model of Architecture”, IEEE Software Nov.1995.

The Template is written as a **Microsoft Word .dot** file. A double click with the mouse on the xx.dot file in the file manager will open a new word document with the default name: Document 1, including the information in the template.

Start with modifying the document title and subject. Document title and subject (e.g. System/product <name>) is modified in **File/Properties – Summary**. Title and subject will automatically be inserted on the front page and in the page header. Save the document with a new name.

If the **.dot** file is installed in the directory where word looks for template files, it will be possible to activate a new document based on this template by selecting **File/New** from the Word program.

For helping the user with writing the document, help text is included for each section in the document. This help text is written as “Hidden Text” in Word.

To show (enable) the help text on the screen:

Menu choice: **Tools/Options – View**

and check the check box: Hidden text (alternatively remove the mark – for removing the help text).

To print the template file with help text:

Menu choice: **Tools/Options – Print**

and check the check box Hidden text.

To modify the template:

Start Word and open the template by selecting: **File/Open**

Table of Contents

1. INTRODUCTION.....	3
1.1 Purpose and Scope	3
1.2 References	3
1.3 Definitions and acronyms	3
1.4 Document structure and reading guide	4
1.5 Document role in an iterative development process	4
2. SYSTEM OVEVIEW	4
2.1 System context	4
2.2 System introduction	4
3. SYSTEMET INTERFACES	5
3.1 Interface to human actors	5
3.2 Interface to external system actors	5
3.3 Interface to hardware actors	5
3.3.1 Pulse interface	5
3.3.2 Infusion pump interface	5
3.4 Interface to external software actors	5
4. USE CASE VIEW	5
4.1 Overview of architecture significant Use cases	6
4.2 Use case #1: Execute and Control Simulation scenarios	6
4.2.1 Use case goal	6
4.2.2 Use case scenarios	6
4.3 Use case # 2: Select and Initiate Scenario scenarios.....	7
5. LOGICAL VIEW.....	8
5.1 Overview	8
5.2 Architecturally significant design packages.....	9
5.2.1 Continuous Package.....	9
5.2.2 Discrete	13
5.2.3 Com.....	14
5.2.4 AbstractHW	15
5.2.5 AbstractOS	15
5.2.6 Application Helper Classes.....	16
5.3 Use case realizations	17
5.3.1 Use case #1: Execute and Control Simulation scenarios	17
5.3.2 Use case 2. realization	23
6. PROCESS/TASK VIEW	24
6.1 Process/task overview	24
6.2 Process/task implementation	24
6.3 Process/task communication and synchronization.....	24
6.4 Process group 1.	24
6.4.1 Process communication in group 1	24
6.4.2 Process 1. description	24
6.4.3 Process 2. description	24
6.5 Process group 2.	25

7. DEPLOYMENT VIEW	25
7.1 System configurations overview	25
7.2 System configurations	25
7.2.1 Configuration 1.	25
7.2.2 Configuration 2.	25
7.3 Node descriptions	25
7.3.1 Node 1. description	25
7.3.2 Node 2. description	25
8. IMPLEMENTATION VIEW	26
8.1 Overview	26
8.2 Component descriptions	26
8.2.1 Component 1	26
8.2.2 Component 2	26
9. DATA VIEW	26
9.1 Data model	26
9.2 Implementation of persistence	26
10. GENEREL DESIGN DECISIONS	26
10.1 Architectural goals and constraints	26
10.2 Architectural patterns	26
10.3 General user interface design rules	27
10.4 Exception and error handling	27
10.5 Implementation languages and tools	27
10.6 Implementation libraries	27
11. SIZE AND PERFORMANCE	27
12. QUALITY	27
13. COMPILATION AND LINKING	27
13.1 Compilation-hardware	27
13.2 Compilation-software	27
13.3 Compilation and linking process	27
14. INSTALLATION AND EXECUTING	27
14.1 Installation	27
14.2 Executing-hardware	27
14.3 Executing-software	27
14.4 Execution-control (start, stop and restart)	28
14.5 Error messages	28
15. APPENDICES	28

1. INTRODUCTION

1.1 Purpose and Scope

The Sapien 190 is a patient simulator, simulating human physiological behaviour according to different patient scenarios. Scenarios are written in an open format and can be downloaded to the Sapien 190 that contains patient records from the PhysioBank database.

In conjunction with the Sapien 110 human doll, the Sapien system provides a complete simulated human interface, with EKG, ECG and respiratory measuring spots and medicine injection spots.

1.2 References

- [1] Erich Gamma et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley (GoF)
- [2] Bruce Powell Douglass, Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems
- [3] PhysioNet and PhysioBank the research resource for complex physiologic signals
<http://www.physionet.org/>
- [4] Project specification for PSIMU: Patient Simulator System
<http://kurser.iha.dk/eit/tiirts/Projekter/PSIMU-project.doc>
- [5] Project specification for LMON: Local Monitor System
<http://kurser.iha.dk/eit/tiirts/Projekter/LMON-project.doc>
- [6] Project specification for IPUMP: Infusion Pump System
<http://kurser.iha.dk/eit/tiirts/Projekter/IPUMP-project.doc>
- [7] Project Interface Specification
<http://kurser.iha.dk/eit/tiirts/Projekter/ProjectInterfaces.doc>
- [8] Requirement specification for Sapien 190

1.3 Definitions and acronyms

- **Model** – A model that represents one physical individual. The model may consist of sub-models for different body subsystems. The model uses an algorithm to compute the output signals based on the input signals or patient records.
- **Model Parameters** – Parameters used in the model.
- **Record** – A patient record file taken from the PhysioBank database
- **Scenario** – Model parameters and a collection of records taken from the PhysioBank database.
- **Scenario Configuration** – A set of files that represents model parameters and patient records.
- **Signals** – Signals in form of waveform files or input from external equipment
- **Simulation** – A continuous mode, where the physiological output signals are updated according to the model and the scenario applied to it.
- **EKG** – Electrocardiogram
- **EDR** – ECG-Derived Respiration

1.4 Document structure and reading guide

Chapter 2 and 3 gives an overview of the product and the interfaces to the patient simulator in terms of other devices and user operation.

The document describes the design using the “4+1” view. For each iteration as specified in ROPES [2] we have selected one or more use cases that is used in describing the Use Case, Logical, Process, Deployment and Implementations Views. These views are described in chapters 5 – 8.

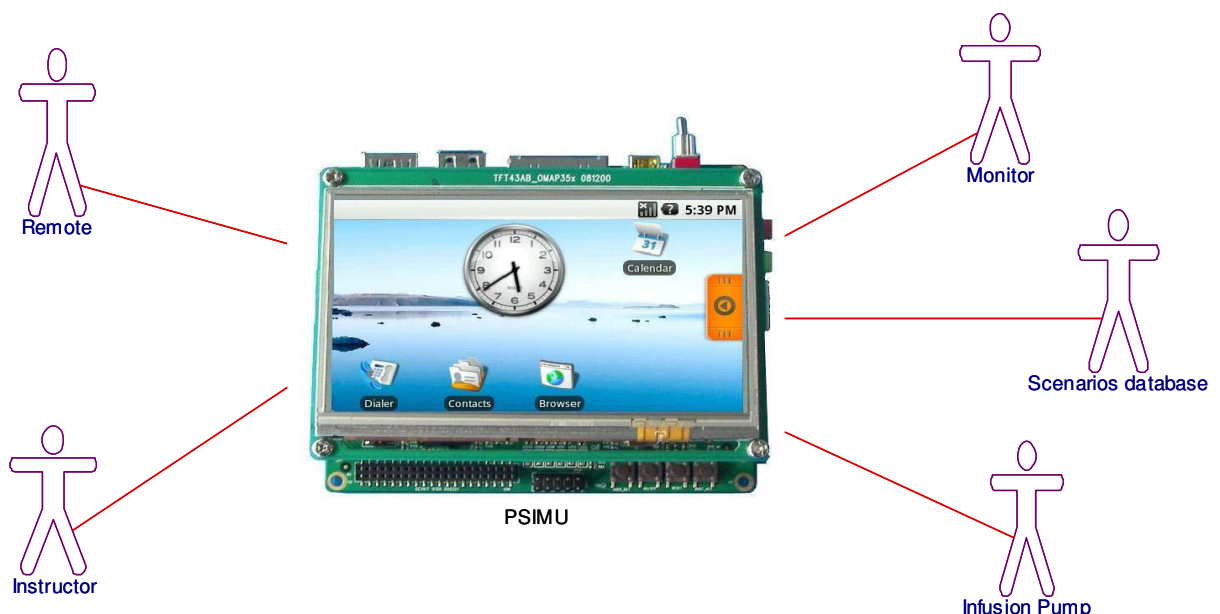
1.5 Document role in an iterative development process

This document is updated in using the ROPES development process [2]. The document is updated for every design cycle of the ROPES spiral microcycle. Every design microcycle is using the 4+1 view

2. SYSTEM OVEVIEW

The Sapien 190 is a compact unit with a graphical user interface (GUI) and interfaces to a range of body monitoring and injection equipment. The Sapien 190 can be operated by the instructor and connected to a bedside monitor that used the output signals from the patient simulator. Optional a medicine infusion pump can be connected to the simulator.

2.1 System context



2.2 System introduction

The instructor will be able to control the simulation like forcing a heart attack and monitor the waveform of the signals that is send to the bedside monitor. Two analogue outputs signals can be connected to a bedside monitor to display ECG and EDR signals. It is possible remotely using an Ethernet connection to the patient simulator to update and delete files in the scenarios database on Sapien 190. The optional infusion pump injects medicine into the patient and the flow of medicine can be monitored on the Sapien 190 LCD display.

3. SYSTEMET INTERFACES

3.1 Interface to human actors

Instructor controls the patient simulation by inputs to the LCD touch screen....

3.2 Interface to external system actors

Remotely it is possible to update the scenarios database by use of a ftp connection with a Ethernet connected to the patient simulator.

3.3 Interface to hardware actors

Analogue outputs signals to the LMON has a resolutions of 12 bits sampled at 250 hz.
The analogue output has a voltage range of 0 - 4.096 volts.

3.3.1 Pulse interface

The pulse is send to a connected monitor using a RS232 connection (115200 baud, 8 databits, even parity, 1 stop bit).

Format of the pulse signal is a maximum 3 bytes ASCII value (beats/minute). It is transmitted from the PSIMU to the LMON with a rate of 1 second. The pulse value is terminated with a <CR>. <Pulse> <CR>

Example a terminal can be connected to display the pulse values updated each second:

103

107

3.3.2 Infusion pump interface

The infusion pump is connected using a RS232 connection with the protocol described below.

A fixed 64 bytes ASCII PDU is transmitted from IPUMP to PSIMU.

This PDU is transmitted every second when the pump is started.

PDU format:

4 bytes startframe (##?*)

46 bytes medicin name

12 bytes volume infused (since started)

2 bytes CRC checksum

3.4 Interface to external software actors

4. USE CASE VIEW

In the first iteration the UC #1 has been selected since it provides the main functionality and is the essential use case for the architecture of the patient simulator.

Execute and Control Simulation

When the patient simulation is running it will perform reading of the digitized physiologic signals and send these “real time” values as analogue signals to local connected bedside monitoring equipments. Up to 2 analogue channels with different signals is possible to be simulated simultaneously. The simulated signals can be ECG or EDR. The pulse will be signaled to the bedside monitoring equipment as a digital signal.

4.1 Overview of architecture significant Use cases

The UC #1 has been selected for the first iteration of the ROPES spiral microcycle [2] in making the architectural, mechanistic and detailed design. This use case is significant and provides the central functionality of the patient simulator. This use case provides the basis functionality that allows the monitor to be connected being able to display the ECG and EDR signals. It also reduces the risk for developing the patient monitor since it covers all the unknown technologies of the product like:

- Reading the patient record files on the target (Linux, WFDB and target)
- Generating the analogue output signals (Writing to drivers)
- Display of signal waveform using Qt on target (Working with Qt on target)

4.2 Use case #1: Execute and Control Simulation scenarios

4.2.1 Use case goal

To simulate signals from the patient based on the selected scenario.
The waveform of signals to monitor and the status of patient are displayed.
The instructor must be able to monitor the simulated patient.

4.2.2 Use case scenarios

Scenario 1 - normal:

1. Opens scenario file
2. Search for record file
3. Opens record file
4. Continues to reads samples from record file and performs:
 - a. Use patient model to generate ECG signal
 - b. Use patient model and ECG signal to calculate EDR signal
 - c. Use patient model and ECG signal to calculate pulse
 - d. Update output signals: Pulse, ECG and EDR

Scenario 2 – normal with alternative scenario record:

1. Opens scenario file
2. Search for record file
3. Opens record file
4. Continues to reads samples from record file and performs:
 - a. Use patient model to generate ECG signal
 - b. Use patient model and ECG signal to calculate EDR signal
 - c. Use patient model and ECG signal to calculate pulse

- d. Update output signals: Pulse, ECG and EDR
5. Open alternative record file after specified simulation time and continues at 2.

Scenario 3 – normal with IPUMP:

1. Opens scenario file
2. Search for record file
3. Opens record file
4. Continues to reads samples from record file and performs:
 - a. Reads medicine and volume from IPUMP
 - b. Use patient model to generate ECG signal
 - c. Use patient model and ECG signal to calculate EDR signal
 - d. Use patient model and ECG signal to calculate pulse
 - e. Update output signals: Pulse, ECG and EDR

Scenario 4 – error opening record file:

1. Opens scenario file
2. Failed to search and open record file
3. Error message on LCD display

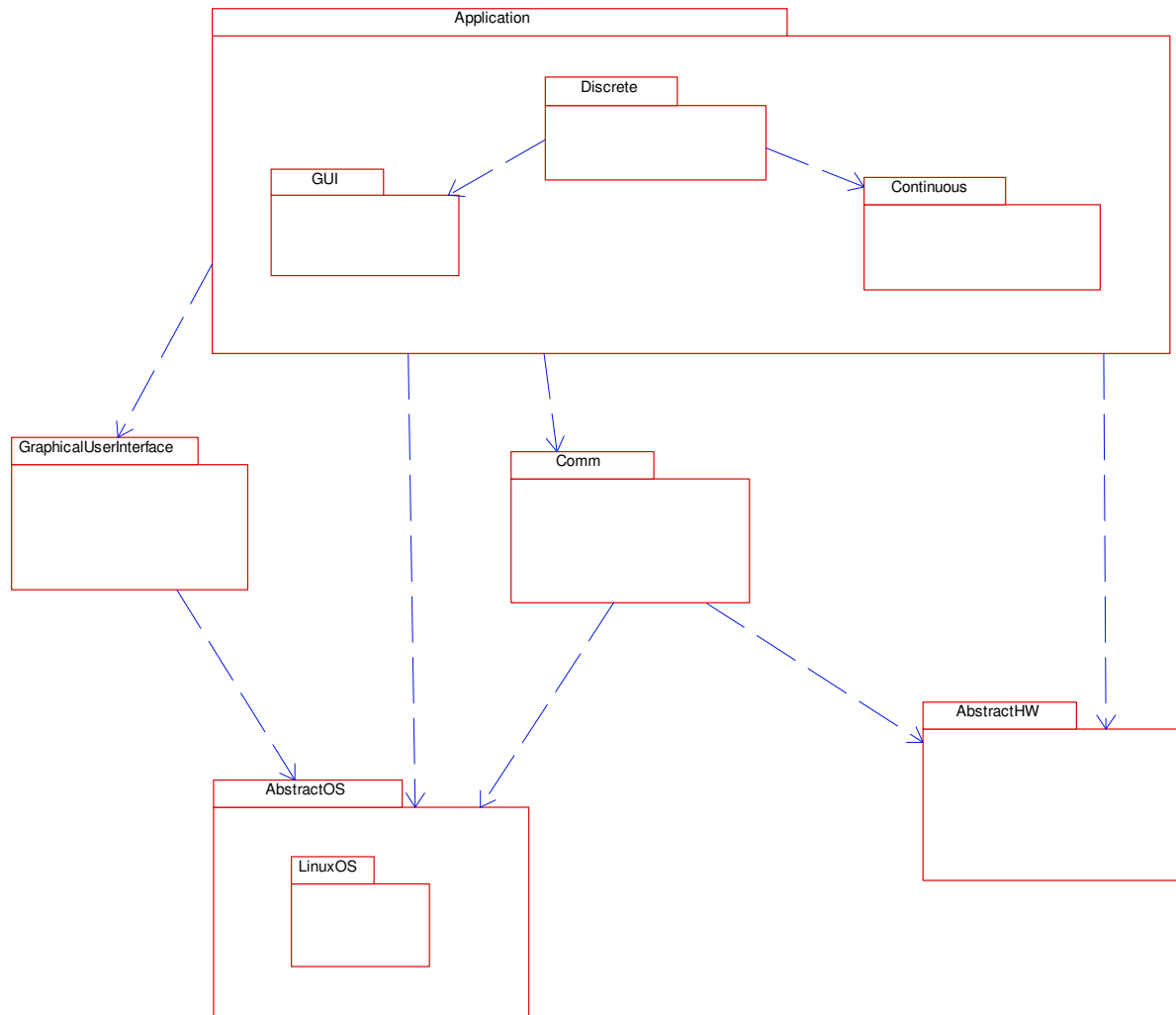
4.3 Use case # 2: Select and Initiate Scenario scenarios

Initiate the simulation using a scenario configuration chosen by the instructor.

This use case is not part of the first iteration and will be updated in the next iteration.

5. LOGICAL VIEW

5.1 Overview



5.2 Architecturally significant design packages

5.2.1 Continuous Package

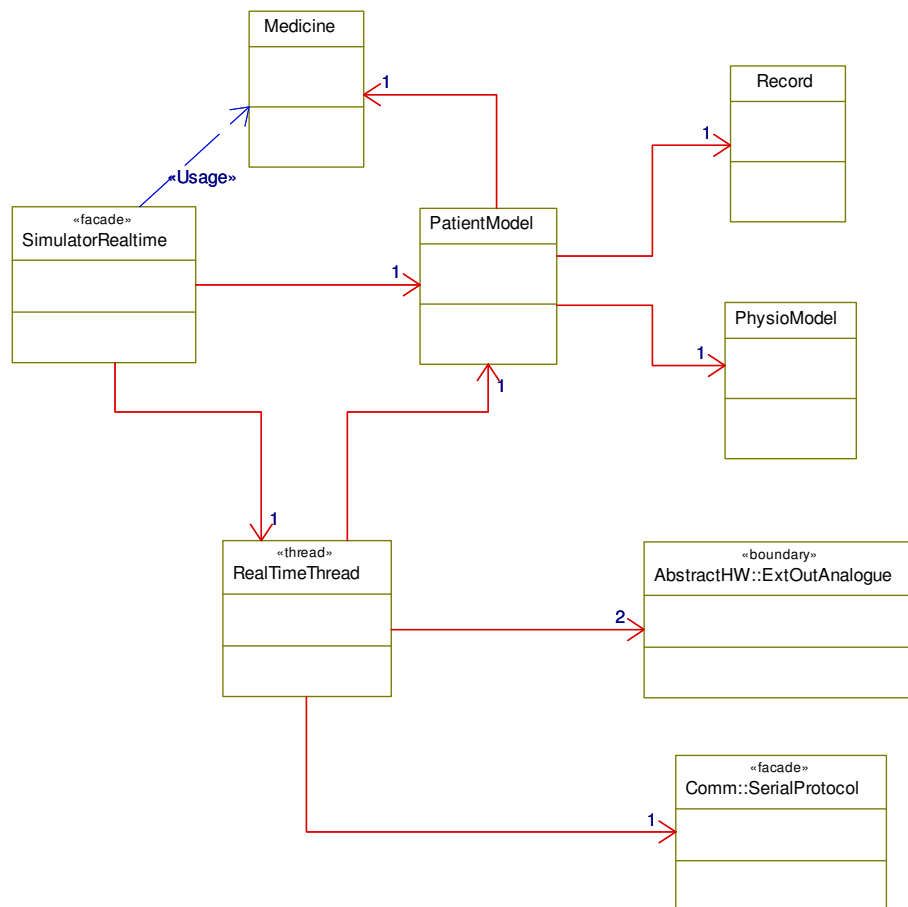


Figure 1 Major Classes in Continuous Package

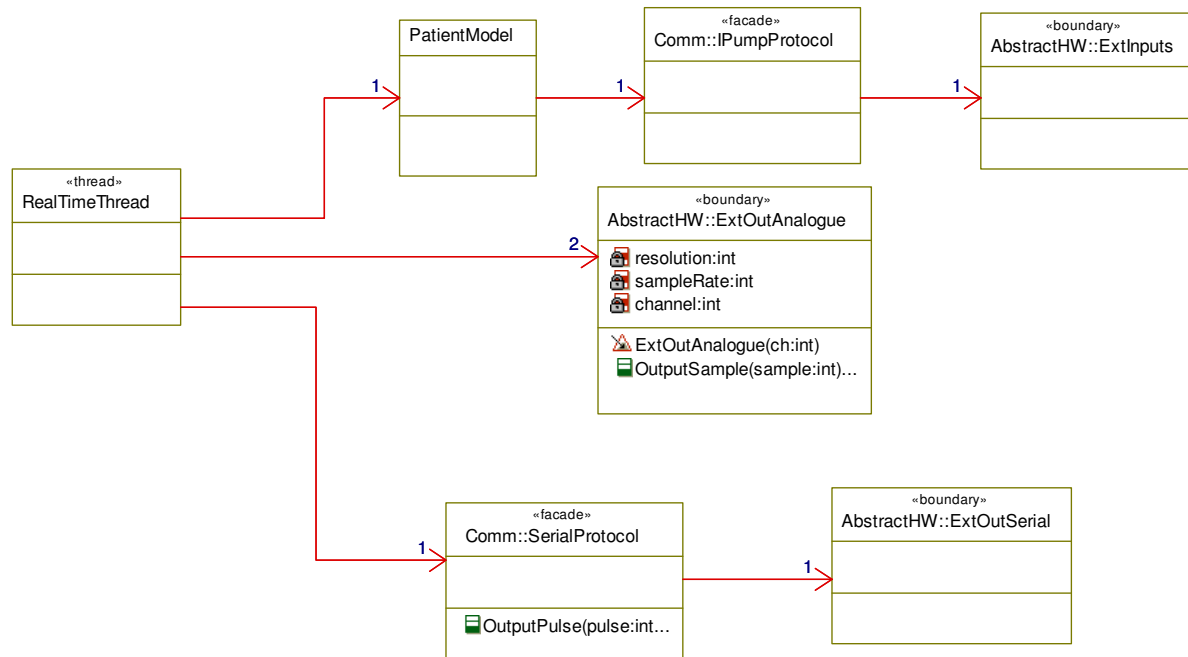


Figure 2 Boundary Classes for RealTimeThread

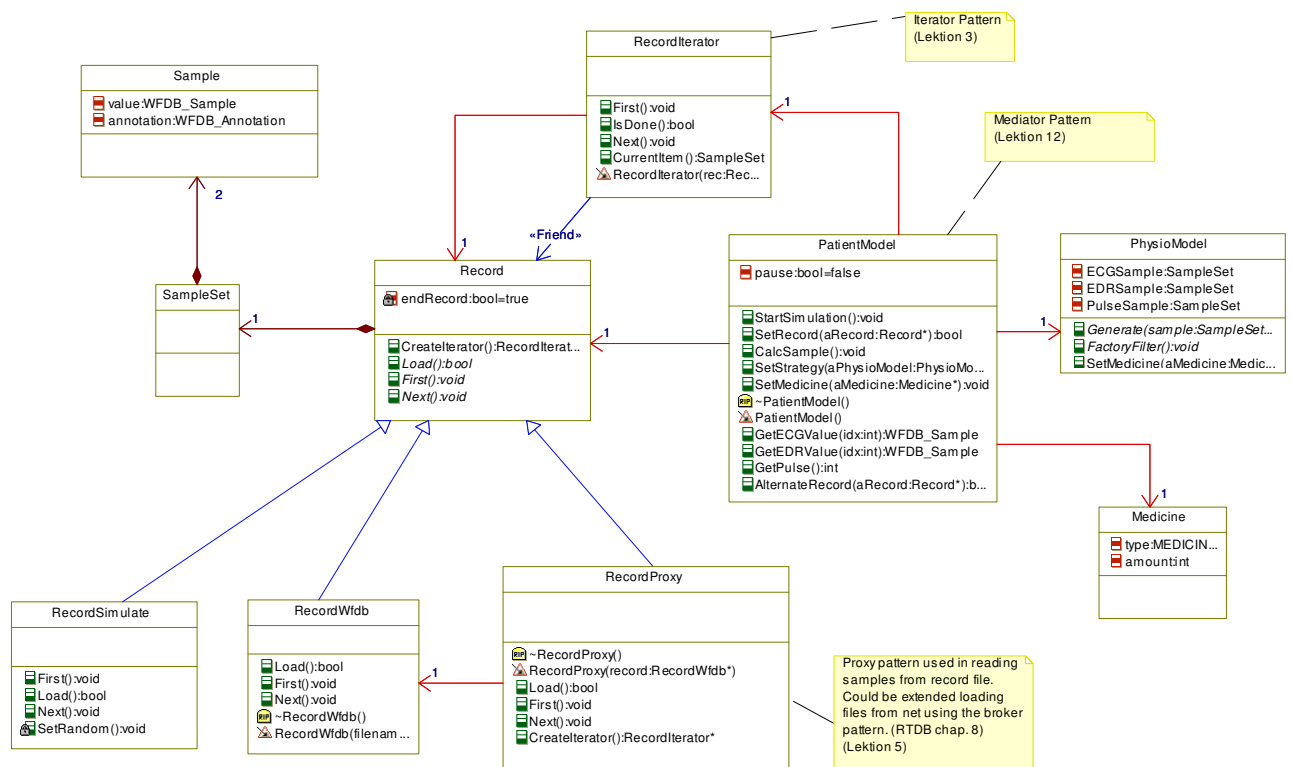


Figure 3 Proxy and Iterator Pattern used to access Records from the PatientModel

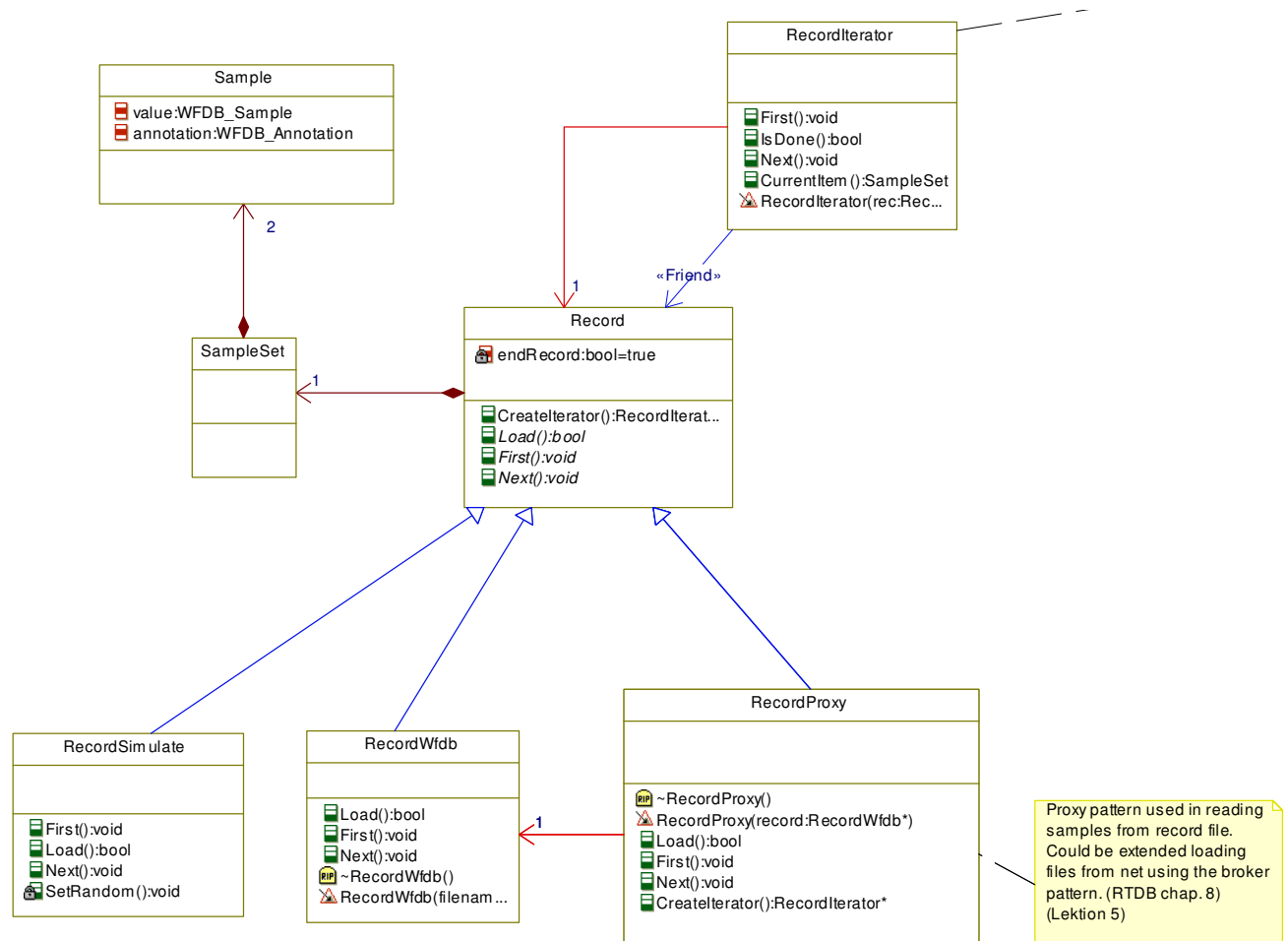


Figure 4 Record and Iterator

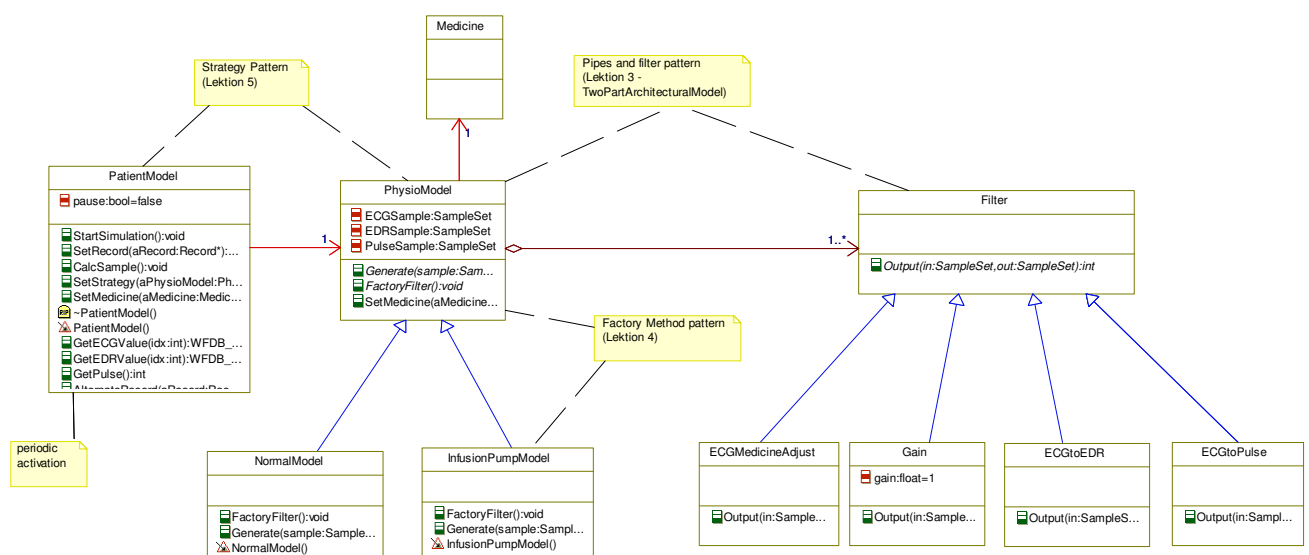


Figure 5 Strategy, Filter and Pipes Pattern used for PhysioModel

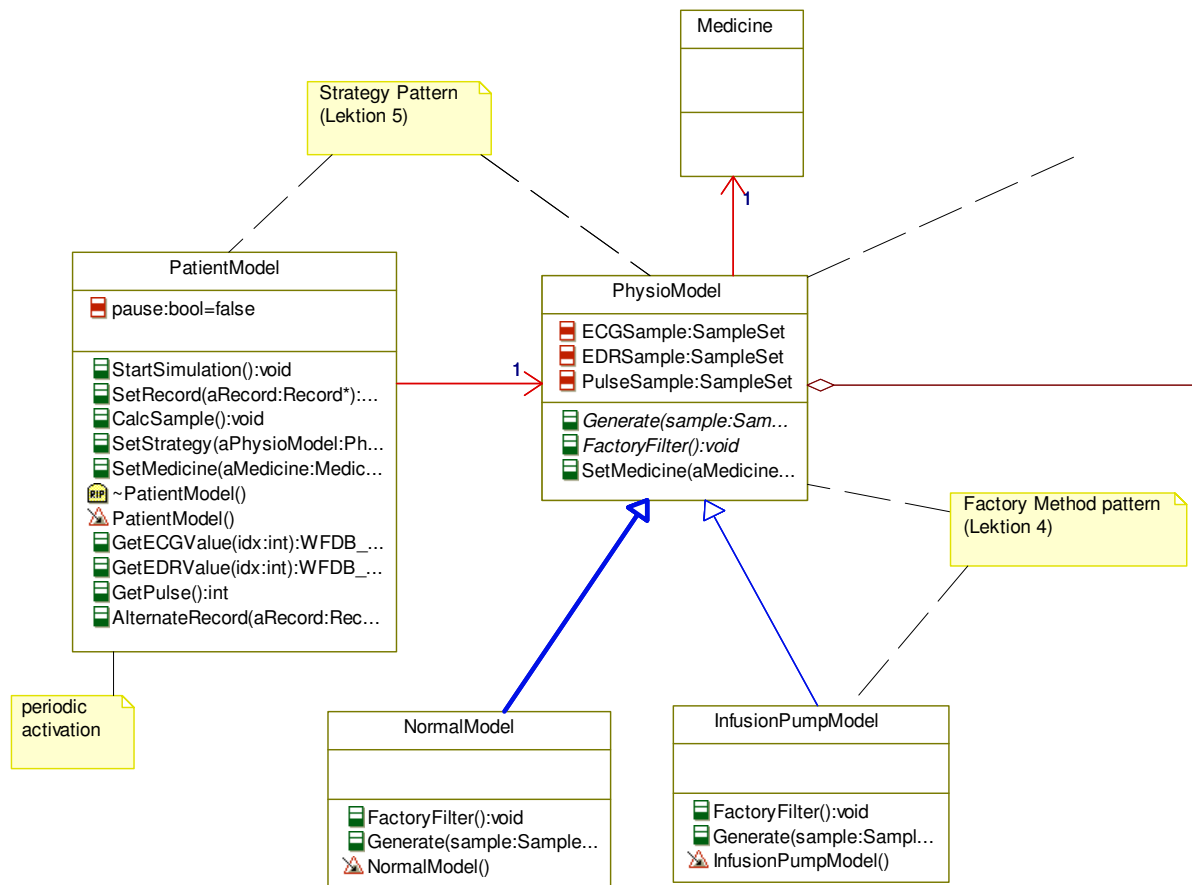


Figure 6 Strategy for PatientModel

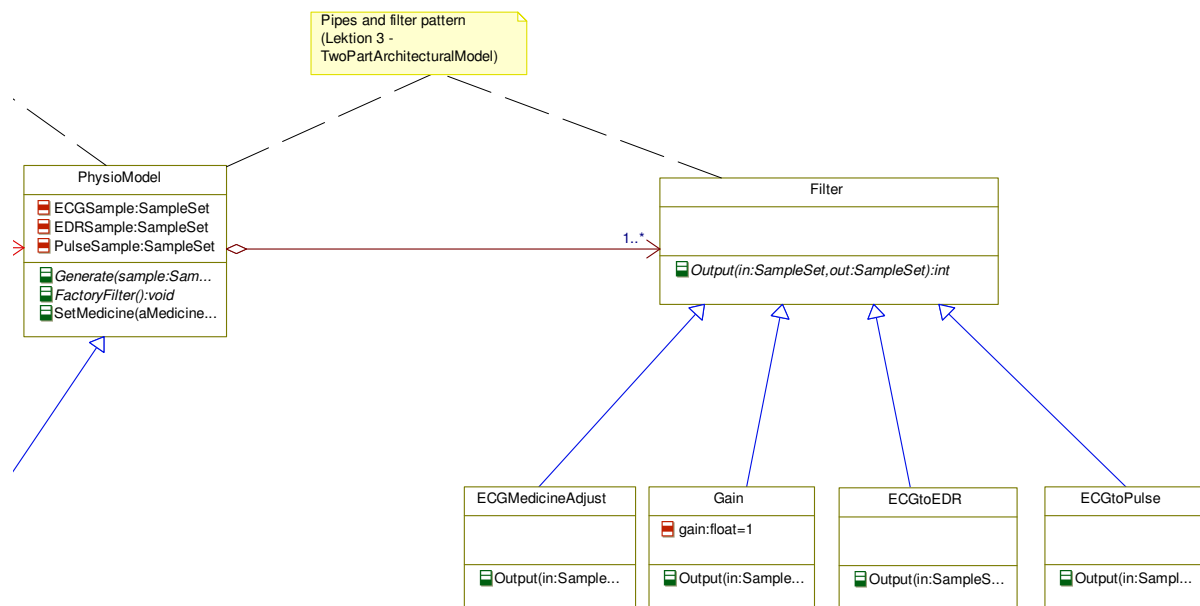


Figure 7 Filter Pattern for PhysioModel

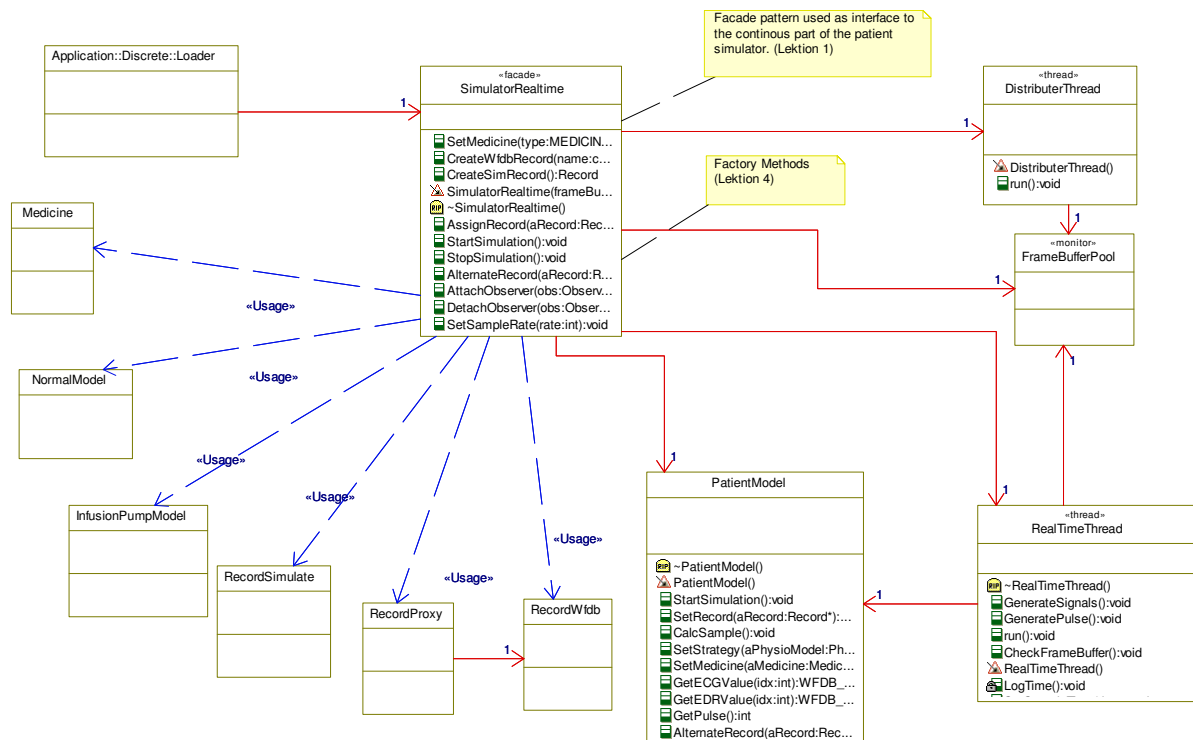


Figure 8 Façade Pattern used for interface to the Continuous Package

5.2.2 Discrete

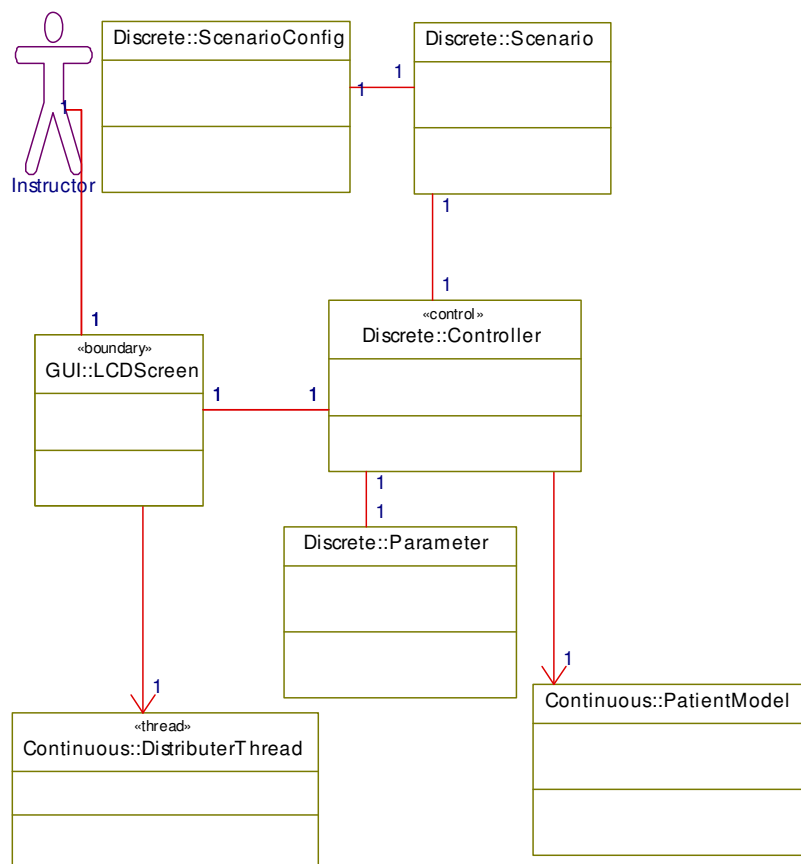


Figure 9 Application model for discrete package

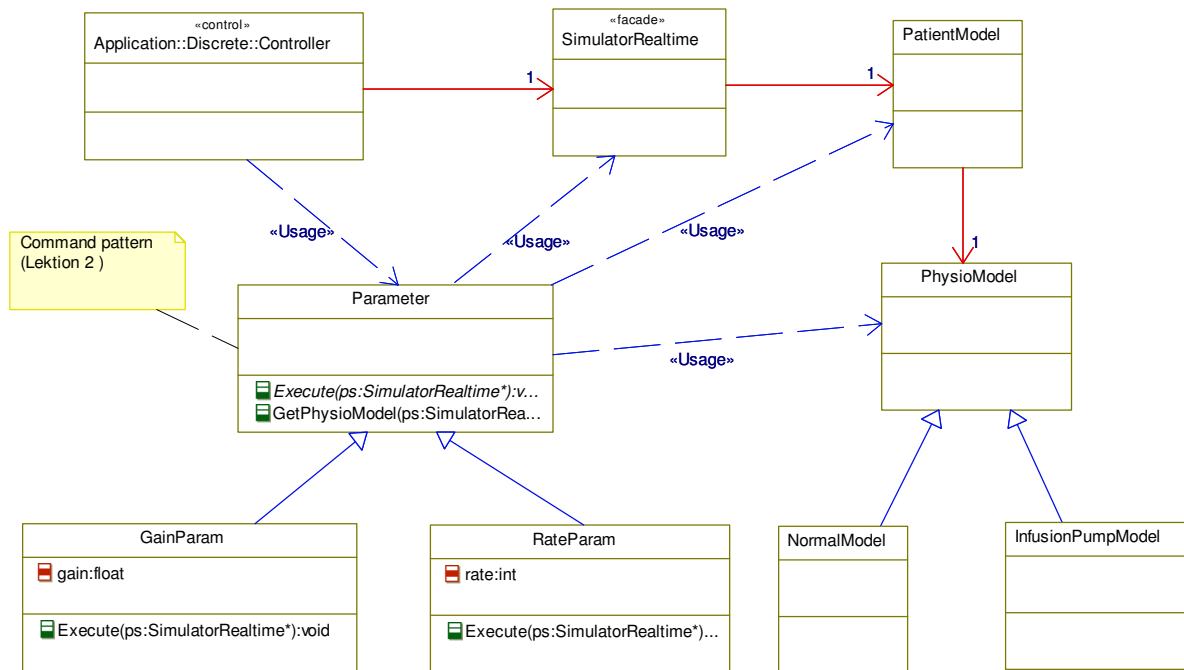


Figure 10 Command pattern used to set parameters for real time simulator

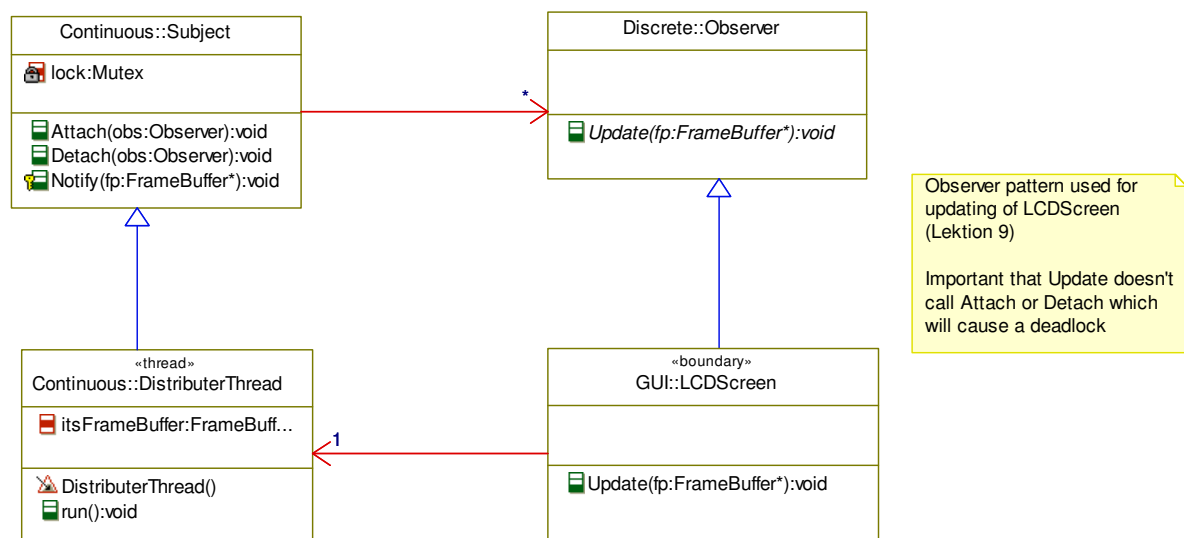


Figure 11 Observer pattern used to notify GUI with new framebuffer

5.2.3 Com

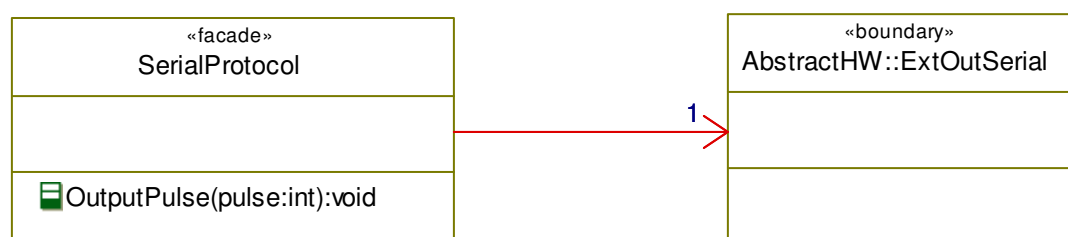


Figure 12 Communication package Serial protocol

5.2.4 AbstractHW

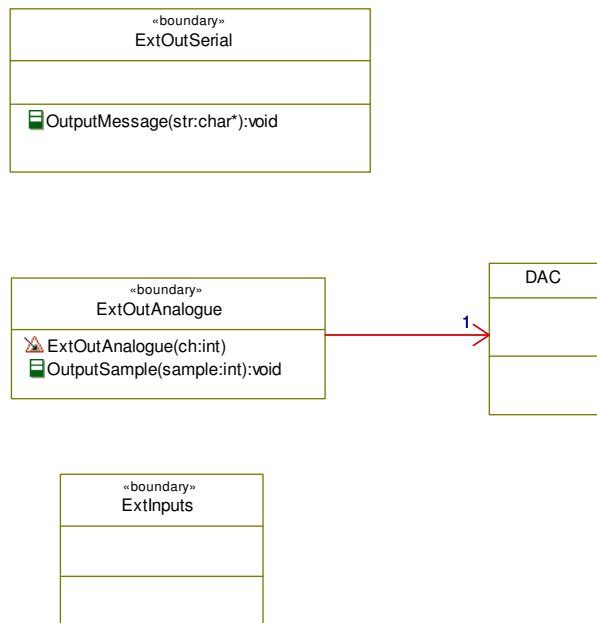


Figure 13 Hardware Abstraction

5.2.5 AbstractOS

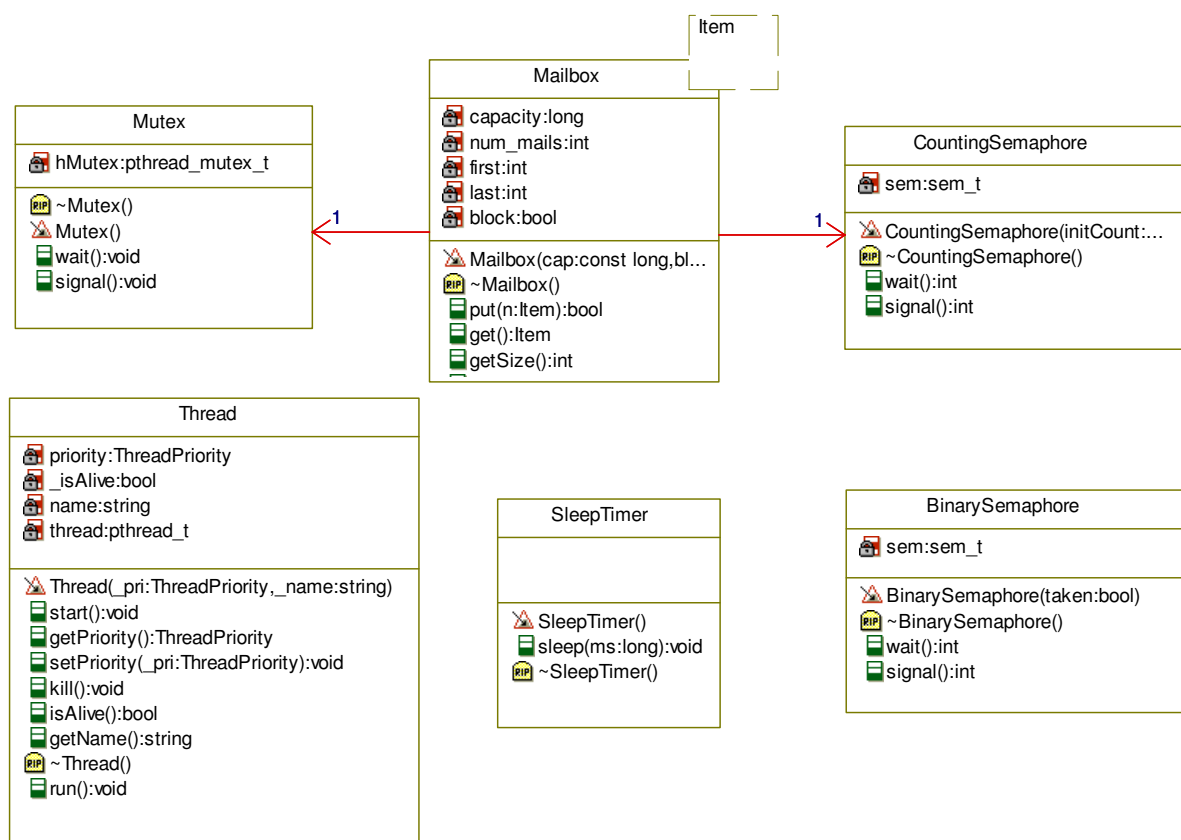


Figure 14 Abstract OS (Linux)

5.2.6 Application Helper Classes

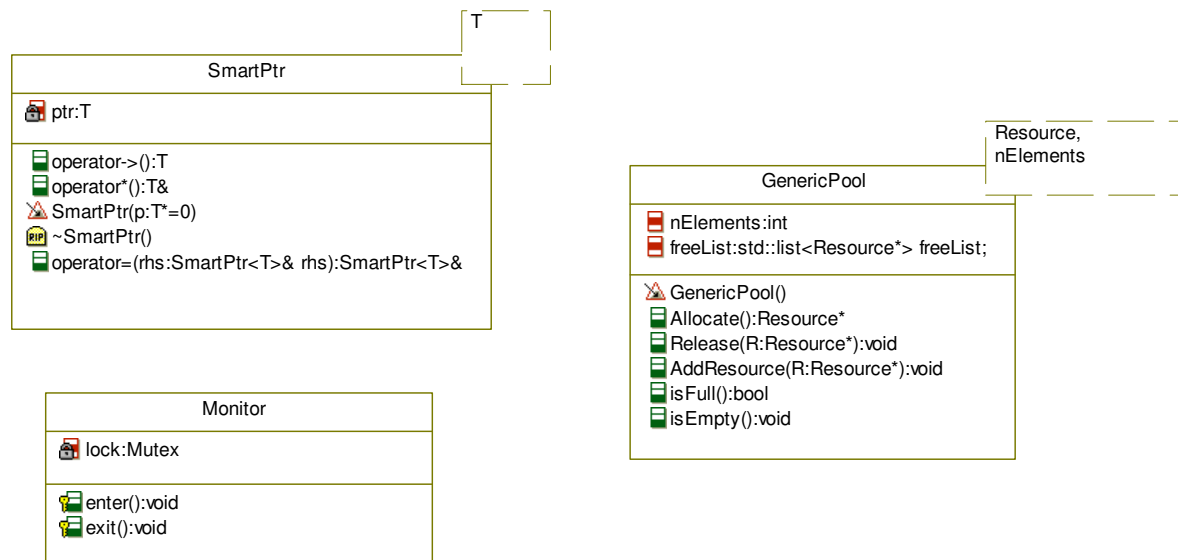


Figure 15 Application Helper classes

5.3 Use case realizations

5.3.1 Use case #1: Execute and Control Simulation scenarios

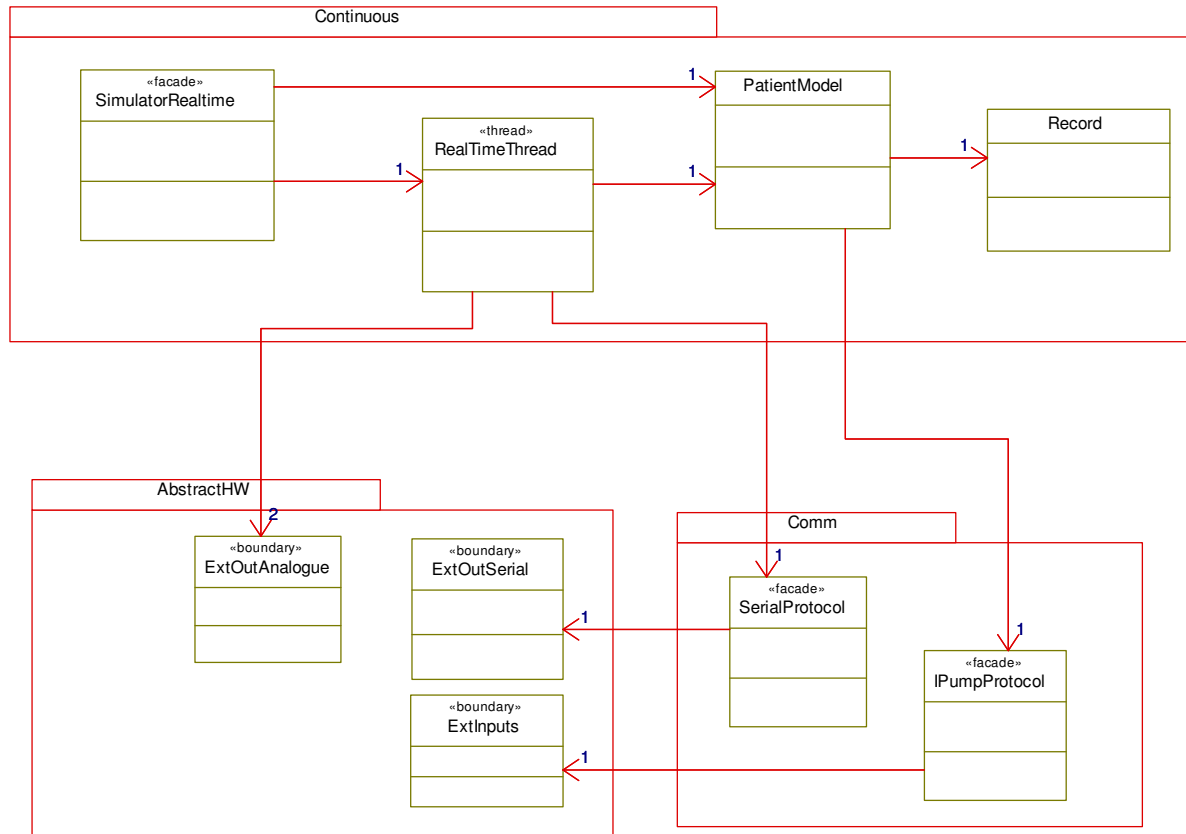


Figure 16 Logical view for use case #1 Execute and Control Simulation

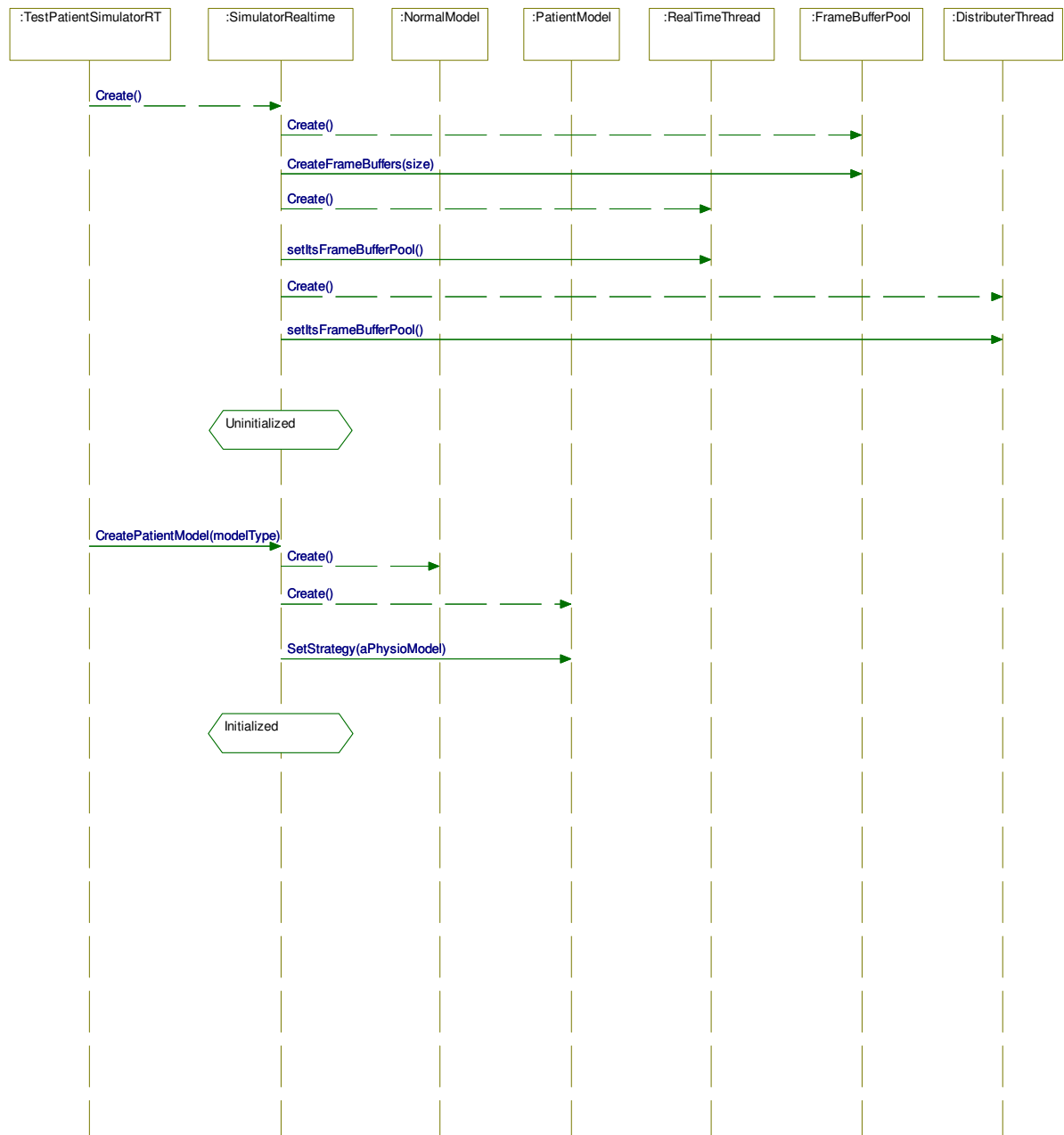
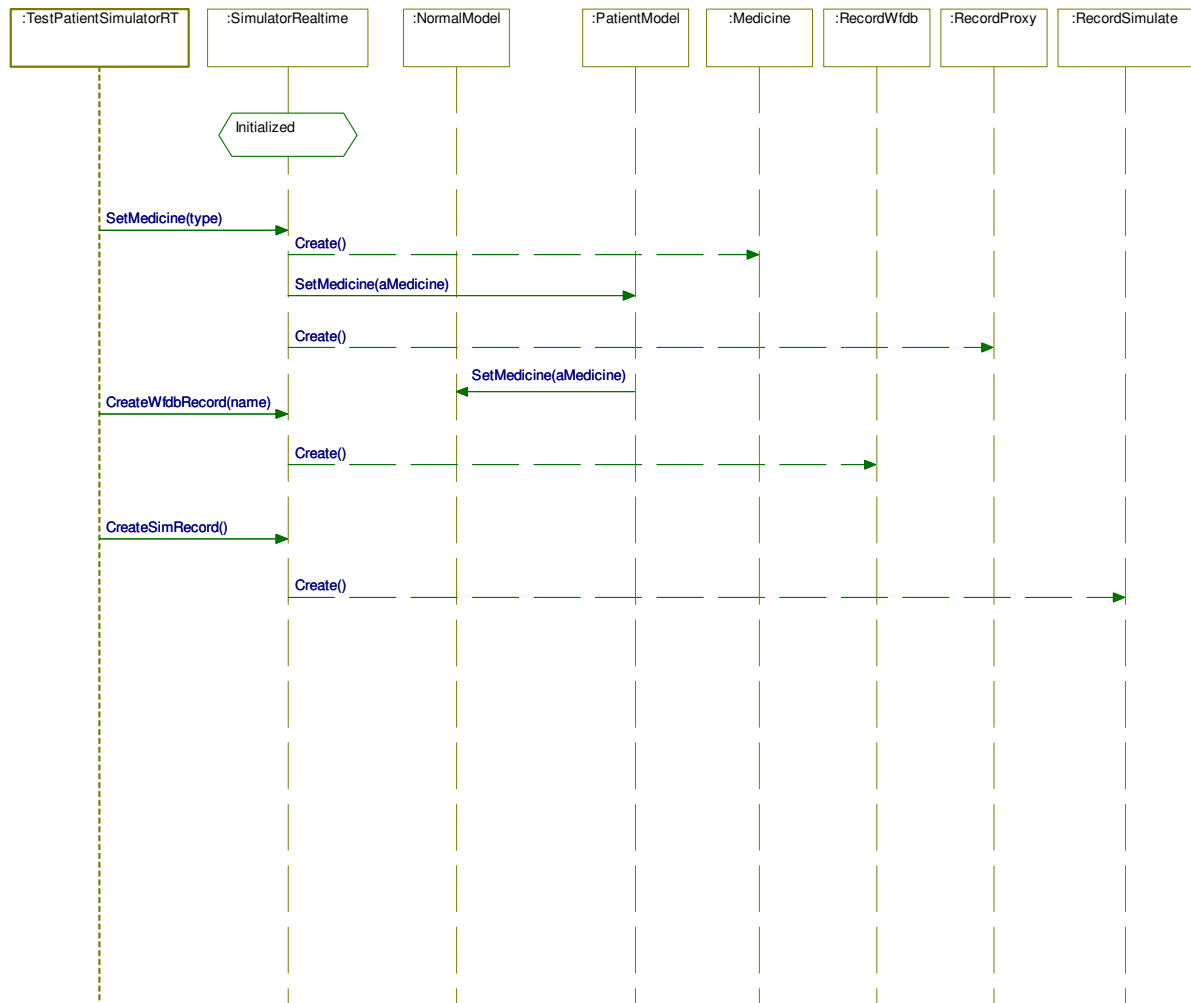
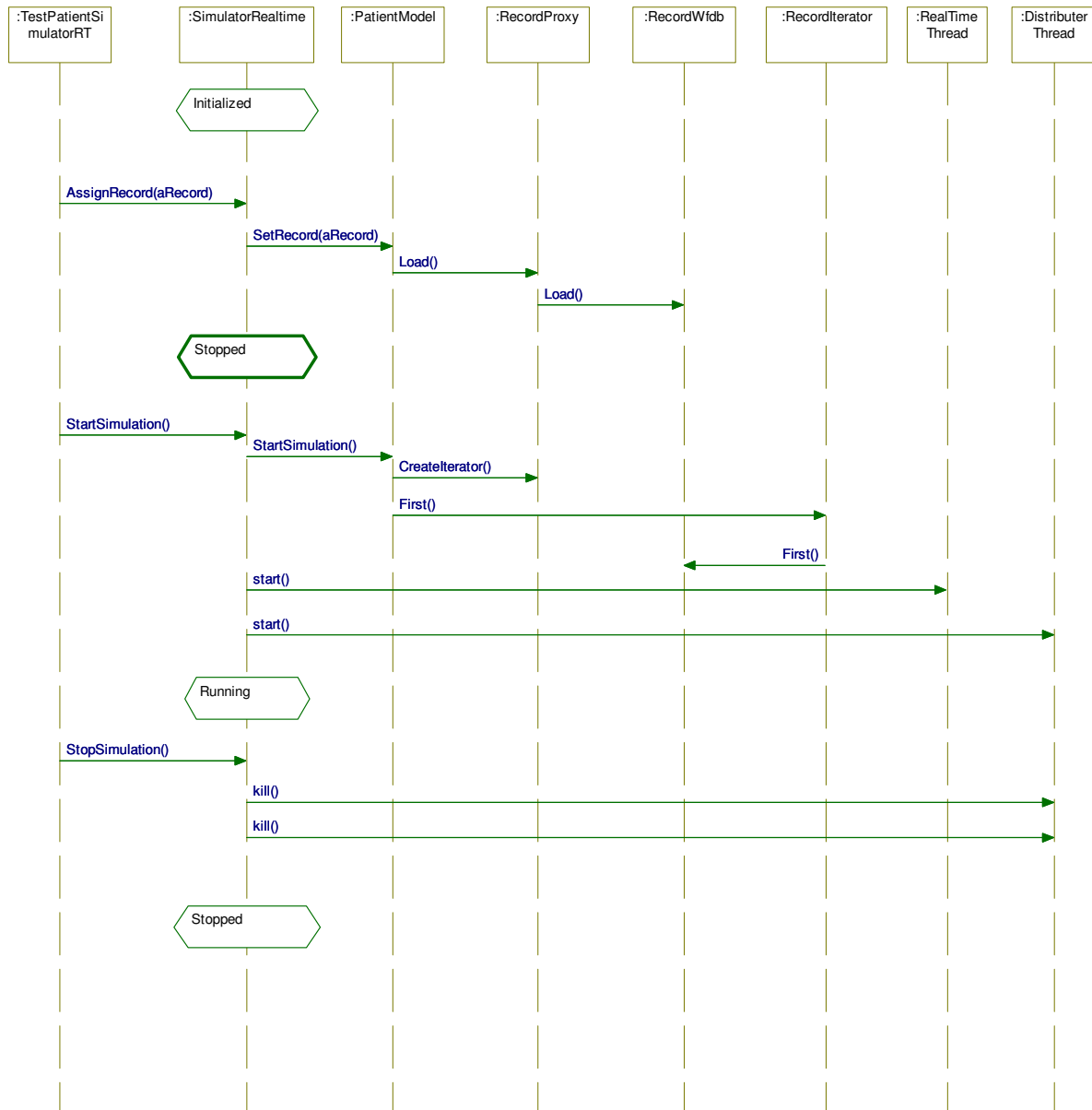


Figure 17 Sequence Diagram: Create Realtime Simulator

**Figure 18 Sequence Diagram: Create Simulation Record and Set Medicine**

**Figure 19 Sequence Diagram: Start Realtime Simulation**

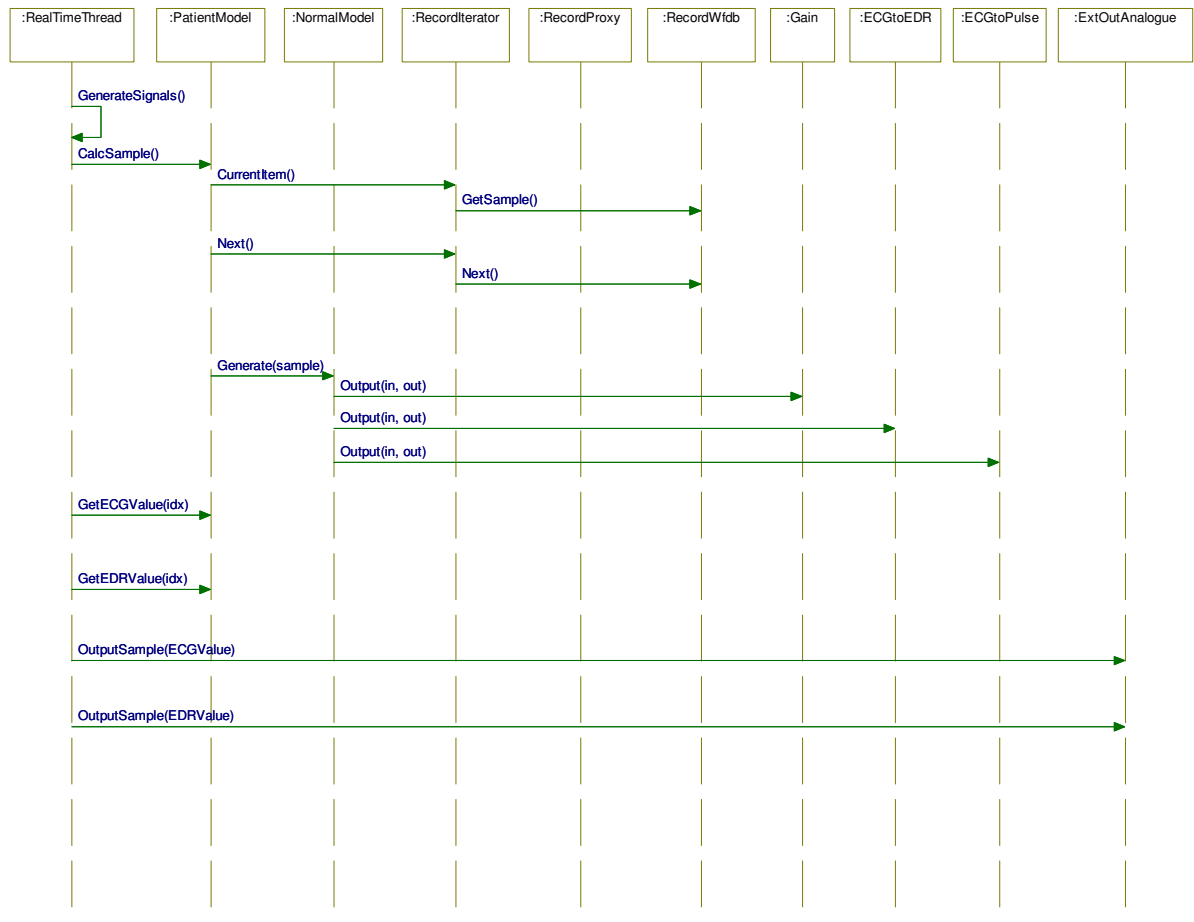


Figure 20 Sequence Diagram: RealTime thread GenerateSignals

5.3.1.1 Test setup for #UC 1 continuous package SimulatorRealtime

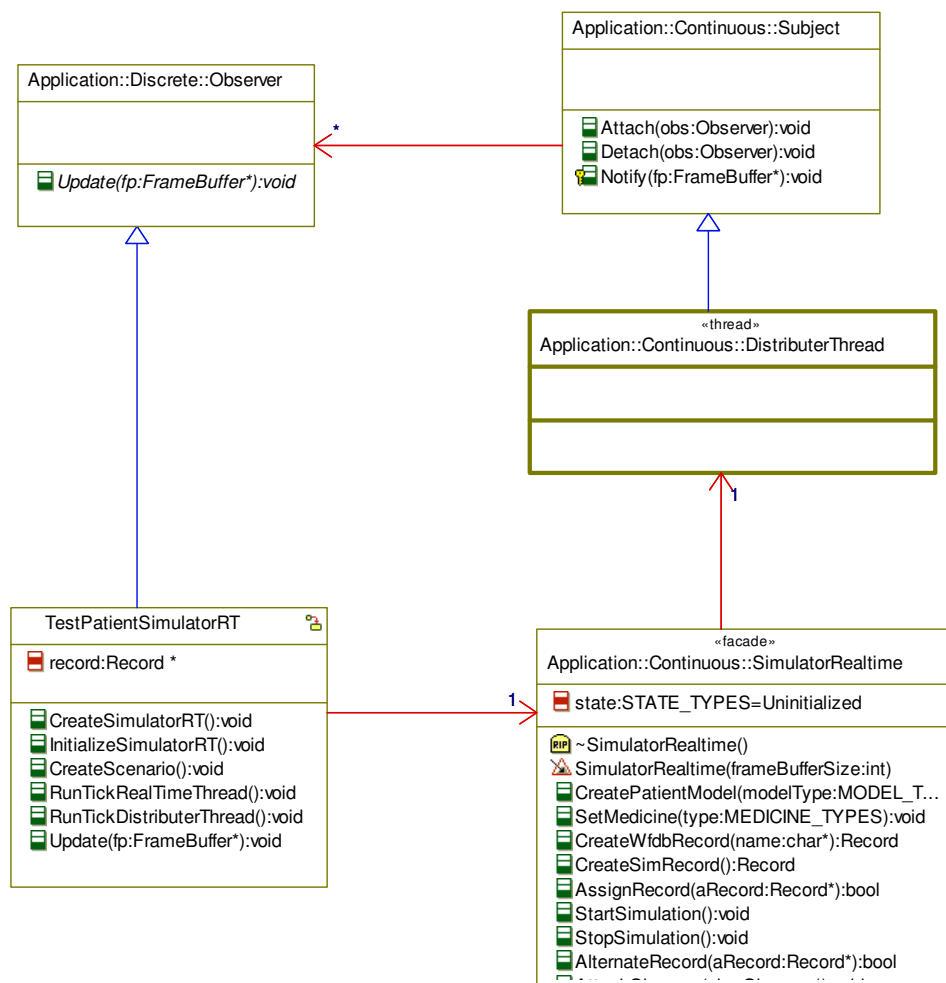


Figure 21 Test setup for UC#1 – test setup in Rhapsody only

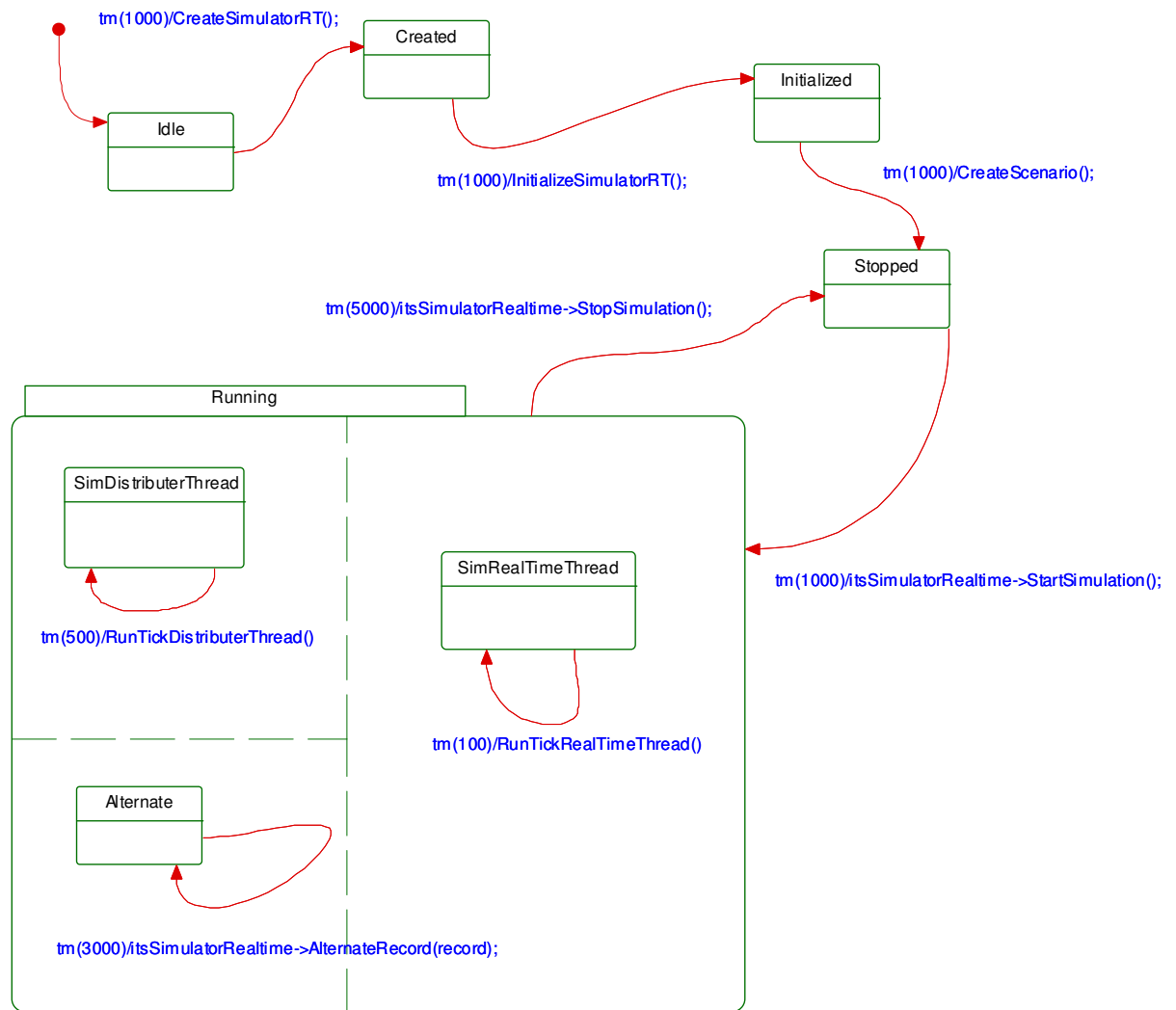


Figure 22 State machine test of UC#1 including simulation of Threads in Rhapsody

5.3.2 Use case 2. realization

6. PROCESS/TASK VIEW

6.1 Process/task overview

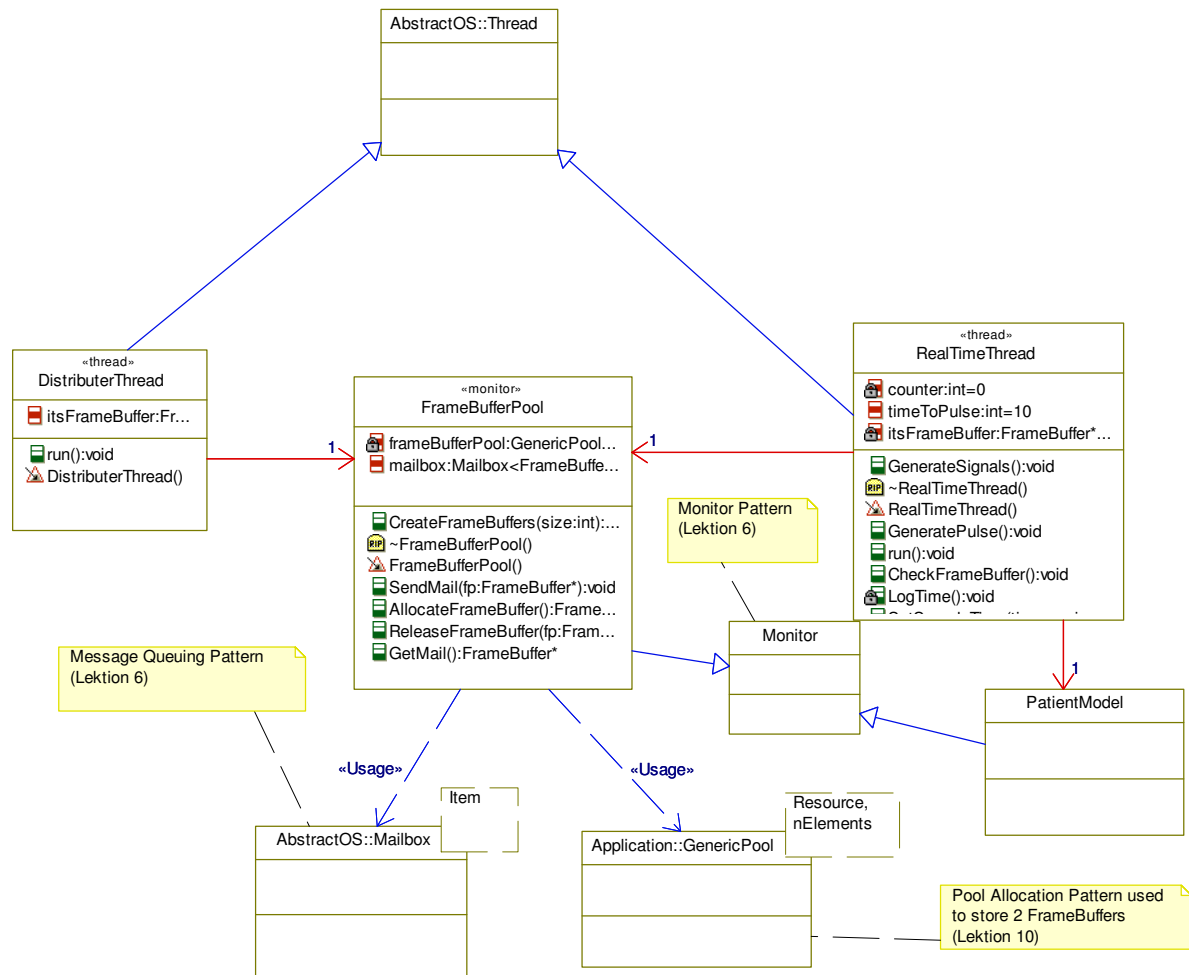


Figure 23 Process view for Distributer and RealTime threads and mechanism for synchronization

6.2 Process/task implementation

6.3 Process/task communication and synchronization

6.4 Process group 1.

6.4.1 Process communication in group 1

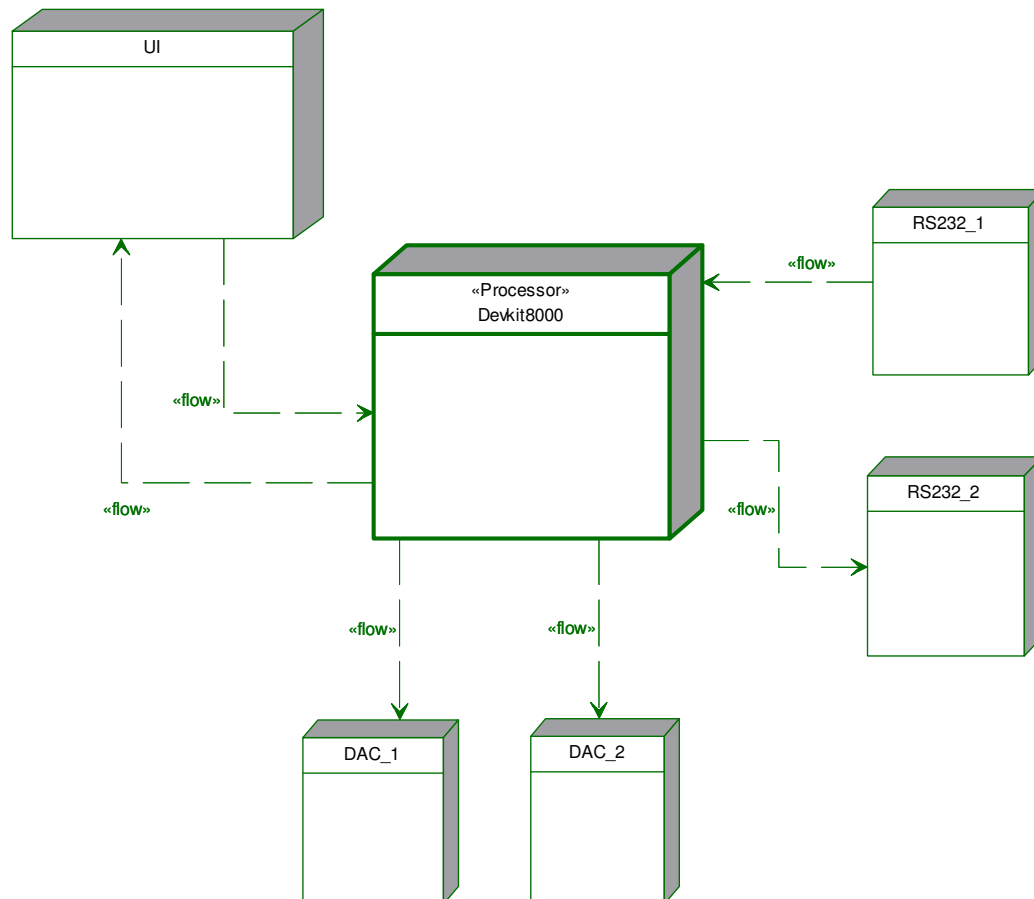
6.4.2 Process 1. description

6.4.3 Process 2. description

6.5 Process group 2.

7. DEPLOYMENT VIEW

7.1 System configurations overview



7.2 System configurations

7.2.1 Configuration 1.

7.2.2 Configuration 2.

7.3 Node descriptions

7.3.1 Node 1. description

7.3.2 Node 2. description

8. IMPLEMENTATION VIEW

8.1 Overview

8.2 Component descriptions

8.2.1 Component 1

8.2.2 Component 2

9. DATA VIEW

9.1 Data model

9.2 Implementation of persistence

10. GENERAL DESIGN DECISIONS

10.1 Architectural goals and constraints

The architectural goals are to make the system modifiable and provide high performance. Since the system is a real time system, there are constraints on how modular system can be, since modular and performance work in the opposite direction. It should be easy to extend the system new types of medicine, different kind of inputs and output. The system should be encapsulated so it's easy to port the system to different platform.

10.2 Architectural patterns

Observer Pattern:

Observer pattern help out with mass distribution of information in a system. If a number of processes are interested in the same information, observer pattern is the pattern to choose. In Sapien the observer pattern is used to distribute information generated by our simulation to the user interface. This a very common way to use observer pattern since it also notify the user interface when new data has arrived so it can be refreshed.

Five Layered Architecture:

Layered pattern organizes domains into a hierarchical organisation based their level of abstraction. The advance of layered architectural is that it make it easier to find relevant code, and make it easier to replace whole layers, for instance if you want to port the system to another device. There used open layered architectural so it's allowed to call more than one layer down.

10.3 General user interface design rules

10.4 Exception and error handling

10.5 Implementation languages and tools

Implementation languages:

- C++

Tools

- Eclipse
- TrollTech QT Creator

10.6 Implementation libraries

QT-Everywhere-4.6.1

wfdb

11. SIZE AND PERFORMANCE

12. QUALITY

13. COMPILATION AND LINKING

13.1 Compilation-hardware

13.2 Compilation-software

13.3 Compilation and linking process

14. INSTALLATION AND EXECUTING

14.1 Installation

14.2 Executing-hardware

14.3 Executing-software

14.4 Execution-control (start, stop and restart)

14.5 Error messages

15. APPENDICES