Embedded Real-Time Systems (TI-IRTS)

Project Report for Sapien 190 (PSIMU)

Spring 2010

Abstract

This project report describes the specification, design and implementation of the Sapien 190 patient simulator, simulating human physiological behavior…. (200 – 300 words)

- Presentation of the problem
- Aim of the project
- Materials and methods
- Most important results
- Conclusion

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

Table of contents

1. Preface

2. Introduction

Basic RT concepts – Lesson 1 – Basic_RT_Concepts

Event and time driven

Soft vs. hard real-time

OOA/D and patterns for real-time systems

What is a design pattern – why is it important – Lesson 1 – DesignPatternIntroduction.pdf

2.1. Report structure

3. Project Description

3.1. Project context

Describe the patient simulator project boundaries – input from specification

3.2. Project execution

In the overall planning and execution of the Project we have been inspired of the ROPES[1] Development Process and Scrum. In this chapter we will describe how we have used parts ROPES and Scrum in planning and execution of the project.

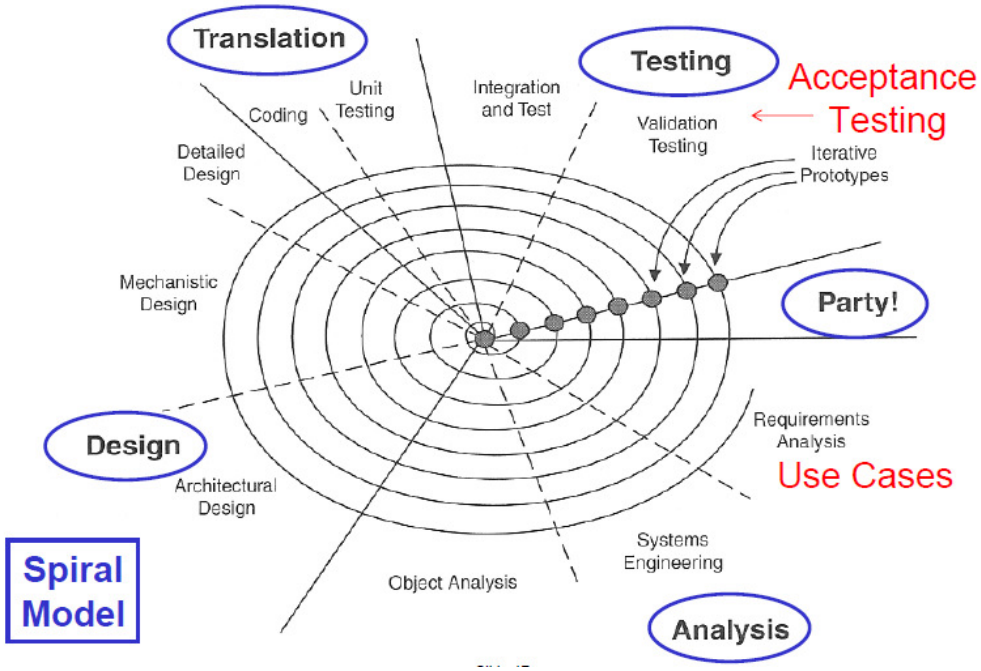

Figure 1 ROPES Spiral Micro-cycle (Detail)

[1] Rapid Object-oriented Process for Embedded Systems described in [2] chapter 3

The ROPES spiral micro-cylce model defines a number of phases to be executed for an iteration of a new prototype release. The purpose is to learn how the prototype performs being able to improve the design and implementation for every iteration. It also allows us to extend the functionality of the product in smaller steps and allows us always to have something to demonstrate. It starts with the basic requirements defined by use cases that have a significant impact on the architecture design of the product. After a number of iteration we will end up with the final product by implementing more and more use cases and functionality for each iteration. We have used the ROPES methodology for the steps we have followed for each prototype in the Sapien 190 project. We have produced a number of prototypes with different purpose to investigate the platform, technologies and finally the implementation of the functionality for the product.

In the analysis phase we have started with the requirement analysis by delivering a use case specification[2] for the Sapien 190 product to our "Customer = Teacher". Here we have identified a number of actors and use cases a shown in Figure 2.

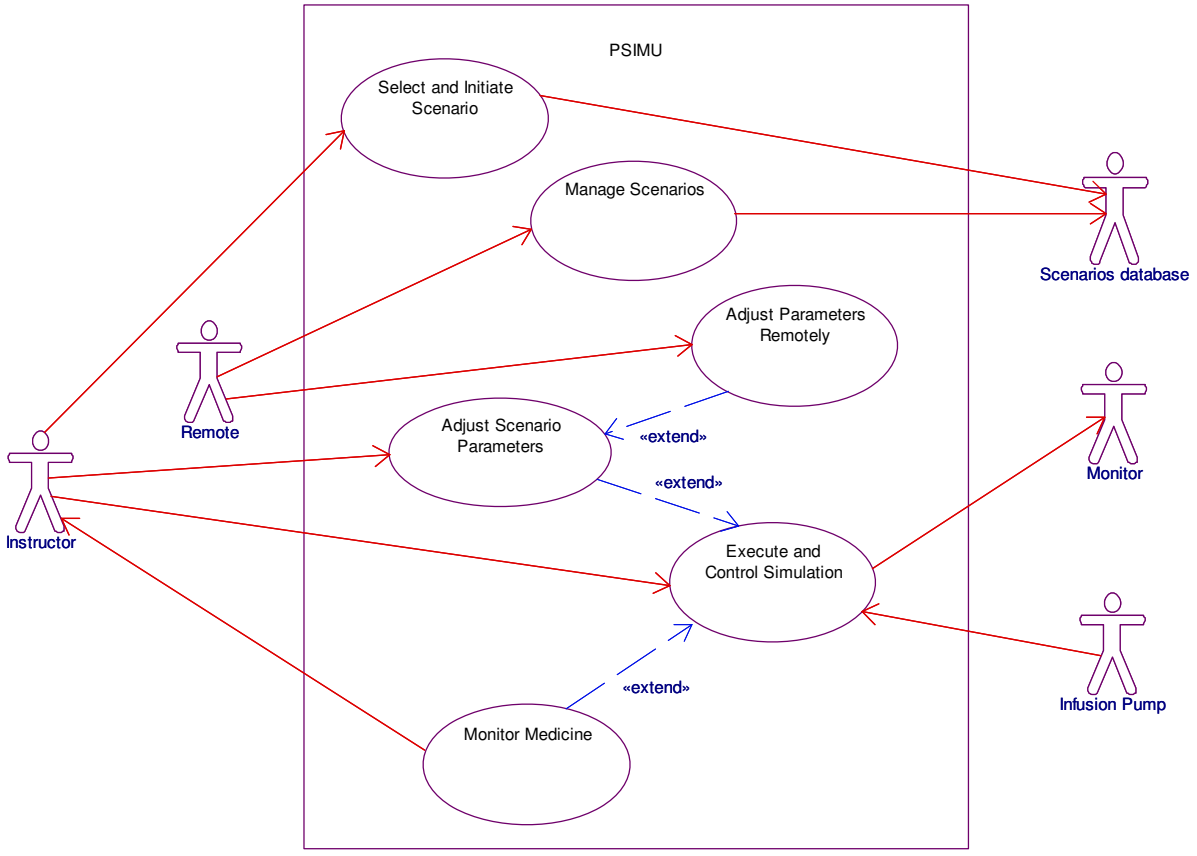

Figure 2 Use case specification and analysis

The next step was to make a domain model for the use case "Execute and Control Simulation" and complete the first iteration in the ROPES spiral micro-cycle. The use case "Execute and Control

[2] See use case specification for Sapien 190 in references [5]

simulation" is the most complex and significant for the architecture of the real-time part of the patient simulator. This fist official prototype we have delivered to our "Customer" together with a first version of the architectural documentation and a demonstration of the Sapien 190 patient simulator. In the W-Model[3] by Alistair Cockburn he defines external visible deliveries for the external stakeholders of a project being able to follow the progress and give feedback to the project.

First Delivery 11. May 2010

- Updated Requirement Specification
- Draft Product Architecture Document
- Status Report
- Simulator prototype first version (Part of Use Case #1)

Final Delivery 4. June 2010

- Requirement Specification
- Product Architecture Document
- Project Report
- Simulator prototype (Use Cases parts of #1, #2 and #3)

[3] Slide 43 from Lesson 8 – L3_ROPES_process

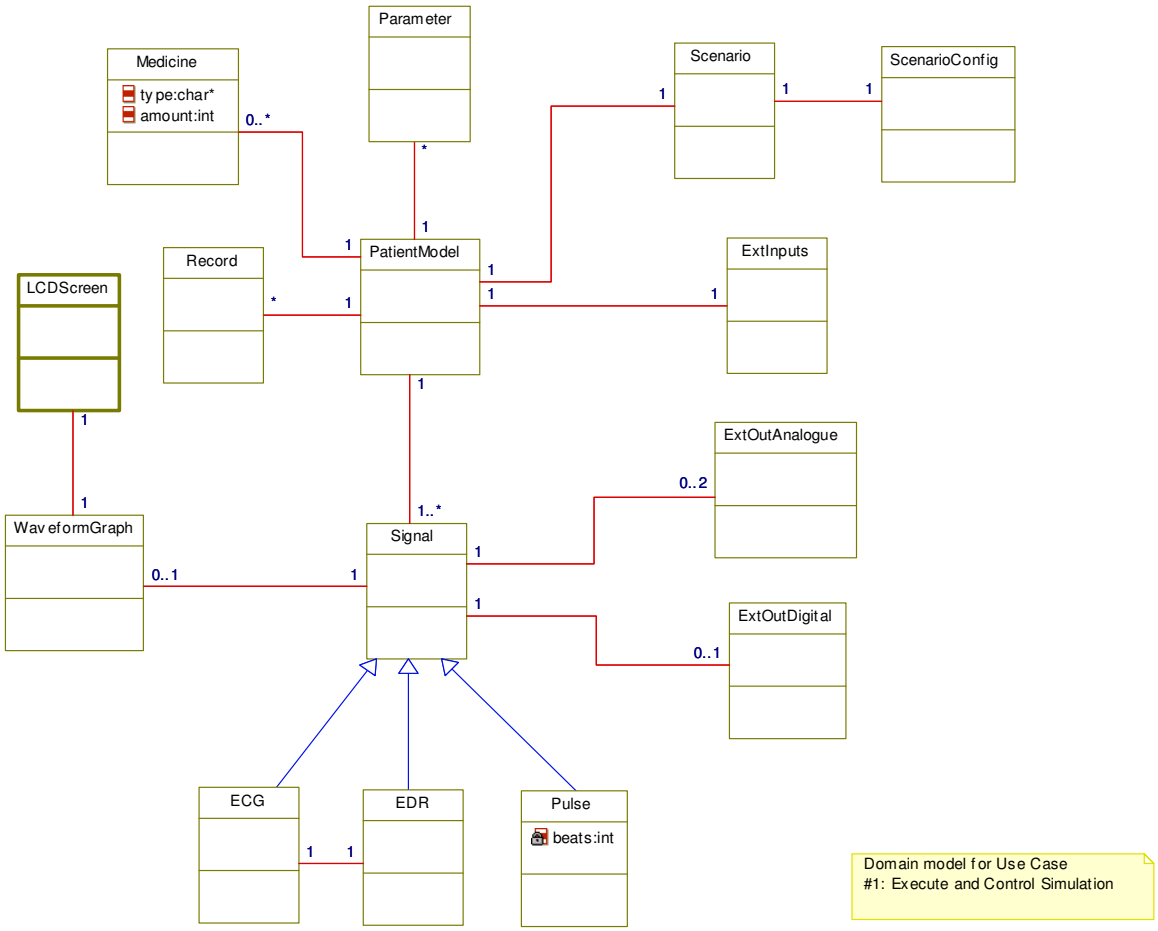

Figure 3 The first domain model for UC#1 Control and Execute Simulation

At the same time using the ROPES micro-cycle spiral to develop the actual product we did make a number of experimental prototypes to reduce the number of unknowns and risk in the project. Some of the questions we had in the beginning were:

- Would it be possible to generate the analogue ECG signal from user space in Linux on the target platform with the sampling rate of 250Hz?
- What would the CPU load be?
- How to plot the waveform ECG signals on the LCD screen using Qt?
- How to cross-compile the patient record WFDB library to the Linux target platform?

The first prototype was to reduce risk by investigate how to use the embedded Linux platform in reading patient record on the target and output the analogue ECG signal. The second prototype was to investigate how to develop with Qt and creating a waveform ECG graph based on the patient record readings.

The following prototypes has been about producing functionality according to the use case specification, where for every iteration parts of a use case or more use cases has been included. These prototypes has been developed by a combination of automatic code-generation from a Rhapsody UML model and source code manual written for the HW and OS abstraction layers. The GUI has been written using Qt from Nokia and integrated with the manual code and code generated from Rhapsody.

ROPES describe the key enabling technologies like visual UML modeling, model execution and mode-code associativity. These key technologies has been possible to realize in this project by use of the IBM Rapsody UML design tool not only for drawing UML diagrams, but also for high level modeling and automatic code generation. This approach has enabled the development process to focus on design instead of implementation. The ROPES key technologies has been a help to make fast iterations between testing new design ideas, by animation of state chats, setting breakpoints in Rhapsody and inspecting variables and states in the model. This higher level modeling approach compared to normally coding, debugging and testing has turned the development process to be more design focused than traditional development. We have used the animated sequence diagrams for test documentation and generating the code directly to Linux and the target platform, which has saved us for time fixing typing errors and trivial debugging and test.

The Rational Unified Process (RUP) has a process workflow that specifies a number of phases: Inception, Elaboration, Construction and finally Transition. In each phase a number of iterations are perform like in ROPES. In the beginning focus is on the process workflow requirements and business modeling. In this project we have primary been working in the elaboration phase working with 2 major iterations of product release. Since focus for this course in some extend has been design patterns, we have use most the time in the elaboration phase with focus on the architectural and mechanistic design of the project.

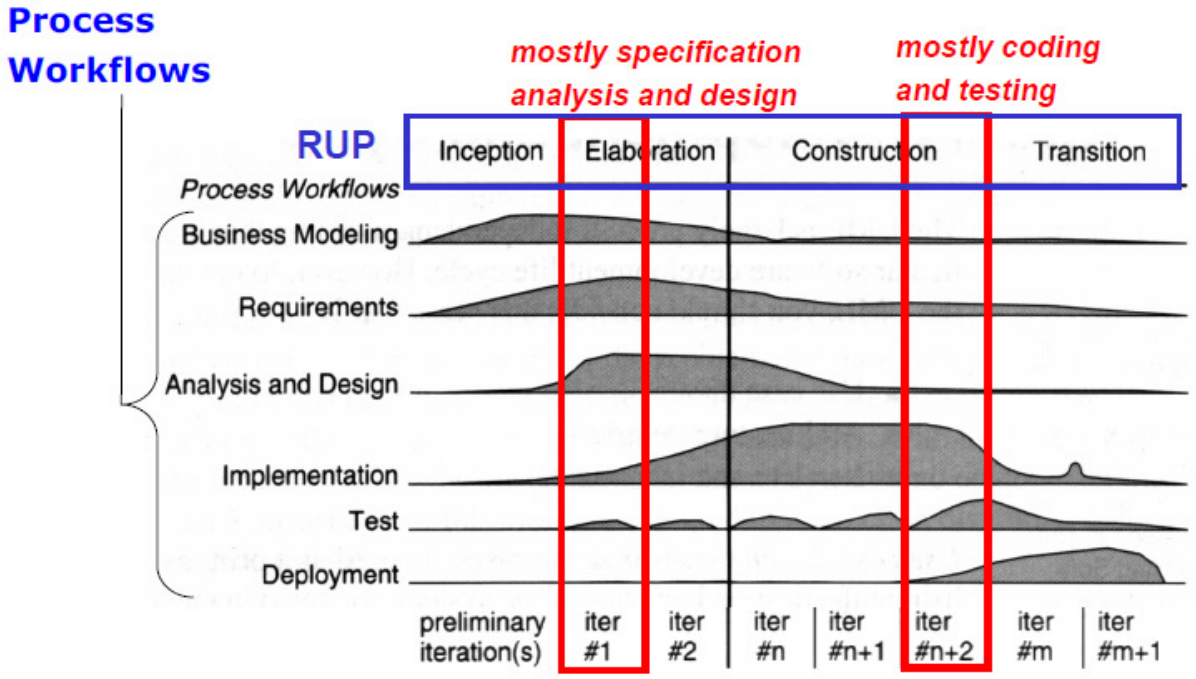

Figure 4 Rational Unified Process workflows

In controlling the progress of the project we have had regularly meetings in which we have been inspired from the way Scrum meetings are organized. In every meeting we have used time on each project member giving a short status of what has been done, issues and problems discovered since last meeting based. The status is based on the planned activities from last meeting. An updated list with notes of meeting status has been updated for every project meetings see appendix A. On every meeting the backlog in Scrum terms has been updated according to our eventually changes priority. We did not define a scrum master or product owner, but on every meeting one of the project members did take the lead on updating the status list and together we did a prioritation of the backlog (Pending activities). Below see example for the contents of meeting status.

Scrum Status 25. May:
_____

<Name>:
Done:
- Meeting minutes
- Added text to Chapter 5

Problems:
- What exactly to put into ch 11+12

Next:
- Architecture document chapter 11. and 12.

Next scrum meeting Tuesday at 9:00
- Scrum status
- Status on writing Architecture document
- Status on report writing and assignment of tasks
- Continue to 16:00

Action list (Backlog) to final delivery 04. June:
- Finalize Product Architecture Document
- Finalize Project Report
- Implement ECG to Pulse filter
- …..

Notes to be deleted:
Scrum meetings and minutes – Lesson 8 – Scrum short
Schedule – Specification
Part of ROPES – Lesson 8 – L3_ROPES_Process
ROPES Microcycle for UC#1 (3-4 weeks for D1)
Iterations see specification D1 and D2
Prototyping (First prototype with technology clarification)
Model Base Development (Testing design in Rhapsody)
Risk analysis – see minutes notes (NotesMeeting1904_2010.txt)
Fast prototype to reduce risk of new technology
WFDB patient record library on target
HW interfacing on target and load
GUI programming with Qt

3.3. Method for analyze work

In the requirement analysis phase with reference to the ROPES spiral model we have created a number of use cases to define the required functionality of the product. A detailed specification of this work can be found in "Requirement Specifiation for Sapien 190" [5]. The first step has been to define the actor-contex diagram to identify the actors of the patient simulator. Here we have used the specifications of the PSIMU, LMON, IPUMP and interface specification referenced in [5] chapter 1.2

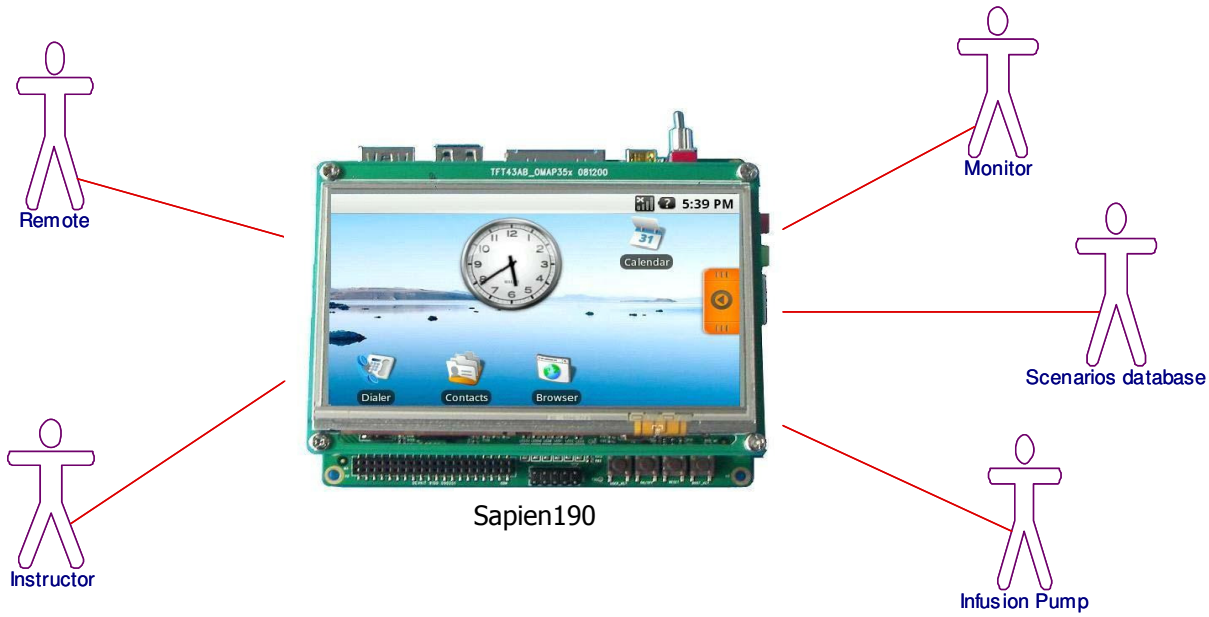

Figure 5 Actor-Context Diagram

The second step in the use case analysis was to identify a number of use cases for each actor as illustrated in use case diagram Figure 2. These use cases is described in details in the identified to be:

1. **Execute and Control Simulation**
2. **Select and Initiate Scenario**
3. **Adjust Scenario Parameters**
4. **Monitor Medicine**
5. **Manage Scenarios**

For each use case specification a main scenario and extension is described like on page 9 in the "Requirement Specification for Sapien 190" [5]. This document contains the textual description of the system context, interface to external actors, non functional requirements like performance, prioritized product qualities and design constrains. Finally the document contains a description of the original planned deliveries of use cases to be contained in the two first deliveries to our "Customer". We have prioritized maintability, correctness and usability as the most important quality factors for the patient simulator. The simulator application is expected to have a longer life time than the hardware and must be able to be maintained for many years in the future. Therefore it is important for the product to focus on these factors in creating the architecture and design.

The domain analysis is based on the use case requirement specification where we did start with the use case "Execute and Control Simulation". This approach is defined in the Unified Process[4] (UP). A domain model is created to identify the conceptual domain classes and relations between them to give us a better and deeper understanding of the actual problem. We have chosen the "Execute and Control Simulation" use case to be the first in creating this domain model since it is the most complicated and contains the essential functionality for the patient simulator. The UML domain class diagram is illustrated in Figure 3. To this first domain model we added boundary and one control classes. The purpose of the boundary classes is to separate the model from the external actors and the control class is used to encapsulates the control of the scenario described in the UC#1.

[4] Craig Larman, Applying UML and Patterns, Third Edition chapter 9 Domain Models

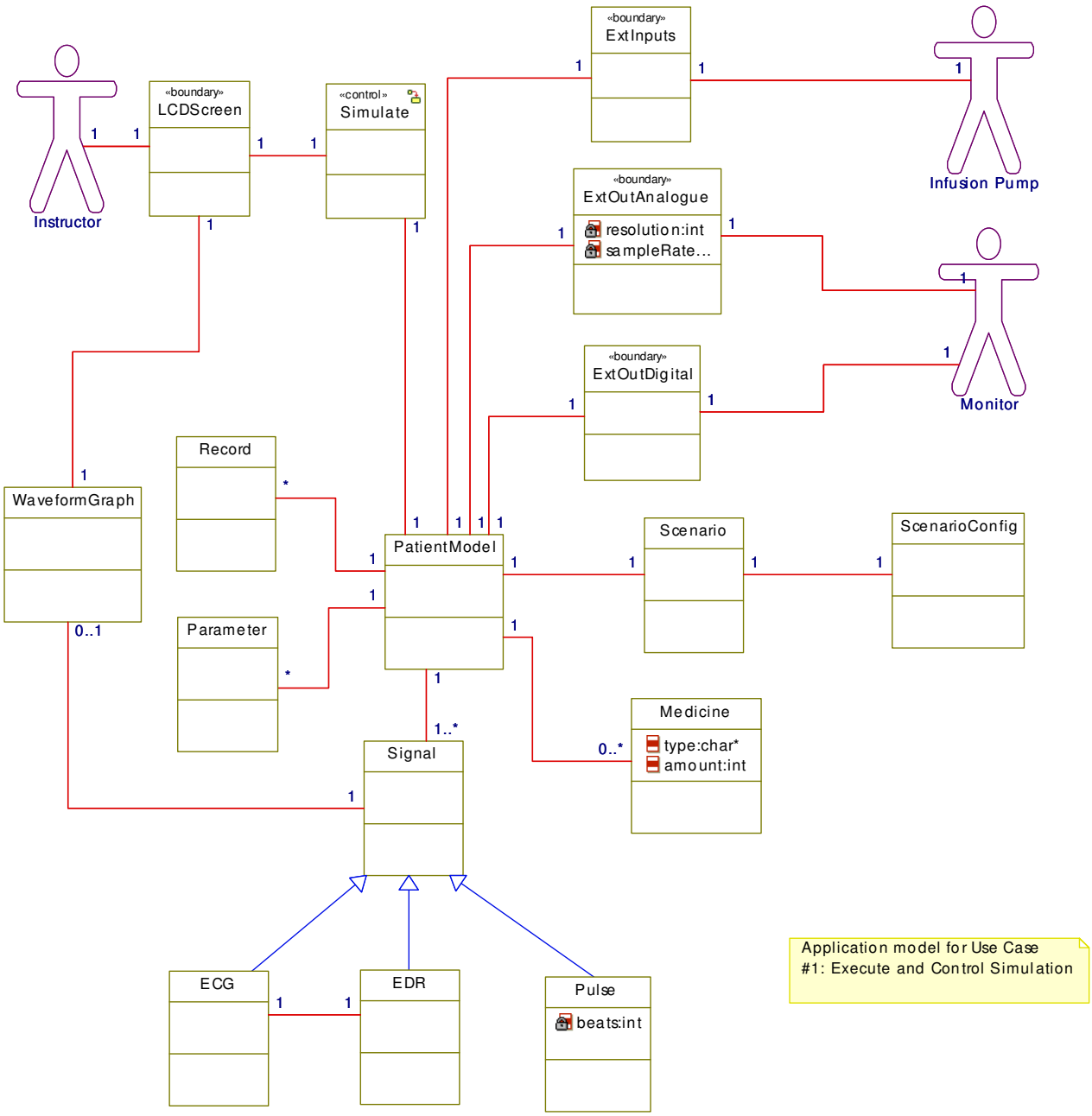

Figure 6 Domain model with boundary and control classes

In the domain model for this first iteration we have identified all the important classes and the relations between them. The domain model is saved in a separate Rhapsody project since the domain model in the following design iterations is changed a number of times. The domain model illustrates how the patient model is the central class for coordinating the patient simulation. When the patient simulation is running it will perform reading of the digitized physiologic signals from a record and send theses "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. The basic idea is that the instructor later should be able to select between different scenarios that describes the configuration of a simulation which include a certain patient model (Normal or with Infusion Pump), patient records from the PhysioBank[5] database and parameter settings for the simulation like gain and rate for replaying the patient record. More details can be found in the requirement specification [5]. In the UML state diagram below we have added a simple functionality for the instructor to start, stop, pause and resume the patient simulation with a fixed scenario configured for the final prototype. That means UC#1, UC#2 and UC#3 is implemented without being able to select a scenario in UC#2.

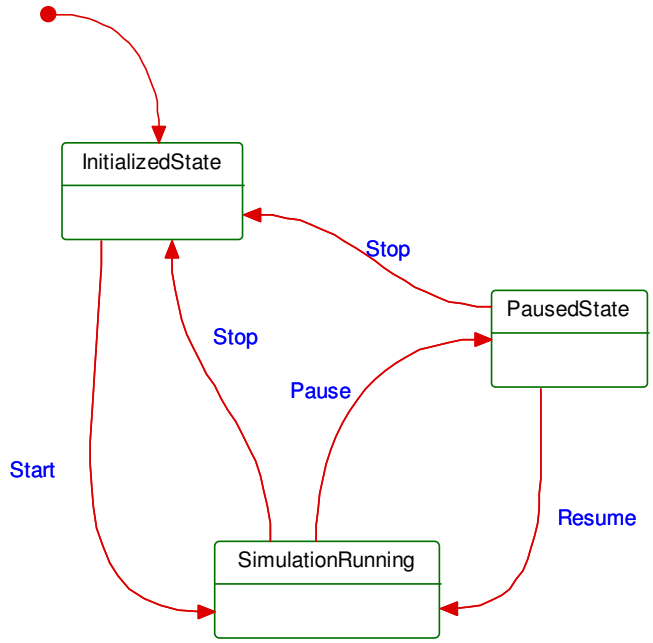

Figure 7 State chart for domain control class simulate

Part of the analysis phase we have identified a number of external and internal generated events that plays an important role in constraining and defining the system behavior. The events listed below are crucial for analysis of the schedulability and deadlines later in the design process (RMA analysis). For this analysis we have defined number of parameter that can be adjusted to find the optimal timing and scheduling being able to generate the ECG, EDR and the pulse signal. At the same time we should be able to processing information from the IPUMP and updating the waveform signal graph on the LCD screen. A frame buffer has been defined for the number of samples to be collected before updating the LCD screen.

[5] http://www.physionet.org/physiobank/physiobank-intro.shtml

Internal and external event list

#	Event Id	System response	Arrival Pattern	Event Source	Response time
1	Sample	Calculate and generate EDR and ECG signals	Frequency of 250 (Fs) max 400	Internal timer	Less than sample periode
2	Pulse	Caculate pulse every 200 (Np) samples	Fs/Np	Internal timer	Less than sample periode
3	PDU	Updates medicine information	Every second (Fi)	IPUMP	Less than ½ second
4	FrameBuffer	Updates signal graph on LCD display	Every 50 (Nf) samples (1/8 of LCD display in pixels)	Internal timer	Less than period updating framebuffer

Parameters identified to be used for RMA analysis in design phase:

	Time units	Default value	Units
Fs	Sample frequency	250	Hz
Np	Num samples for pulse calculation	200	Number
Fi	PDU frequency	1	Hz
Nf	Frame buffer size	50	Number

Notes to be deleted:
Analysis method (Identification of the essential characteristics of possible correct solution)
Use Case Specification and analysis work
Use case UML domain model analysis
Event analysis

3.4. Method for design work

We have used the 4+1 view designed by Phillipe Kruchten[6] as the process driver for the design work in the project. The Use Case view is the driver for focus on what to design in the other 4 views as show in the figure below. We have started with UC#1 defining a number of scenarios that is used as the foundation for the architectural, mechanistic and detailed design that is described in details in the "System/product architecture document for Sapien 190" reference [4].

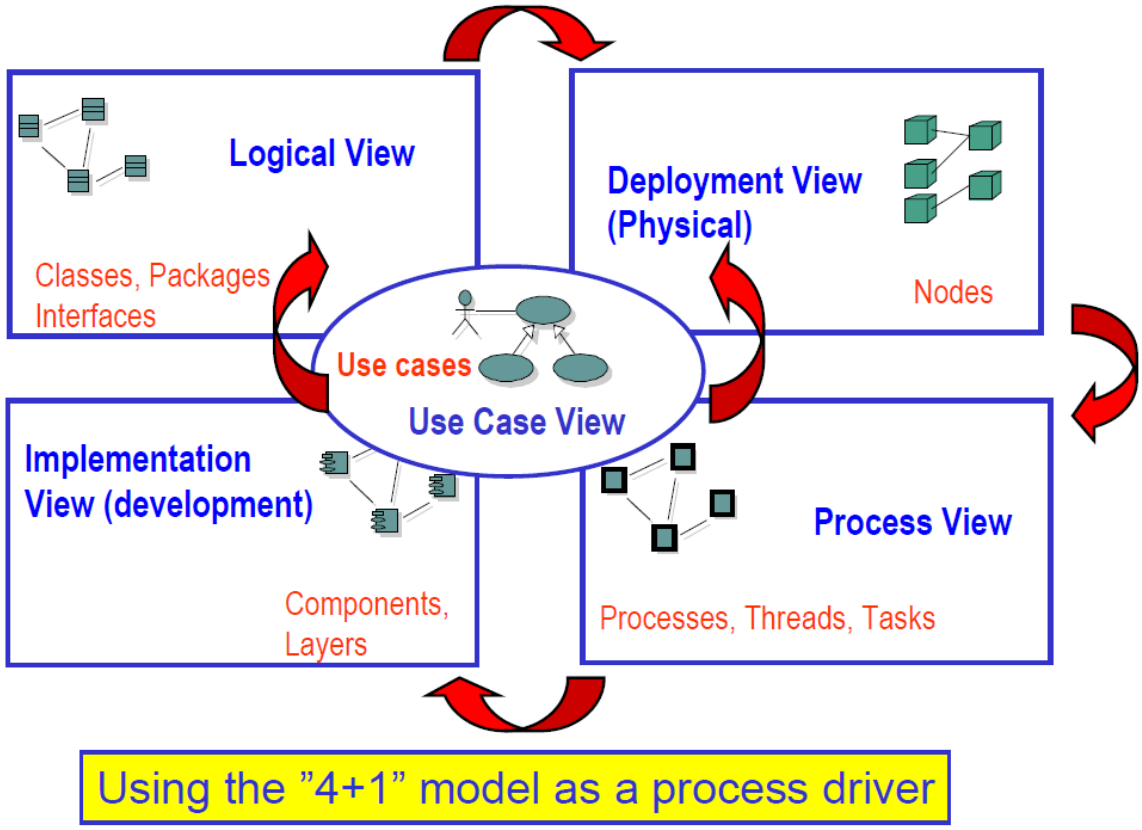

Figure 8[7] "4+1" view model

The UC #1 has been selected for the first iteration of the ROPES spiral microcycle in making the architectural, mechanistic and detailed design. This use case is significant and provides the central functionality of the patient Simulator and contains the real-time constraints. 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)

[6] Kruchten, Phillipe, Architectural Blueprints – The "4+1" View Model of Software Architecture
[7] Slide 4 from Lesson 8, DevelopmentOfRealtimeSystems

- Display of signal waveform using Qt on target (Working with Qt on target

The deployment, logical, process and deployment views are designed according to the steps described in development of real-time embedded systems[8]:

Step 2: Find and document use case #1, #2 and #3 (Chapter 4. and 5.3 Ref. [4])

Step 3: Select a suitable HW architecture (Chapter 7. Ref. [4])

Step 4: Develop a logical model (Chapter 5. Ref. [4])

Step 5: Select a concurrency model (Chapter 6. Ref. [4])

Step 6: Design, Implement, Test and Measure (Chapter 6.5 Rate Monotonic Analysis Ref. [4])

The architectural design is based on the five-layer architecture pattern described in chapter 4.2 reference [2] illustrated in the package diagram shown below.

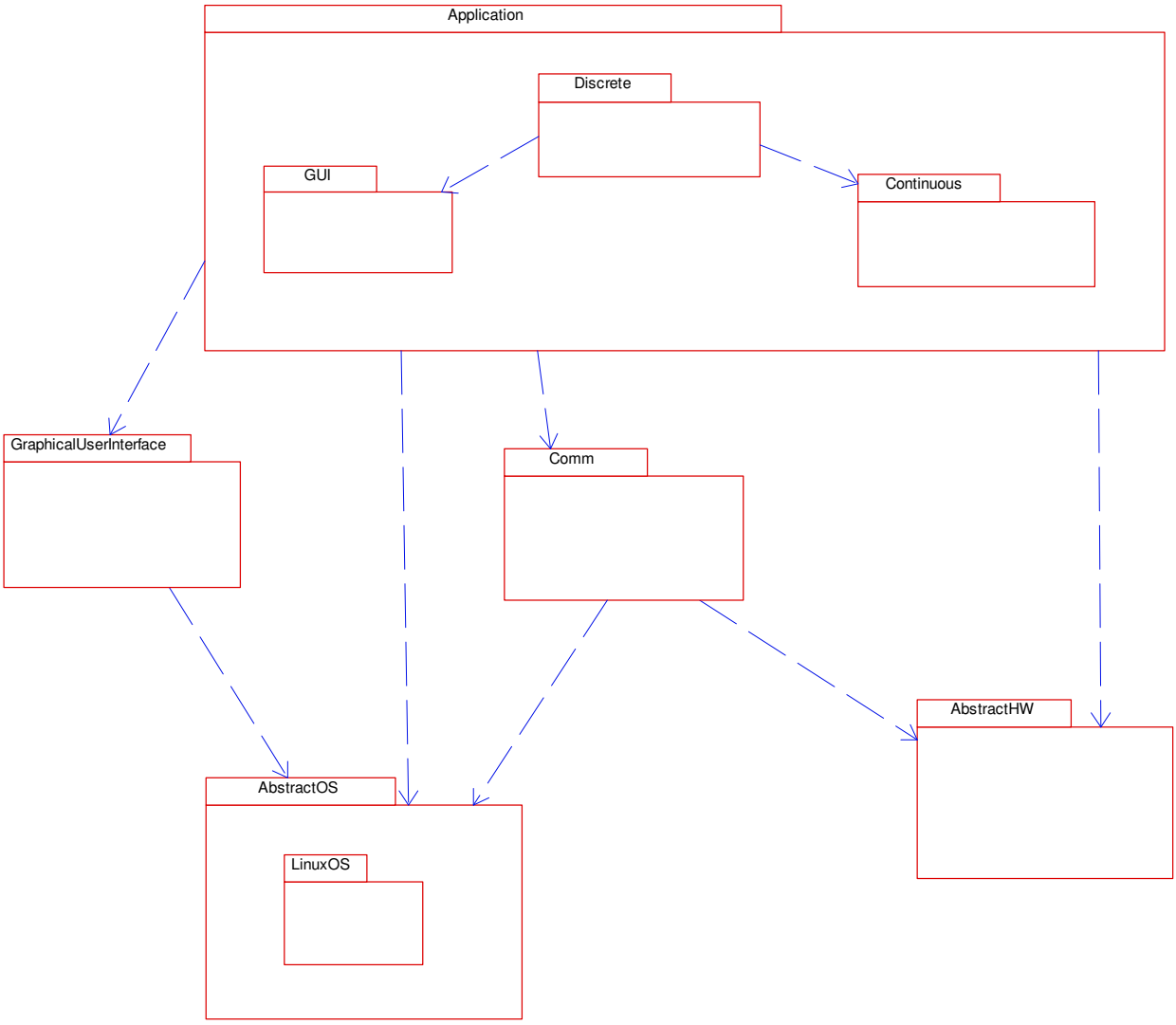

Figure 9 Five layred architecture for logical view

[8] Slides 1 – 17 Lesson 8, DevelopmentOfRealTimeSystem

Each abstraction layer is a logical layer representing a well defined domain. Dividing the system into several layers ensures high cohesion for each domain and low coupling between the domains. This simplifies the process of modifying our design or extending it. The application package uses the two-part[9] architectural model for the discrete and continuous parts of the patient Simulator. These packages contains also the most complex part of our design. After having defined the overall architecture of the design we have focused on mechanistic design of the continuos and discrete package by refining the domain model in where we have introduced a number of creational, strutural and behaviroal design patterns from the GoF[10] book.

The list below briefly summarize the patterns that has been used in the design:

- Observer – to notify the UI with new a new frame buffer of samples
- Proxy – to provide a place holder for reading records (Local or remotely)
- Iterator – to provide a way to access samples in the records without exposing the underlying representation
- Mediator (PatientModel)
- Strategy (PhysioModel)
- Filters and pipes (Caclulations)
- Factory method pattern (Creating filters)
- Façade (Simulator Realtime)
- Command (Setting parameters)
- Monitor (PatientModel and FrameBufferPool)
- Message Queuing Pattern (Mailbox)
- Pool Allocation Pattern (FrameBuffers)
- Command (Parameters and in combination with state)
- State (Discrete)
- Singleton (DAC + Command State pattern)

Notes to be deleted:
Design process for development of real-time systems – Lesson 8 - DevelopmentOfRealtimeSystem.pdf
Design method (Adds elements to the analysis the defines one particular solution)
Ropes design activities – Lesson 5 – ArchitectureAndUML.pdf – slide 20
Architectural
Mechanistic
Detailed
Architecture – Five layered and two layered patterns
UC# 1 most important for the architecture discrete and continuous
Two layered and five layered based on UC#1
Alternative solutions …..
Mechanistic design using patterns from course

[9] Hans Peter Jepsen, Finn Overgaard Hansen, Designing Event-Controlled Continuous Processing Systems
[10] Gang of four, Erich Gamma and co., Design Patterns, Elements of Reusable OO Software

Detailed design by adding methods and using sequence diagrams for UC

Description of the design process and an evaluation of this.
Why we havn't used ports – complicated to implement (Lesson 5 – DesigningWithPortsInUML2.pdf)
Open-Close Principle where in design patterns it is used (Lesson 5 – GeneralDesignPrinciples-2.pdf)
Liskov Substituion Principle (LSP) (Lesson 5) –Strategy pattern is a good example PhysioModel
Arguments for the design choises:
Mediator for PatientModel and Medicine – to be extended
RMA even analysis on the screen update and sample calculation (RealTimeThread and DistributerThread and GUI) – see Lesson 6 – ThreadsAndSchedulability
Scheduling policy – see Lesson 10 – Critcal Section Pattern used.

3.5. Translation and testing

Test design in high level modeling with Rhapsody
Test design integrated with Qt on Linux
Translation
Creates executable deployable realization of the design
Automatic code generation from Rhapsody
GUI programming in Qt
HW and OS Abstraction manual programming

3.6. Development tools

Qt, Eclipse, Linux, WFDB, Rhapsody, Crosscompiler, DevKit8000, Cygwin
See ArchiteturalDocument chapter 13 + 14
File structure and how tools have been integrated.

3.7. Results

Objective -
Results in short from – screen dumps – Qt graph – printing samples
Top on target
Measurements of DAC outputs with scope

3.8. Discussion on achieved results

Comments with own opinion

3.9. Experience obtained

See guide

3.10. Excellence of the project

See guide

3.11. Suggestion to improvements

- Project – we should have started with design from ex 1-5
- Product – to be completed – design patterns that could improve and been implemented
- Execution time on target – speed

4. Conclusion

Learning outcomes and competences:

The participants must at the end of the course be able to:
– ***analyze and describe*** the requirement for an
embedded real-time system
– ***design and constructs*** an architecture for an
embedded real-time system
– ***evaluate and apply*** design patterns in development
of an embedded real-time system
– ***prepare*** a product documentation for an embedded
real-time system using UML

See guide

Eg: "We have realized in this project that modelling is a powerful tool in helping us to **evaluate different ideas** applied to the problem being studied. While analyzing the situation we came up with several strategies that have been modelled using UML and VDM++. It was possible to determine whether the solutions to the problem were a correct approach or not and the advantages and disadvantages they presented"

5. 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] …

[4] Product architecture document for Sapien 190

[5] Requirement specification for Sapien 190
http://code.google.com/p/iirtsf10grp5/downloads/detail?name=Sapien190Spec.doc&can=2&q=