Object-Oriented Integrated Approach for the Design of Scalable ECG Systems

Dusanka BOSKOVIC 1, Ingmar BESIC, Zikrija AVDAGIC
Department of Computing and Informatics, Faculty of Electrical Engineering, University of Sarajevo, Bosnia and Herzegovina

Abstract. The paper presents the implementation of Object-Oriented (OO) integrated approaches to the design of scalable Electro-Cardio-Graph (ECG) Systems. The purpose of this methodology is to preserve real-world structure and relations with the aim to minimize the information loss during the process of modeling, especially for Real-Time (RT) systems. We report on a case study of the design that uses the integration of OO and RT methods and the Unified Modeling Language (UML) standard notation. OO methods identify objects in the real-world domain and use them as fundamental building blocks for the software system. The gained experience based on the strongly defined semantics of the object model is discussed and related problems are analyzed.

Keywords. object-oriented design, ECG system, real-time system

1. Introduction

The importance of detailed design prior to implementation grows with the complexity of systems, thus becoming vital for understanding the system and for the appropriate system development. To achieve the full benefit of using OO technology, it must be incorporated into all stages of the software engineering process. System development is the process of adding details to that model until it can be executed on a computer [1].

OO was successfully applied in non-RT-applications, with the following benefits: a consistent model throughout the software engineering process, close mapping of the real-world problem, ability to modify, extend and maintain software, software interoperability and reusability. In RT systems the concepts of timing, concurrency and hardware/software interfaces are the main issues faced in the design process. Early evaluation to ensure that proper functionality can be achieved within the established time frame is crucial for successful development. The UML defines a standard of notation for documenting OO analysis and design, but does not specify a process for software development. This paper describes the integration of OO design of RT systems with the UML standard notation, using the Multi-Lab/2-based RT station for implementation [2].

The ECG system to be designed and afterwards simulated is described in section two. Section three discusses the functional requirements and the system analysis resulting in the fundamental UML diagrams. The approach made application development from the design model easier and more efficient.

1 Corresponding Author: Dusanka Boskovic, Mr.Sci., Faculty of Electrical Engineering, University of Sarajevo, Zmaja od Bosne bb.; Tel: +387 33 250 700; E-mail: dboskovic@etf.unsa.ba.
2. System Description

In this section the intended structure of the ECG system is described. The main objective is to design the core ECG system, which can then be used as the foundation for building different ECG devices for use in hospitals or in the field. Such devices can differ in number and the type of connected ECG channels, persistence of the captured ECG signals, available reviewing modes and ECG data analysis. The maximum set of features would include:

- Monitoring of the ECG waveforms – direct monitoring of ECG signal or preparation for recording – evaluation of the signal quality, calculating average pulse rate.
- Recording of the ECG waveforms.
- Reviewing mode – scrolling through recorded ECG charts; patient data administration and reports printing.
- Administration mode – services for patient data administration, including ECG charts.

Apart from standard or specially-developed user interfaces, the system should be equipped with hardware modules for capturing analogue signals, a standard or specially-designed monitor and a device for creating a hard copy of the signal (printer or plotter).

The heart is built of specialized tissue that has the unique property of rhythmically emitting electrical impulses. We can see the impulse as it crosses the heart by measuring the electrical current with the electrodes placed on the patient’s skin. The normal ECG complex consists of three key elements:

- the P-Wave, representing the impulse across the atria to the A/V Node;
- the QRS complex, representing the impulse as it travels across the ventricles;
- the T-Wave, representing the repolarization of the ventricles.

The ECG signal varies in voltage from 0.25 to 3 mV, with a signal bandwidth between 0 and 120 Hz, thus the sampling frequency should be at least 240 Hz. By noting the shape, consistency, and the time between these waveforms we can learn more about the conduction system of the heart, identify damaged areas and areas that are not receiving enough oxygen, and thus establish proper diagnosis or monitor the patient’s condition. Stored ECG waveforms can be used to train medical personnel to read ECGs and for statistical analysis of medical data.

3. Object-Oriented System Analysis

3.1. Requirements Analysis – Use Cases

The functional requirements analysis defines the desired functionality of the system. RT systems are in constant interaction with their environment, so the boundaries and interactions between the system and the external world along with time constraints are the basis for the requirements analysis. The primary system functions are identified and connected to the external ‘actors’ – users and devices. Interactions between actors and systems are established in Use Case (UC) diagram, supported by the UML standard.
UC diagrams show the main instances of interaction of the system and external actors. For a general ECG system four external actors are identified: the physician, the patient, the monitor and the hard copy device. The patient is the primary data source and the physician is the data recipient and initiator of the events. The other two are data output devices: ECG monitor and the hard copy device for ECG chart creation. The ECG system has six different Use Cases: Configure System, Administer Patient, Capture ECG, Monitor ECG, Store ECG, Make a Hard Copy. Associations between actors and Use Cases behave according protocols to be defined. Further analysis of object behavior will allow for the creation protocols using message sequence diagrams.

3.2. Events and Scenarios

In reactive systems, by definition, the primary function is to respond to external events [1]. The context of the external events is important so the required system responses to these events and the timeframe in which these responses have to occur/complete can be identified. The external events for the ECG system are listed in Table 1.

Arrival pattern can be E – episodic or P – periodic. Direction Pattern is IN or OUT but for the reactive systems mainly all events are IN. User working modes are Monitoring, Recording, Reviewing and Admin. Services are described in the Use Case diagram: Capture the ECG signal, Make a Hard Copy of the ECG signal, etc. Some services cannot run in parallel, but some are compatible – like signal capturing and signal monitoring. Mutual exclusion of the services is specified by the limitations of the concrete implementation, like exclusion of simultaneous ECG storage and printing.

3.3. Object Structure

The system class diagram gives the general solution for the application. The class diagram in Figure 1 presents the main application classes and their architectural relationships as associations, aggregations and generalization. We have applied the ‘underlay the noun’ strategy, and subsequently performed object classification [1].

Active objects produce and control actions (Controller and Timer), or produce and analyze data (ProcessSignal, InSignal, and OutSignal). The other main types of objects are: device wrapper objects such as HardCopyDevice and ADSignal, interface objects such as UserInterface, DisplaySignal and HardCopySignal; and persistent data objects such as StoreSignal.

<table>
<thead>
<tr>
<th>Event</th>
<th>System response</th>
<th>Direction pattern</th>
<th>Arrival pattern</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power On</td>
<td>a. Configure system&lt;br&gt;b. Enter Admin mode</td>
<td>IN</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>Time unit passes</td>
<td>a. Update clock&lt;br&gt;b. Inform controller</td>
<td>IN</td>
<td>P</td>
<td>Defined time unit</td>
</tr>
<tr>
<td>A sample period</td>
<td>a. Capture input signal values&lt;br&gt;b. Inform controller</td>
<td>IN</td>
<td>P</td>
<td>$\frac{1}{2}$ sample period</td>
</tr>
<tr>
<td>User selects</td>
<td>a. Stop current mode&lt;br&gt;b. Enter new mode</td>
<td>IN</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>User starts service</td>
<td>a. Stop current service if needed&lt;br&gt;b. Start new service</td>
<td>IN</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>User terminates</td>
<td>a. Stop current service</td>
<td>IN</td>
<td>E</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. External events
3.4. Object Behavior

The system class diagram presents only a static view of the application. Message Sequence Diagrams (MSDs), collaboration diagrams and state charts are UML concepts used for behaviour modeling. MSDs are a much better way to describe the dynamic aspects of the system and they are considered a powerful tool for timely identification of hidden requirements.

Figure 2 presents an MSD for the Use Case: “Make a Hard Copy”. It shows the objects participating in the implementation of that scenario and the corresponding messages and actions. It is presented in a simplified manner since the participants in the communication are top-level objects.

Modeling of MSDs is iterative, and during that process objects and messages are decomposed down to detailed protocols. The MSD documents behavior of all parts of the system and a number of detailed scenarios are required to cover functionality. It is possible that one message starts at one originator object and ends at several target objects, initializing several methods to be executed simultaneously. This is an appropriate way to model concurrency. Unlike concepts in structured design, objects are inherently concurrent. Theoretically it is possible for each object to run on its own processor [1].

4. Design Integration

The modeling tool used must offer developers an appropriate method to document all steps made during the analysis and design. The analysis model is elaborated in the design phase and some new objects important for the implementation are added. Development of the specific application was facilitated by the generic design model [3]. In a real-world scenario, the ECG signal is produced by automatic data monitoring and captured on one or more channels (up to 12) at regular intervals. This produces a time series with a constant sampling interval. The ECG signal object model provide for interpretation based on the identification and inspection of the specific ECG complex waveforms and intervals.
Subsequent research steps focused on implementation of the ECG system using the hardware module Multi-LAB/2 with OSX, an RT operating system (RTOS), on a conventional Windows PC. The RT station runs a parallel MS NT operating system for administrative tasks and RTOS for ECG signal processing. ECG signal recordings were taken from the MIT-BIH Arrhythmia Database (www.physionet.org/physiobank/database/mitdb/), many of them in full 12-lead format, including normal sinus rhythm. Several waveforms were generated to provide for testing purposes.

5. Conclusion

During the OO analysis and the OO design phase the basic components are identified and separated for use in different projects or as a common basis for different scale implementations of a single system. OO techniques promote understanding of requirements leading to flexible and extensible designs which support the modeling of different scale real-time systems. An approach based on computer simulation opens up a range of possibilities for research results to become one of the most successful modern methods for experimental scientific discovery. Some parts of the design model will be adapted to allow for easy data exchange with other research projects with the main purpose of building a modern intelligent laboratory for biomedical signals and systems.

References