Modelling a Cardiac Pacemaker Visually and Formally

Jérôme Leemans, Nuno Amálio
Faculty of Science, Technology and Communication University of Luxembourg, Luxembourg
leemansjerome@gmail.com, nuno.amalio@uni.lu,

Abstract—This paper gives an outline of a visual model of a cardiac pacemaker system, a case study from the grand challenge in software verification. The model is expressed in the Visual Contract Language (VCL), a formal modelling language that describes predicates visually. From VCL diagrams it is possible to generate Z specifications. This is the first visual and formal model of one of the software verification challenges.

Index Terms—visual languages; modelling; formal methods

I. INTRODUCTION

Embedded software in medical devices, such as cardiac pacemakers, is becoming ubiquitous and increasing in content and complexity. Recently, there has been interest in applying formal methods to this domain to answer regulators’ increasing demand for rigorous assurance that software running on medical devices is safe, reliable and functions correctly [1].

This paper outlines a formal model of a cardiac pacemaker, a medical device that regulates the heart beat. The model is built according to an informal specification [2] of a pilot case study of the software verification challenge [3]. The novelty of the work presented here is that this is done visually. The formal model is expressed in the Visual Contract Language (VCL) [4], [5], a formal language for the abstract specification of software systems that is capable of describing predicates visually and from which it is possible to generate formal Z specifications.

II. BACKGROUND: HEART AND PACEMAKERS

The heart is a beating muscle that continuously pumps blood to the rest of the body. It comprises four chambers (Fig. 1): two atria (top) and two ventricles (bottom). The heartbeat is the rhythmic contraction of the heart’s four chambers, which is stimulated by electrical signals that travel through a specific nerve pathway in the heart. These electrical signals begin at the sinoatrial node (SA Node), which is located in the right atrium (Fig. 1). The signal then travels through the atria, causing their contraction, to reach the node, which amplifies the signal sending it through the ventricles, causing their contraction and the consequent pumping of blood into the rest of the body. The heart’s conduction system functions as the body’s own pacemaker. If it does not work properly, it can cause an irregular heartbeat, compromising the flow of blood to the brain and other parts of the body.

A cardiac pacemaker (PM) is an electronic device implanted in the body to regulate the heartbeat [7].

III. THE PACEMAKER’S VCL MODEL

The complete VCL model the pacemaker device (PM) is given in [6]. All diagrams were drawn using VCL’s tool, the Visual Contract Builder (VCB) [5].

A. Overall Model

A VCL model is organised around packages. Each package is a module that isolates a particular concern and that can be composed to build larger modules from smaller ones. The complete set of packages that makes the VCL model of PM is given in [6]. Here, we give some packages of this model.

B. Package CommonPM

This container package defines sets that are common across the PM’s VCL model. In VCL, packages are represented as clouds. Figure 2 presents PD of CommonPM container package.

1http://vcl.gforge.uni.lu
In a container package, a SD introduces sets to be used by other packages. Figure 3 presents package CommonPM’s SD. In VCL, the rounded contours are called blobs; they denote sets. The rectangles denote set elements. The symbol  means that a blob is defined by the things it has inside; when this symbol is not present the topological enclosure relation has a subset meaning. The symbol  means that the blob is a derived blob: a blob defined from other existing sets and objects.

In VCL, the rounded contours are called blobs. The rectangles denote set elements. The symbol  means that the blob is defined by the things it has inside; when this symbol is not present the topological enclosure relation has a subset meaning. The symbol  means that the blob is a derived blob: a blob defined from other existing sets and objects.

The derived blobs are an enumerated set containing all possible operating modes of the pacemaker.

In Fig. 3, BradycardiaMode is a primary blob defining an enumerated set containing all possible operating modes of DDD pacemaker [2], [7]. Other primary blobs defined this way are Bool, Switch, BatteryStatus, and PMEvent. The derived blobs ChambersPaced, ChambersSensed, ResponseToSensing, and RateModulationMode are defined extensionally from the BradycardiaMode primary blob. The derived blobs MicroSec (micro-seconds, μs), MilliSec (milliseconds, ms), MicroVolt (micro-volts, μv), MilliVolt (millivolts, mv) and PPM (pulses per minute) are defined from the primitive blob Nat (represents set of natural numbers).

C. Package RateSmoothing

This package factors the rate smoothing programmable feature, responsible for preventing sudden changes in the rhythm of pacing [7].

This package’s PD (Fig. 4a) says that ensemble package RateSmoothing sees CommonPM (hollow-headed arrow). The package’s SD (Fig. 4b) uses two typical constructions: (a) the blob in double-lined bold is a package blob, (b) the derived blob RateSmoothingValue is defined through a constrained blob expression, where each arrow represents a constraint. The package blob, named RateSmoothing after the package, introduces global state2; the VCL property edges rsUp and rsDown (arrowed lines) represent, respectively, the programmable parameters for the limit in the increase and decrease in the current cardiac cycle interval (a percentage between 0 and 25).

Figure 5a presents BD of package RateSmoothing. In a BD, operations that observe state are represented as assertions (single-lined elongated hexagons) and operations that change state as contracts (double-lined elongated hexagons). Init is the package’s initialisation. Operations IncRSDown and DecRSDown are the operations to increment and decrement the parameter rate smoothing down (rsDown); operations IncRSUp and DecRSUp perform the same operation for rsUp. Operation CalcSmoothedCCI calculates the value of a new cardiac interval based on the rate smoothing parameters. The package’s initialisation (Fig. 5b) sets the parameters rsDown and rsUp to 0.

IV. Discussion

We have formally modelled a cardiac pacemaker visually. We could describe all properties of the complex PM critical system visually in VCL. The full model of PM is given in [6].

Table I presents some data collected from the model. PM is a medium sized system. Our VCL model comprises 17 packages, which define 53 blobs and 113 operations.

The VCL model was built from the informal specification of Boston Scientific [2]. We found many omissions, ambiguities and inconsistencies that are documented in [6]. Often, we had to resort to the wealthy literature on cardiac pacemakers (e.g. [7]) to resolve requirements issues. We have covered all the functionality of the case study.

V. Conclusions

The VCL model outlined here is part of our work on validating and evaluating VCL and its tool. We have modeled in VCL a benchmark case study from the software verification challenge. To our knowledge, this is the first complete visual model of one of the formal methods challenges. We are currently working on formally validating the model against its requirements.

REFERENCES