Contents

CHAPTER 1

Verifying Trustworthy Cyber- Physical Systems using Closed-loop modeling

Neeraj Kumar Singh

McMaster Centre for Software Certification, McMaster University
Hamilton, Ontario, Canada

Mark Lawford

McMaster Centre for Software Certification, McMaster University
Hamilton, Ontario, Canada

Thomas S. E. Maibaum

McMaster Centre for Software Certification, McMaster University
Hamilton, Ontario, Canada

Alan Wassyng

McMaster Centre for Software Certification, McMaster University
Hamilton, Ontario, Canada

CONTENTS

Trustworthy Cyber Physical Systems (TCPS) are safety critical systems, in which failure can lead to injuries and loss of life. These systems demand strong integration and co-ordination between the computing sciences, network communication and physical modeling. Analyzing system requirements is a major challenge in the area of safety-critical software, where requirements quality is also an important issue in building a dependable cyber-physical system. Most projects fail due to a lack of understanding of user needs, inadequate knowledge of the system's environment, and inconsistent system specifications. This typically results in poor system requirements. Since software plays such an important role in critical systems embedded in a physical environment, it is essential that we trace unidentified and hidden requirements by validating and checking the consistency of the system requirements. To this end, formal methods that model the closed-loop system are invaluable. In this chapter, we present an incremental proof-based development of a closed-loop model of the Cardiac Resynchronization Therapy (CRT) and heart. We analyze the prime benefits of a closed-loop modeling approach in requirements engineering to validate the appropriateness and correctness of system behaviors in the early stage of system development, including new research directions.

1.1 INTRODUCTION

The trustworthy cyber-physical systems (TCPS) are dependable critical systems that refer to the tight integration of and coordination between computational and physical resources [7]. The TCPS innovate several industrial sectors

related to avionic, transportation, medical, space and automotive domains, in which main research goal is to improve our own ability to understand and exploit interfaces between the cyber and physical worlds and to innovate new behaviors and capabilities from their seamless integration. An increasing demand for new technology forces for the rapid adoption of commercial firmware and software for TCPS. The rapid adoption for TCPS increases vulnerabilities could lead to devastating system failures. A failure in these system could result of loss of life, including reputation and economical damage. In fact, any failure in medical domain is a serious public health problem and poses a threat to patient safety. For example, the USA Food and Drug Administration (FDA) has reported several recalls for the cardiac pacemaker and implantable cardioverter-defibrillator (ICD). These recalls are responsible for a large number of serious illnesses and deaths. During 1900-2002, 17,323 devices (8834 pacemakers and 8489 ICDs) were explanted and 61 deaths (30 Pacemaker patients, 31 ICD patients) were reported due to erroneous behavior according to the FDA report. The FDA found that the deaths and adverse-events associated with the cardiac pacemaker and ICDs were caused by the product design and engineering flaws including firmware problems [23].

Requirements engineering (RE) provides a framework for analysing and simplifying a complex system to define the system requirements consistent and precisely using informal, semiformal and formal techniques. It plays an important role in the early stage of system development for reducing the development cost, meeting system qualities and success of the system. The main reasons of any project failure are inconsistent system requirements, missing system behavior and lack of understanding of the system requirements. An increasing demand for new emerging technologies and the growing system complexities require to pay more attention on the requirement engineering to omit failures in systems. The associated tools for requirement engineering assist to identify the common problems, such as incompleteness, ambiguity, inconsistencies, and vagueness encountered during the elicitation and specification of the system requirements [6].

Software play a vital role in developing and controlling the critical embedded systems. Over the past forty years, formal techniques have shown some promising results in several domains, including healthcare, automotive, avionic and nuclear by identifying possible errors through formal reasoning. The formal reasoning has great impact in developing the system requirements or checking the correctness of functional requirements. In the current industrial practices, formal methods have been used to meet the standard requirements or certification requirements. For example, ISO 26262 [31] standard has adopted the formal methods to design a passenger vehicle, particularly to meet safety requirements of ASIL D. Validation of requirements specification is an integral and essential part of the requirements engineering. Validation is a process of checking, together with stakeholders, whether the requirements specification meets its stakeholders' intentions and expectations [24].

To identify emergent behaviors according to the given physical environ-

ment (i.e. the heart in cardiac pacemaker), missing requirements and inconsistencies in the early stage of system development for developing a safe and dependable system, we need to look beyond the system itself and into the working environment, including human interactions for specifying and verifying the given system requirements. In this chapter, we demonstrate the results of our new work on the formalization of a closed-loop model of the *Cardiac Resynchronization Therapy (CRT)* pacemaker and heart. The closed-loop model is an integration of system model (CRT pacemaker) and environment model (heart), in which both the system and environment models are formalized using formal techniques. For developing the closed-loop model, we use the Event-B modeling language that supports stepwise refinement. This stepwise refinement is used to introduce safety properties at each layer of refinement to guarantee a safe behavior of the CRT pacemaker. This closed-loop modeling approach is used to identify emergent behaviors according to the dynamic functions of the biological environment heart and to help in the certification of the CRT pacemaker. Moreover, the formal verification and validation of this closed-loop model helps to identify missing system requirements and new emergent behaviors that are not covered earlier during the requirements elicitation process. The closed-loop modeling approach helps for finding not only the missing system requirements but also to get a confidence in the early stage of system development by providing required safety properties under the virtual environment. Our main objectives and contributions are given below. We can consider all these objectives for developing any other TCPS that will use the closed-loop modeling approach for verifying the system requirements using formal techniques.

1. Closed-loop modeling in the early stage of TCPS development;
2. Identifying gaps or inconsistencies in the requirements of TCPS;
3. To verify and validate the behavior requirements of TCPS;
4. Strengthening the given TCPS requirements;
5. To support “what-if" analysis during the formal reasoning of TCPS;
6. Traceability of missing behaviors that leave a TCPS in undesired states;
7. Automatic identification of emergent behaviors;
8. Validation of the TCPS assumptions;
9. To demonstrate how we can help to meet the FDA requirements for certifying the medical TCPS using the closed-loop formal modeling;

The structure of the chapter is as follows. In Section 1.2, we review preliminary material: requirement engineering and CRT pacemaker. Section 1.3

presents an environment modeling of the heart, and the CRT pacemaker control requirements are presented in Section 1.4. Section 1.5 explores an incremental proof-based formal development of a closed-loop system of the CRT pacemaker and heart. A brief discussion is provided in Section 1.6. Section 1.7 presents related work and finally, in Section 1.8, we conclude the chapter.

1.2 PRELIMINARIES

1.2.1 Requirements Engineering

The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as a condition or capability that must be met or possessed by a system or system components to satisfy the contract, standard, specification, or other formally imposed document [1]. Requirements engineering is a branch of the software engineering that allows to use systematic techniques to analyse system requirements for checking the required properties of completeness and consistency of a given system [35]. The requirements engineering is a complex process that contains several small steps, such as elicitation, specification, validation, analysis and management, for developing a system. In these steps, the requirement elicitation is a process for identifying, reviewing, checking and documenting the stakeholder requirements; the requirement specification is used to documenting the stakeholder needs and constraints precisely using formal or semi-formal techniques; the requirement analysis checks the stakeholder requirements and system constraints using formal and informal techniques; the requirements verification ensures the completeness, correctness, understandable and consistent of system behavior according to stakeholders; and finally, the requirements management is used for managing, coordinating, and documenting the system development life-cycle.

The elicitation process of requirements engineering is an important step for capturing the rationales and sources adequately to understand the requirements evolution and verification. Requirement analysis plays an important role to advocate the required properties of the systems and software development process. In addition, it also supports feedback mechanism to improve system requirements by incorporating useful information for reducing complexities by removing complex requirements by simple requirements. There are several techniques that help for improving the quality of requirements for both the ordinary and dependable systems. In this chapter, we demonstrate the result of our work for analyzing the given system requirements of a TCPS under the virtual environment by developing a closed-loop model. This approach has potential benefits to identify new emergent behaviors and missing system requirements in order to meet the required functional behavior of the system under the given environment. This approach is useful for analyzing the system requirements as long as its adoption decision is present preferably during the early stages of the system development, and we need to understand how a decision on analyzing requirements is made and which factors influence an

adoption of the requirements engineering. In our work, we present a conceptual treatment for analyzing the system requirements by developing a closed loop model of TCPS and virtual operating environment for identifying the emergent properties and peculiar requirements, which eventually provide us with a theoretical lens to examine this adoption in a systematic manner [27].

1.2.2 CRT Pacemaker

The cardiac pacemaker is a complex electronic device that is designed to maintain an adequate heart rate in cases of *bradycardia*. This device is equipped with a microprocessor that controls heart rhythm intelligently by observing an actual behavior of the heart. The pacemaker generally serves two main activities known as *pacing* and *sensing*. A sensor is used to sense an intrinsic activity of the heart, and an actuator is used to deliver a short intense electrical pulse into the heart. There are several sensors and actuators are required together to sense and to actuate into multiple heart chambers, and all these sensors and actuators are controlled by the microprocessor [4].

A Cardiac Resynchronization Therapy (CRT) or multi-site pacing device is one of the advanced pacemakers that is designed to maintain heart rate by treating a specific form of heart failure – poor synchronization of the two lower heart chambers. This device has mainly three electrodes, equipped with sensors and actuators, for right atrium, right ventricle and left ventricle. The sensors of this device sense intrinsic activities from the chambers and the actuators deliver a short intense electrical pulse to the chambers of the heart to help them beat together synchronously. The basic elements of a CRT pacemaker are given as follows:

1. **Leads:** A set of insulated flexible wires for transmitting electrical impulses between microprocessor and heart to fulfill the requirement of pacing and sensing.

2. **The CRT Generator:** A metal case that contains the microprocessor and battery. The microprocessor is also called the brain of the CRT pacemaker that controls entire system functionalities.

3. **Device Controller-Monitor (DCM):** It is an external device that communicates with an implanted CRT pacemaker through wireless connection, and it helps for setting new parameters, changing configuration and monitoring an actual behavior of the heart and implanted CRT pacemaker.

4. **Accelerometer:** It is a specific sensor that is used to measure body motion or dynamic activities to allow modulated pacing and sensing to control the heart rhythm according to the physical needs of a patient.

Chambers Paced	**Chambers Sensed**	**Response to Sensing**	**Rate Modulation**	**Multisite Pacing**
O-None	**O**-None	**O**-None	**O**-None	**O**-None
A-Atrium	**A**-Atrium	**T**-Triggered	**R**-Rate Modulation	**A**-Atrium
V-Ventricle	**V**-Ventricle	**I**-Inhibited		**V**-Ventricle
D-Dual(A+V)	**D**-Dual(A+V)	**D**-Dual(T+I)		**D**-Dual(A+V)

TABLE 1.1: The NASPE/BPEG Generic Code for Antibradycardia Pacing

1.2.3 Definition of the NBG Code

The NASPE/BPEG generic (NBG) code is summarized in Table 1.1. There are five columns in the table, where each column represents a sequence of letters for presenting the different type of operating modes. The first letter of the code indicates the chamber(s) in which pacing occurs, the second letter indicates the chamber(s) in which sensing occurs, and the third letter indicates each instance of sensing on the triggering or inhibition of subsequent pacing stimuli. The fourth letter is optional, and indicates the presence (R) or absence (O) of an adaptive rate modulation. The last letter indicates whether multisite pacing is present in (O) none of the cardiac chambers, (A) one or both atria, (V) one or both ventricles, and (D) any combination of atria and ventricles.

1.2.4 Event-B

In our work, we choose the Event-B modeling language [2, 32] that enables to formalize a system using *correct by construction* approach. The *correct by construction* approach allows to design a complex system incrementally by adding concrete details in each new refinement level. The incremental development gradually builds a concrete system by introducing new safety properties and checking the correctness of required behavior at each refinement layer. The basic system modeling components are *context* and *machine*. The main elements of the *context* are *carrier set*, *constant*, *axiom* and *theorem* that describe the static properties of a system. The main elements of another component *machine* are *variable*, *invariant*, *event* and *theorem* that specify the dynamic properties of a system. An *event* is composed of the *guard* and *action* that models a changing state of a system. At each refinement step, we can either refine abstract events or introduce a new set of variables, invariants and events. In addition, we can also introduce new safety properties and theorems for developing a safe system. The abstract events can be refined by: (1) keeping the event as it is; (2) splitting an event into several events; or (3) refining by introducing another event to maintain state variables. At each new refinement level, the developing system always preserves the abstract functional behavior and required safety properties.

Rodin [32] is an open source eclipse based Integrated Development Environment (IDE) for developing the Event-B models. This is a collection of plugins that supports model management, model development, refinement based modeling, model composition/decomposition, proof obligation generation, dis-

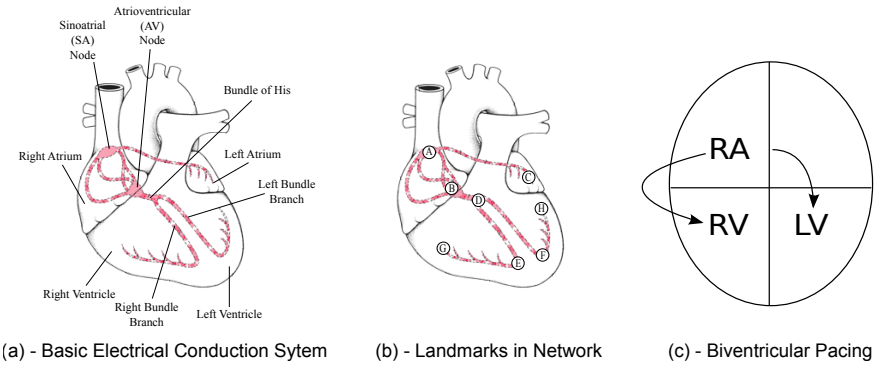


FIGURE 1.1: The Electrical Conduction and Landmarks of the Heart

charging the generated proof obligations using automated theorem provers, and code generation. Due to page limitation, we do not discuss the Event-B modeling language in detail. There are several publications and books [2, 32] available for fundamental and refinement strategies to gaining experience and knowledge in Event-B.

1.3 ENVIRONMENT MODELING (THE HEART)

The heart, a biological muscular organ, pumps blood to circulate in the entire body. It consists of four chambers. The left and right atrium chambers collect blood and pump it into the lower ventricle chambers to pump blood out to the lungs or other parts of the body. The heart requires an electrical stimulus to contract and relax periodically. An electrical stimulus is generated by the small mass of specialized tissue called *sinus node*. The generated electrical impulse travel down through the conduction network. The flow of an electrical impulse varies, and it is time dependent to synchronize the heart chambers: atria and ventricles. For example, the atria contract earlier than ventricles, so that the blood pumps out from atria and pumps it into ventricles. The basic components of the heart are depicted in Fig. 1.1(a). To model the heart behavior abstractly, we consider a set of landmark nodes *(A, B, C, D, E, F, G, H)* on the conduction network (see Fig. 1.1(b)). All theses landmarks are identified through literature survey [15, 19, 5, 16] and extensive discussions with the cardiologists and physiologists.

In this section, we present a formal definition of the heart, and the required properties related to the impulse propagation time and impulse propagation speed. Moreover, we also discuss heart blocks and cellular automata that are used for specifying the heart behavior correctly. This brief introduction allows users to understand the modeling concepts of the heart and for developing a closed-loop model together with the CRT pacemaker. A detailed description

about the heart and formalization steps are available in [26, 25, 33]. Below we introduce only necessary elements to formally define the heart.

Definition 1 (The Heart). *Given a set of nodes N, a transition (conduction) t is a pair (i, j), with i, j ∈ N. A transition is denoted by i ⇝ j. The heart system is a tuple HSys = (N, T, N_0, TW_{time}, CW_{speed}) where:*

• $N = \{A, B, C, D, E, F, G, H\}$ *is a finite set of landmark nodes in the conduction pathways of the heart;*
• $T \subseteq N \times N = \{A \mapsto B, A \mapsto C, B \mapsto D, D \mapsto E, D \mapsto F, E \mapsto G, F \mapsto H\}$ *is a set of transitions to represent electrical impulse propagation between two landmark nodes;*
• $N_0 = A$ *is the initial landmark node (SA node);*
• $TW_{time} \in N \to TIME$ *is a weight function as time delay of each node, where* $TIME$ *is time delay in range;*
• $CW_{speed} \in T \to SPEED$ *is a weight function as impulse propagation speed of each transition, where* $SPEED$ *is propagation speed in range.*

Property 1 (Impulse Propagation Time). *In the biological heart, electrical impulse originates from SA node (node A) and then it travels through the conduction network and it terminates to the atrial muscle fibers (node C) and at the end of Purkinje fibers into both side of the ventricular chambers (node G and node H). The impulse propagation time delay differs for each landmark nodes (N). The Impulse propagation time is represented as a total function* $TW_{time} \in N \to \mathbb{P}(0..230)$. *The impulse propagation time delay for each node is represented as :* $TW_{time}(A) = 0..10$, $TW_{time}(B) = 50..70$, $TW_{time}(C) = 70..90$, $TW_{time}(D) = 125..160$, $TW_{time}(E) = 145..180$, $TW_{time}(F) = 145..180$, $TW_{time}(G) = 150..210$ *and* $TW_{time}(h) = 150..230$.

Property 2 (Impulse Propagation Speed). *Similar to the impulse propagation time, the impulse propagation speed also differs for each transition (i ⇝ j, where i, j ∈ N). The impulse propagation speed is represented as a total function* $CW_{speed} \in T \to \mathbb{P}(5..400)$. *The impulse propagation speed for each transition is represented as:* $CW_{speed}(A \mapsto B) = 30..50$, $CW_{speed}(A \mapsto C) = 30..50$, $CW_{speed}(B \mapsto D) = 100..200$, $CW_{speed}(D \mapsto E) = 100..200$, $CW_{speed}(E \mapsto G) = 300..400$ *and* $CW_{speed}(F \mapsto H) = 300..400$.

The sinoatrial (SA) node spontaneously emits some electrical current that spreads through the walls of the atria, causing them to contract. This SA node is known as physiological pacemaker of the heart that is represented by the node A in Fig 1.2(a) and it is responsible for maintaining the heart rhythm. From SA node, an electrical impulse propagates through atria chamber and it reaches to the nodes B and C (see Fig. 1.2(b)) at the end of muscle fibers without crossing the boundary between atria and ventricles.

An electrical impulse generated from the SA node only enters through the atrioventricular (AV) node. The atrioventricular node (AV node) shown as the node B (see Fig. 1.1(b)) is located at the boundary between atria and

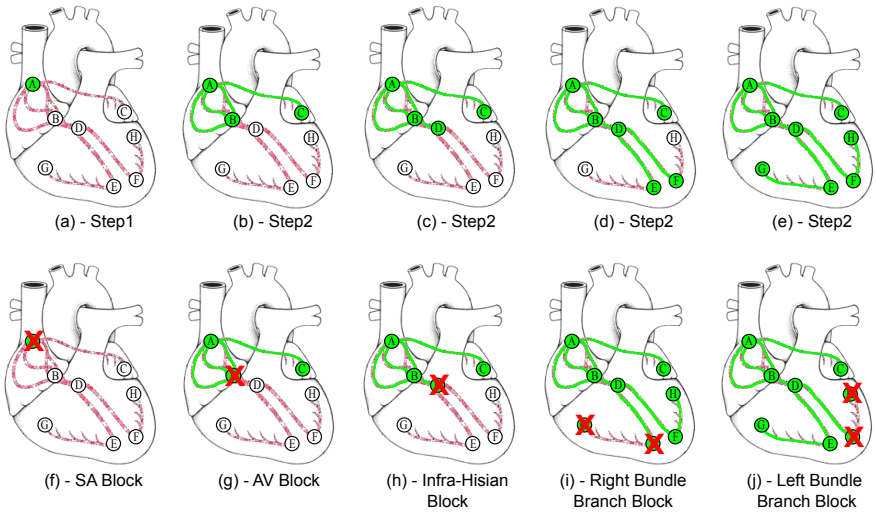


FIGURE 1.2: Impulse Propagation through Landmark nodes and Heart Blocks

Location in the heart	**Cardiac Activation Time (ms.)**	**Location in the heart**	**Conduction Velocity (cm/sec.)**
SA Node (A)	0..10	A $\mapsto$ B	30..50
Left atrium muscle fibers (C)	70..90	A $\mapsto$ C	30..50
AV Node (B)	50..70	B $\mapsto$ D	100..200
Bundle of His (D)	125..160	D $\mapsto$ E	100..200
Right Bundle Branch (E)	145..180	D $\mapsto$ F	100..200
Left Bundle Branch (F)	145..180	E $\mapsto$ G	300..400
Right Purkinje fibers (G)	150..210	F $\mapsto$ H	300..400
Left Purkinje fibers (H)	150..230		

TABLE 1.2: Cardiac Activation Time and Cardiac Velocity [15]

ventricles. There is a small delay at this node to synchronize the atria and ventricles to flow blood effectively. The distal portion of the AV node is made of the bundle of His denoted as the landmark node D (see Fig. 1.1(b)). The bundle of His splits into two branches in the inter-ventricular septum, the left bundle branch and the right bundle branch. Then the electrical impulses enter to the base of the ventricle at the Bundle of His (node D) and then follow the left and right bundle branches along the inter-ventricular septum (see Fig. 1.2(c)).

Two separate the left and right bundle branches propagate together on each side of the septum. Two landmark nodes at the downside of the heart into both the left and right bundle branches are denoted as E and F (see Fig. 1.1(b)). The specialized fibers of left and right bundle branches conduct an impulse rapidly, and the left bundle branch activates the left ventricle and the right bundle branch activates the right ventricle (see Fig. 1.2(d)).

The bundle branches are divided into an extensive system of Purkinje

fibers that conduct the impulses at high velocity (see Table 1.2) throughout the ventricles. The Purkinje fibers stimulate individual groups of myocardial cells to contract. Two landmark nodes G and H (see Fig. 1.1(b)) are denoted at the end of the Purkinje fibers in the ventricles (see Fig. 1.2(e)). At the end of the Purkinje fibers, the electrical impulse is transmitted through the ventricular muscles [15, 19].

The heart electrical behavior plays an important role to synchronize atria and ventricles and it helps to optimize the haemodynamic. Some minor changes in the conduction time or conduction speed between landmark nodes cause different types of abnormalities known as *arrhythmias*. These arrhythmias can be categorized as bradycardia (slow heart rate) or tachycardia (rapid heart rate). All the possible range of values for the conduction speed and conduction time are given in Table 1.2.

1.3.1 Heart Block

Heart block is a disorder of impulse conduction which stimulates heart muscle contraction. The normal cardiac impulse emits from the SA node that spreads throughout the atria and ventricles. Disturbance into conduction may demonstrate as slow conduction, intermittent condition failure, or complete conduction failure. All these kinds of conduction failures are also known as 1st, 2nd and 3rd degree blocks. Fig. 1.2 depicts different kinds of heart blocks throughout the conduction network using a set of landmark nodes.

1.3.1.1 SA block:

This type of block occurs within the SA node (A) known as sinoatrial (SA) nodal block or sick sinus syndrome. In this block, the SA node fails to originate an impulse and the heart misses one or two beats at regular or irregular intervals (see Fig. 1.2(f)).

1.3.1.2 AV block:

The AV block occurs due to conduction defects between atria and ventricles. This block may cause by the AV node (B), bundle of His (D) or the both nodes B and D (see Fig. 1.2(g)).

1.3.1.3 Infra-Hisian block:

This type of block occurs due to defect after the AV node (B) known as Infra-Hisian block (see Fig. 1.2(h)).

1.3.1.4 Left bundle branch block:

The left bundle branch block occurs when the conduction is interrupted into the left branch of the bundle of His. A block that occurs within the fascicles of the left bundle branch is known as hemiblocks (see Fig. 1.2(i)).

1.3.1.5 Right bundle branch block:

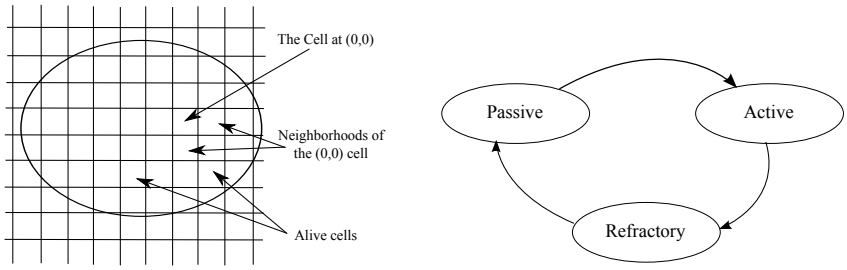
The right bundle branch block occurs when the conduction is interrupted into the right branch of the bundle of His (see Fig. 1.2(j)).

1.3.2 Cellular Automata Model

A cellular automata (CA) model is a set of spatially distributed cells that contains uniform connection pattern among the neighbouring cells and local computation laws. In 1940, Ulam and von Neumann [17] proposed the cellular automata (CA) for investigating the behavior of complex and distributed systems. CA is a discrete dynamic system corresponding to space and time that provides uniform properties for state transitions and interconnection patterns. A CA model can have an infinite number of cells in any dimension. In our work, we only consider a finite number of cells in two dimensions as shown in Fig. 1.3(a). A two-dimensional CA model is defined below.

Definition 2 (The Cellular Automata Model).
Cellular Automata $(CA) = < S, N, T >$ *: Discrete Time System*
S : a set of states
N: a set of neighbouring patterns at (0,0),
T: a transition function
A typical case of the Cellular Automata (CA) is realized in D-dimensional grid, N consists of D-tuples of indices from a coordinate set:
I: $N \subseteq I^D$,
Hence the cellular model for 2D becomes,
$N \subseteq I^2$.
$T : S^{|N|} \rightarrow S$
To consider automaton specified by the cellular automata (CA), let λ *and* α *be a global state and the global transition function of the cellular autamata (CA), respectively. Then* $\lambda = \{\tau | \tau : I^2 \rightarrow S\}$ *and* $\alpha(\lambda(i,j)) = T(\tau | N + (i,j))$ *for all* τ *in* λ *and (i,j) in* I^2.

Definition 3 (State Transition of a Cell). *The heart muscle is composed of heterogeneous cells, the cellular automata model of the muscle,* CAM_{CA}*, is characterized with no dependencies on the type of cells.* CAM_{CA} *is defined as follows:*
$CAM_{CA} = < S, N, T >$
$S = \{Active, Passive, Refractory\}$
$N = \{(0,0), (1,0), (-1,0), (0,1), (0,-1)\}$
$s'_{m,n} = s_{m,n}(t+1)$



(a) - A Two-DImensional Cellular Automata Model (b) - State Transition of a Cell

FIGURE 1.3: Two-Dimensional Cellular Automata (CA) and State Transition Model

$s'_{m,n} = T(s_{m,n}, s_{m+1,n}, s_{m-1,n}, s_{m,n+1}, s_{m,n-1})$
where, $s_{m,n}$ denotes the state of the cell located at (m,n) and T is a transition function of cellular automata (CAM_{CA}), which is a function for the next state to be defined in Fig. 1.3(b).

The cardiac heart muscle cells have mainly three states: *Active*, *Passive* and *Refractory*. Initially, all the cells are in the *Passive* state. In the *Passive* state, the cells are electrical discharged and they do not affect any neighbouring cells. When an electrical impulse passes through a cell than the cell would be charged and it is eventually activated, and the current state of the cell switches in the *Active* state. An active cell can transmit an electrical impulse to its neighbouring cells, and then the active cell can switch into the *Refractory* state, in which the cell can not be reactivated instantly. After delaying, the *Refractory* state cells can switch in the *Passive* state to await for the next impulse (see Fig. 1.3(b)).

1.4 CRT PACEMAKER CONTROL REQUIREMENTS

The CRT pacemaker is an advanced electronic device that controls the heart rate by sensing and pacing in various heart chambers. In this section, we describe the system requirements of biventricular sensing with biventricular pacing (BiSP) considering other complex operating modes. The BiSP allows pacing and sensing in the right atrium, left ventricle and right ventricle (see Fig. 1.1(c)).

Biventricular pacing coordinates the left ventricle (LV) and right ventricle (RV), and intra-ventricular regional wall contractions, by synchronizing with the sinus rhythm. There are various intrinsic activities related to pacing and sensing events that can reset escape intervals, such as atrioventricular interval (AVI) and ventriculoatrial interval (VAI). Biventricular pacing controls the heart rate using various combinations of the timing form events in either LV or RV. For example, the first ventricular sense either from the left or right ventricular chamber can reset the ventriculoatrial interval (VAI) and the heart

rate depends on intervals between the first ventricular events in each cycle. However, heart rate intervals can vary due to stimulation in the opposite chambers.

Minor delays between RV and LV pacing introduce complications in biventricular timings. These timings allow several definitions of escape intervals, such as atrioventricular interval (AVI) and ventriculoatrial interval (VAI). The pacing rate is the sum of the VAI and AVI for dual chamber timing. The definition preserves for biventricular timing of the VAI and AVI to pacing either the RV for RV-based timing or the LV for LV-based timing. The pacing delay can be represented by RVI - LVI interval. It can be negative, positive or zero as per the occurrence order of the stimulation in both the left and right ventricles. The possible scenarios of the biventricular sensing and pacing are depicted in Fig. 1.4 that is described below assuming normal sensing and pacing activities in the right atrium.

1. **Scenario A** shows a situation in which the CRT pacemaker paces in both left and right ventricles after an AVI in which no intrinsic heart activity is detected.
2. **Scenario B** shows a situation in which the CRT pacemaker paces in the left ventricle only after an AVI, while RV pacing is inhibited due to sensing of an intrinsic activity in the right ventricle.
3. **Scenario C** shows a situation in which the CRT pacemaker paces in the right ventricle only after an AVI, while LV pacing is inhibited due to sensing of an intrinsic activity in the left ventricle.
4. **Scenario D** shows the case where pacing activities are inhibited in both left and right ventricles due to sensing of intrinsic activities in both left and right ventricles.

In the following section, we provide a detailed description of given scenarios for biventricular sensing and pacing in order to capture the possible behavioral requirements.

We consider biventricular sensing for RV-based timing with the positive RVI - LVI interval in the absence of intrinsic conduction (see Fig. 1.5(a)). Following the AVI, an LVP event follows an RVP event.

An event sense related to the right ventricle resets all the pacing intervals for both the right and left ventricles, so pacing is not allowed in the right ventricle or in the left ventricle following an RVS event. Fig. 1.5(b) shows that an RVS event resets the timing cycle and starts a new VAI.

An event sense related to the right ventricle may reset the VAI so the right ventricle is not paced, but pacing is allowed in the left ventricle if any intrinsic activity is not detected in the left ventricle. Fig. 1.5(c) shows that the RVS event starts a new VAI but also is followed by an

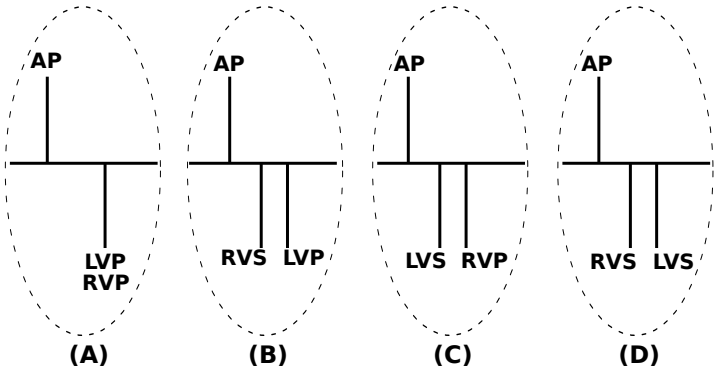


FIGURE 1.4: Possible scenarios of the biventricular sensing and pacing. AS = atrial sensed; AP = atrial paced; LVS = left ventricular sensed; LVP = left ventricular paced; RVS = right ventricular sensed; RVP = right ventricular paced.

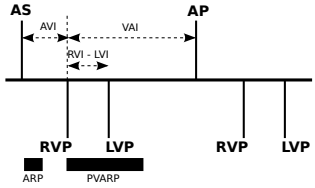
LVP event (*) unless an LVS event occurs first (**). Alternatively, the VAI may be initiated from the right ventricle pace at the end of AVI. Fig. 1.5(d) shows that a delivery of RVP starts a new VAI but also is followed by an LVP event (*) unless an LVS event occurs first (**).

An event sense related to the right ventricle may allow an immediate trigger to pace in the left ventricle without delaying to synchronize the left ventricle and right ventricle contractions. Fig. 1.5(e) shows that the RVS event results in an immediate LVP event.

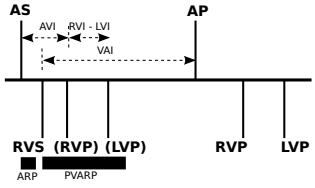
Sometimes, pacing in opposite chambers can be advantageous after some delay rather than immediately. Fig. 1.5(f) shows that an RVS event is followed by an LVP event after the modified RVI - LVI interval. This interval may be different from the RVP - LVP interval.

An event sense related to the left ventricle in AV or RVI - LVI interval before pacing in the left ventricle may inhibit the left ventricle pace, and the right ventricle pace would be unaffected unless the right ventricle sense occurs before the AVI. Fig. 1.5(g) shows that the LVS event does not inhibit the RVP event. However, if left ventricle to right ventricle conduction occurs quickly enough, the RVS event starts a new timing cycle.

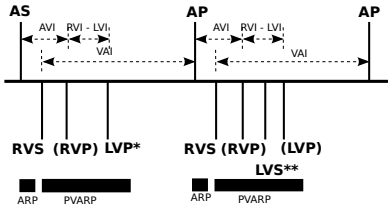
During AVI, an event sense related to the left ventricle may trigger an immediate pace in the right ventricle, which can reset the pacing intervals. Fig. 1.5(h) shows that the left ventricle sense event does not reset the timing cycle but initiates an immediate RVP event.



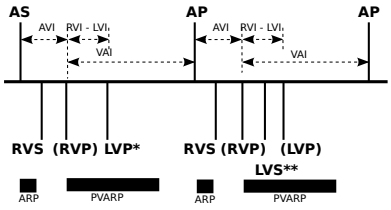
(a) - BiSP in the absence of AV conduction



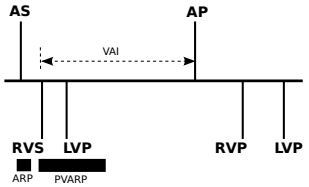
(b) - BiSP with a RVS event



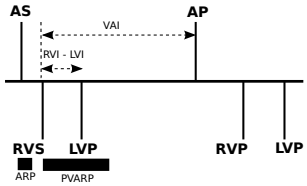
(c) - The RVS event starts a new VAI in BiSP



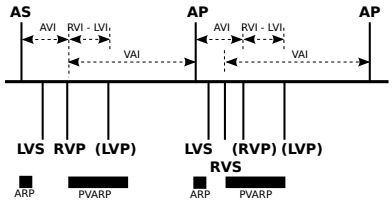
(d) - The theoretical point of delivery of RVP starts a new VAI in BiSP



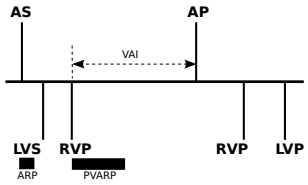
(e) - BiSP with triggering



(f) - A RVS event is followed by a LVP event after a modified RVI-LVI interval in BiSP



(g) - BiSP with RV-based timing



(h) - BiSP with right ventricular-based timing and trigger

FIGURE 1.5: Biventricular sensing and pacing (BiSP) Requirements

1.5 CLOSED-LOOP MODEL OF THE CRT PACEMAKER AND HEART

In this chapter, we present a closed-loop formal model of the CRT pacemaker and heart, in which the formal model of the heart is used as a virtual environment, and the formal model of the CRT pacemaker is used as a TCPS that guarantees to response according to intrinsic activities of the heart (see Fig. 1.6). The main objective of this closed-loop model is to verify and validate the complex properties of CRT pacemaker under the virtual environment, identifying new emergent behaviors and strengthening the given system requirements. As far as we know, this is the first closed-loop formal model of the CRT pacemaker and heart to analyze the functional behavior of the CRT pacemaker under the virtual environment by satisfying the required safety properties. For developing the closed-loop model, we use the previously developed and verified formal models of the CRT pacemaker [34] and heart [26, 33]. In fact, we use our previous works as the basis for developing a closed-loop model of the CRT pacemaker and heart using stepwise refinement from scratch. To check the correctness of the closed-loop system, we introduce several safety properties and discharge all the generated proof obligations at each refinement level. An abstract model and a series of gradually refined models are described below.

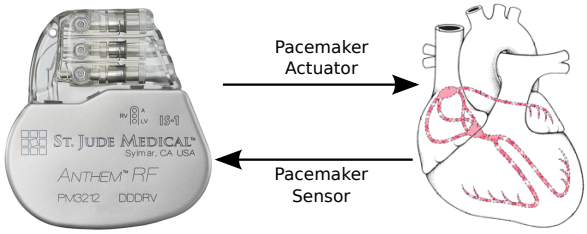


FIGURE 1.6: The Closed-loop Model[1]

1.5.1 Abstract Model

To model the functional behavior of the heart abstractly, we formalize the characteristics of electrical impulse propagation using conduction network in various chambers. An impulse propagation controls the temporal activities of the heart, and it allows heart muscle to contract and relax periodically. We identify a set of landmark nodes (see Fig. 1.1(b)) on the conduction network. These nodes are connected to form a path for impulse propagation from the

[1]The image of CRT pacemaker is taken from: `http://www.amayeza.co.za/files/content/images/img331.jpg`

SA node to the end of Purkinje fibers. An impulse propagation in the conduction network shows normal or abnormal behavior of the heart. To specify the static properties of the heart, we declare an enumerated set *ConductionNode* and three constants *ConductionTime*, *ConductionPath* and *ConductionSpeed*. The enumerated set *ConductionNode* is a collection of landmarks nodes, and the constants *ConductionTime*, *ConductionPath* and *ConductionSpeed* are impulse propagation time for each landmark node, impulse propagation network of the heart and impulse propagation speed for each path, respectively. These are defined using axioms ($axm1$ - $axm4$), which are extracted from the heart definition and the given properties (see Section1.3).

An abstract model of the CRT pacemaker formalizes only sensing and pacing behaviors for each chamber (RA, RV and LV) without considering any temporal requirements. To model the functional behavior of the CRT pacemaker abstractly, we define an enumerated set *Status* in $axm5$ that shows the *ON* and *OFF* states of the actuators and sensors.

$$
\begin{array}{l}
axm1 : partition(ConductionNode, \{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{G\}, \{H\}) \\
axm2 : ConductionTime \in ConductionNode \rightarrow \mathbb{P}(0 \mathinner{..} 230) \\
axm3 : ConductionPath \subseteq ConductionNode \times ConductionNode \\
axm4 : ConductionSpeed \in ConductionPath \rightarrow \mathbb{P}(5 \mathinner{..} 400) \\
axm5 : partition(Status, \{ON\}, \{OFF\})
\end{array}
$$

To model an abstract dynamic behavior of the closed-loop system, we need to develop a formal model of the CRT pacemaker and heart together. In this closed loop model, the CRT pacemaker behaves appropriately by observing the normal and abnormal behaviors of the heart. A virtual environment model of the heart simulates according to an impulse propagation in the conduction network using conduction nodes and the defined properties. An abstract model declares a list of variables to model the closed-loop system. For modeling the heart, we define four variables ($inv1$ - $inv4$): *ConductionNodeState* – to show boolean states of the landmark nodes to distinguish the visited ($TRUE$) and unvisited ($FALSE$) landmark nodes; *CConductionTime* – to present the current impulse propagation time in the conduction network; *CConductionSpeed* – to present the current impulse propagation speed in the conduction network; and *HeartState* – to show boolean states of the heart to represent normal ($TRUE$) and abnormal ($FALSE$) conditions. The CRT pacemaker contains three electrodes, equipped with sensors and actuators, that delivers pacing stimulus in the heart chambers (RA, RV and LV) as per the physiological needs through sensing intrinsic activities of the heart of a patient. All these three electrodes synchronize together to sense and to pace appropriately in different heart chambers whenever required. A list of variables is declared to define different actuators and sensors for each chamber. The actuators are defined by $PM\_Actuator\_A$, $PM\_Actuator\_LV$ and $PM\_Actuator\_RV$, and the sensors are defined by PM_Sensor_A, PM_Sensor_LV and PM_Sensor_RV. These actuators and sensors are defined as the type of *Status* using invariants ($inv5$ -$inv10$).

$inv1 : ConductionNodeState \in ConductionNode \rightarrow BOOL$
$inv2 : CConductionTime \in ConductionNode \rightarrow 0 \mathinner{..} 300$
$inv3 : CConductionSpeed \in ConductionPath \rightarrow 0 \mathinner{..} 500$
$inv4 : HeartState \in BOOL$
$inv5 : PM\_Actuator\_A \in Status$
$inv6 : PM\_Actuator\_RV \in Status$
$inv7 : PM\_Actuator\_LV \in Status$
$inv8 : PM\_Sensor\_A \in Status$
$inv9 : PM\_Sensor\_RV \in Status$
$inv10 : PM\_Sensor\_LV \in Status$

The abstract model of the closed-loop system contains a set of events to show changing states of the CRT pacemaker and heart. For modeling the heart, we define three events, namely *HeartOK* to show a normal state of the heart, *HeartKO* to express an abnormal state of the heart, and *HeartConduction* to trace the current updated values of each landmark node in the conduction network. The event *HeartOK* models the required behavior of the heart when the heart is in the normal state. The guards of this event state that all the landmark nodes of the conduction network must be visited during the impulse propagation, the current impulse propagation time for every landmark node must belong to the given range of the conduction time and the impulse propagation speed for every path must belong to the given range of the conduction speed (see *Property 1 and Property 2*). The action of this event shows that the heart is in the normal state when all the guards of this event are satisfied.

EVENT HeartOK
WHEN
grd1 : $\forall i \cdot i \in ConductionNode \Rightarrow ConductionNodeState(i) = TRUE$
grd2 : $\forall i \cdot i \in ConductionNode \Rightarrow CConductionTime(i) \in ConductionTime(i)$
grd3 : $\forall i, j \cdot i \mapsto j \in ConductionPath \Rightarrow$
$CConductionSpeed(i \mapsto j) \in ConductionSpeed(i \mapsto j)$
THEN
act1 : $HeartState := TRUE$
END

The event *HeartKO* models the required behavior of an abnormal condition of the heart. The guards of this event state that if any landmark node does not visited during the impulse propagation, the current impulse propagation time of any landmark node does not belong to the given range of the conduction time, or the impulse propagation speed of any path does not belong to the given range of the conduction speed (see *Property 1 and Property 2*). The action of this event shows that the heart is in the abnormal state as $FALSE$ when the given guard ($grd1$) of this event is satisfied.

EVENT HeartKO
WHEN
grd1 : $\exists i \cdot i \in ConductionNode \land ConductionNodeState(i) = FALSE)$
$\lor$
$(\exists j \cdot j \in ConductionNode \land CConductionTime(j) \notin ConductionTime(j))$
$\lor$
$(\exists m, n \cdot m \mapsto n \in ConductionPath \land CConductionSpeed(m \mapsto n)$
$\notin ConductionSpeed(m \mapsto n))$
THEN
act1 : $HeartState := FALSE$
END

The event *HeartConduction* models the dynamic behavior of heart conduction abstractly. This event allows to set new updated values to the visited state of the landmark nodes, impulse propagation time, impulse propagation velocity and state of the heart. This event is described to show updated values only in one shot. This event is further refined to model the concrete behaviors of the heart.

EVENT HeartConduction
BEGIN
act1 : $ConductionNodeState :\in ConductionNode \rightarrow BOOL$
act2 : $CConductionTime :\in ConductionNode \rightarrow 0 \mathinner{..} 300$
act3 : $CConductionSpeed :\in ConductionPath \rightarrow 0 \mathinner{..} 500$
act4 : $HeartState :\in BOOL$
END

In the abstract model of the closed-loop system, the CRT pacemaker model describes only discrete functional behaviors for modeling sensors and actuators without considering any temporal requirements. For modeling the CRT pacemaker, we define twelve new events to formalize the sensing and pacing activities in form of changing states for actuators and sensors for each chamber (RA, RV and LV). All these events are very simple, and these events contain only single guard to specify the current state of the actuators/sensors, and then the action of these events allow to change the current state of the actuators/sensors. For example, we present two events *PM_Pacing_On_RV* and *PM_Sensing_On_RV*. The event *PM_Pacing_On_RV* is used to set ON for the right ventricle actuator, when the right ventricle actuator is OFF, and the event *PM_Sensing_On_RV* is used to set ON for the right ventricle sensor, when the right ventricle sensor is OFF. Other events are formalized in the similar way.

EVENT PM_Pacing_On_RV
WHEN
grd1 : $PM\_Actuator\_RV = OFF$
THEN
act1 : $PM\_Actuator\_RV := ON$
END

EVENT PM_Sensing_On_RV
WHEN
grd1 : $PM\_Sensor\_RV = OFF$
THEN
act1 : $PM\_Sensor\_RV := ON$
END

1.5.2 First Refinement: Impulse Propagation and Timing Requirements

This refinement step allows to add more detail in the abstract model of the closed-loop system by refining the CRT pacemaker and heart models together. The heart model is refined by adding conduction behavior that introduces the functional behavior of the impulse propagation in the conduction network. The SA node originates an impulse that passes through the conduction network by visiting all the landmark nodes to reach at the Purkinje fibers of ventricles. For example, an impulse generates from the node A and finally sinks to the terminal nodes (C, G and H). The conduction model uses a clock counter to specify the required temporal properties for impulse propagation.

For modeling the CRT pacemaker, we define a list of constants as static properties for describing the timing requirements for controlling the pacing and sensing events, and to simulate the desired heart behavior. We define four constants atrioventricular interval (AVI), ventriculoatrial interval (VAI), left ventricular interval (LVI) and right ventricular interval (RVI). All these constants are defined in axioms ($axm1$ - $axm4$). An extra axiom $axm5$ is defined as a constraint that specifies that the RVI is greater than or equal to the LVI. In this study, we consider all times to be in milliseconds.

$$
\begin{array}{l}
axm1 : AVI \in 50 \mathinner{..} 350 \\
axm2 : VAI \in 350 \mathinner{..} 1200 \\
axm3 : LVI \in 0 \mathinner{..} 50 \\
axm4 : RVI \in 0 \mathinner{..} 50 \\
axm5 : RVI \geq LVI
\end{array}
$$

In this refinement, we introduce a logical clock for modeling the timing requirements for the CRT pacemaker and heart. To model the clock behavior, we declare a variable now to specify the current clock counter in $inv1$. This clock counter is always progressed by 1 millisecond in every clock tick. In the CRT pacemaker model, we define a variable *PSRecord* to store a time whenever any pacing or intrinsic activity related to sensing occurs within the chambers in $inv2$. The stored time can be used to control the future activities related to pacing and sensing events. The CRT pacemaker model contains a list of variables to synchronize the sensing and pacing events by observing the various states of sensors and actuators for all the three chambers (RA, RV and LV) in order to specify the required behavior. In the heart model, we declare only a variable *CCSpeed_ CCTime_ Flag* to synchronize and for preserving the desired behavior of the heart for capturing the current values of the impulse propagation time and impulse propagation speed in the conduction network.

$$
\begin{aligned}
&inv1 : CCSpeed\_CCTime\_Flag \in BOOL \\
&inv2 : now \in \mathbb{N} \\
&inv3 : PSRecord \in \mathbb{N} \\
&inv4 : HeartState = TRUE \Rightarrow CConductionTime(B) \in ConductionTime(B) \\
&\qquad \wedge\ CConductionSpeed(A \mapsto B) \in ConductionSpeed(A \mapsto B) \\
&inv5 : HeartState = TRUE \Rightarrow CConductionTime(C) \in ConductionTime(C) \\
&\qquad \wedge\ CConductionSpeed(A \mapsto C) \in ConductionSpeed(A \mapsto C) \\
&inv6 : HeartState = TRUE \Rightarrow CConductionTime(D) \in ConductionTime(D) \\
&\qquad \wedge\ CConductionSpeed(B \mapsto D) \in ConductionSpeed(B \mapsto D) \\
&inv7 : HeartState = TRUE \Rightarrow CConductionTime(E) \in ConductionTime(E) \\
&\qquad \wedge\ CConductionSpeed(D \mapsto E) \in ConductionSpeed(D \mapsto E) \\
&inv8 : HeartState = TRUE \Rightarrow CConductionTime(F) \in ConductionTime(F) \\
&\qquad \wedge\ CConductionSpeed(D \mapsto F) \in ConductionSpeed(D \mapsto F) \\
&inv9 : HeartState = TRUE \Rightarrow CConductionTime(G) \in ConductionTime(G) \\
&\qquad \wedge\ CConductionSpeed(E \mapsto G) \in ConductionSpeed(E \mapsto G) \\
&inv10 : HeartState = TRUE \Rightarrow CConductionTime(H) \in ConductionTime(H) \\
&\qquad \wedge\ CConductionSpeed(F \mapsto H) \in ConductionSpeed(F \mapsto H) \\
&inv11 : now = 0 \Rightarrow PM\_Sensor\_RV = OFF \wedge PM\_Actuator\_RV = OFF \\
&inv12 : now = 0 \Rightarrow PM\_Sensor\_LV = OFF \wedge PM\_Actuator\_LV = OFF \\
&inv13 : now = 0 \Rightarrow PM\_Sensor\_A = OFF \wedge PM\_Actuator\_A = OFF \\
&inv14 : PM\_Actuator\_RV = ON \Rightarrow now \geq AVI \vee Immd\_Pace\_RV = 1 \\
&inv15 : PM\_Actuator\_LV = ON \Rightarrow \\
&\qquad now \geq AVI + (RVI - LVI) \vee Immd\_Pace\_LV = 1 \vee Delay\_Pace\_LV = 1 \\
&inv16 : PM\_Actuator\_A = ON \Rightarrow \\
&\qquad now \geq PSRecord + VAI \vee now \geq AVI + VAI
\end{aligned}
$$

We define a list of safety properties to model the correct functionalities of the closed-loop system. Most of the safety properties are defined independently for both the CRT pacemaker and heart. A set of invariants ($inv4$ - $inv10$) defines safety properties for the heart model that states if the heart is in the normal state then the current impulse propagation speed and current impulse propagation time are always within the given range for each landmark node and the defined path of the conduction network. Another list of safety properties are introduced for the CRT pacemaker. Invariants ($inv11$, $inv12$, and $inv13$) state that when the current clock counter is zero then the sensors and actuators are OFF of the right ventricle, left ventricle and right atrium. The next safety property $inv14$ states that if the current clock counter elapses AVI or an immediate pacing is required in the right ventricle then the actuator of the right ventricle must pace. Similarly, the next safety property $inv15$ states that if the current clock counter elapses the total duration of the atrioventricular interval and pacing delay, an immediate pacing is required in the left ventricle, or a delay pacing is detected in the left ventricle then the actuator of the left ventricle must pace. The last safety property $inv16$ states that the current clock counter is elapsed the VAI after detecting the last pacing or sensing activity, or the current clock counter elapses the total duration of the AVI and VAI then the actuator of the right atrium must pace.

In this refinement, we introduce several events to specify the desired behavior of the CRT pacemaker and heart according to the given timing requirements and impulse propagation. There are total thirty events to specify the concrete behavior of the closed-loop model. Few events are the refinement of the abstract events, and other new added events at this level are the refinement of the *skip*. In the heart modeling, we introduce an event *SinusNodeFire* that refines the abstract event *HeartConduction*. This event models the func-

tional behavior of the sinoatrial (SA) node that initially generates an electrical impulse for propagating in the conduction network. The guards of this event state that all the landmark nodes are unvisited, the current impulse propagation time of each node is 0 ms., and the impulse propagation speed for every path is 0 cm/sec. The actions of this event show that the sinoatrial (SA) node is visited, and the current impulse propagation time of the SA node (A) is set to 0 ms.

EVENT SinusNodeFire Refines HeartConduction
WHEN
grd1 : $\forall n \cdot n \in ConductionNode \Rightarrow ConductionNodeState(n) = FALSE$
grd2 : $\forall n \cdot n \in ConductionNode \Rightarrow CConductionTime(n) = 0$
grd3 : $\forall n, m \cdot n \in ConductionNode \wedge m \in ConductionNode \wedge$
$n \mapsto m \in ConductionPath \Rightarrow CConductionSpeed(n \mapsto m) = 0$
THEN
act1 : $ConductionNodeState(A) := TRUE$
act2 : $CConductionTime(A) := 0$
END

We also introduce a list of events to model the stepwise progression of an electrical impulse from SA node to Purkinje fibers. The introduced events synchronize all the heart chambers through progressing an impulse in the conduction network. All these set of events are the refinement of the abstract event *HeartConduction*. An event *HeartConduction_A_B* is formalized to show the basic formalization steps of the other events. The guards of this event states that the sinoatrial (SA) node is visited, the atrioventricular (AV) node is not visited, the current conduction time of the atrioventricular (AV) node belongs to the given range of the conduction time, the conduction speed from sinoatrial (SA) node to atrioventricular (AV) node belongs to the given range of the conduction speed, and the current impulse propagation time and speed flag is $FALSE$. The actions of this event shows that the atrioventricular (AV) node is visited, and the current impulse propagation time and speed flag becomes $TRUE$. All the other refined events are modeled in the similar way.

EVENT HeartConduction_A_B Refines HeartConduction
WHEN
grd1 : $ConductionNodeState(A) = TRUE$
grd2 : $ConductionNodeState(B) = FALSE$
grd3 : $CConductionTime(B) \in ConductionTime(B)$
grd4 : $CConductionSpeed(A \mapsto B) \in ConductionSpeed(A \mapsto B)$
grd5 : $CCSpeed\_CCTime\_Flag = FALSE$
THEN
act1 : $ConductionNodeState(B) := TRUE$
act2 : $CCSpeed\_CCTime\_Flag := TRUE$
END

A new event *Update_CCSpeed_CCtime* is introduced that is the refinement of the event *HeartConduction*. The main functionality of this event is to update the current impulse propagation time and impulse propagation speed according to the progressive conduction flow using landmark nodes in the conduction network. The guards of this event are used to select a pair of nodes that belongs to the defined conduction paths, and to select the current

conduction speed and current conduction time from the given range at the present time. The actions of this event update the current conduction time and current conduction speed according to the selected nodes and path.

EVENT Update_CCSpeed_CCtime Refines HeartConduction
ANY $i, j, CSpeed, CTime$
WHERE
grd1 : $i \in ConductionNode$
grd2 : $j \in ConductionNode$
grd3 : $i \mapsto j \in ConductionPath$
grd4 : $CSpeed \in 0 \,..\, 500$
grd5 : $CTime \in 0 \,..\, 300$
grd6 : $CCSpeed\_CCTime\_Flag = TRUE$
grd7 : $HeartState = FALSE$
grd8 : $tic = CTime$
THEN
act1 : $CConductionTime(j) := CTime$
act2 : $CConductionSpeed(i \mapsto j) := CSpeed$
act3 : $CCSpeed\_CCTime\_Flag := FALSE$
END

In the CRT pacemaker modeling, we introduce total eighteen events, in which seventeen events are the refinement of the abstract events. For example, the event *PM_Pacing_On_RV* refines the abstract event *PM_Pacing_On_RV* by adding new guards and new actions. The guards of this event state that the actuator of the right ventricle is OFF; the current clock counter is equivalent to the atrioventricular interval (AVI), no immediate pacing is required, there is no pace in the left and right ventricles and the right ventricular sensor is OFF, or an immediate pacing is required in the right ventricle; the current clock counter is not 0; there is no pacing in the right ventricle; and the actuator and sensor of the right atrium are OFF. The actions of this event state that the actuator of the right ventricle is ON and it is paced, and the present time is stored for controlling the future pacing or sensing activity. In the other refined events, we have added the temporal requirements for specifying the desired pacing and sensing behaviors considering synchronization in the heart chambers (RA, RV and LV) by observing the heart behavior.

EVENT PM_Pacing_On_RV Refines PM_Pacing_On_RV
WHEN
grd1 : $PM\_Actuator\_RV = OFF$
grd2 : $((now = AVI \wedge Immd\_Pace\_RV = 0 \wedge No\_Pace\_LV\_RV = 0 \wedge PM\_Sensor\_RV = OFF) \vee Immd\_Pace\_RV = 1)$
grd3 : $\neg now = 0$
grd4 : $Pace\_RV = 0$
grd4 : $PM\_Actuator\_A = OFF \wedge PM\_Sensor\_A = OFF$
THEN
act1 : $PM\_Actuator\_RV := ON$
act2 : $Pace\_RV := 1$
act3 : $PSRecord := now$
END

A logical clock plays an important role for modeling the temporal requirements of the CRT pacemaker and the functional behavior of the heart. In order to design a logical clock, we introduce a new event *tic* that allows to increase the current clock counter by 1 ms. progressively. This event *tic* does not have any guard, but in the further refinements, we introduce a guard to control different functionalities of the CRT pacemaker and heart within the restricted time intervals.

EVENT tic
WHEN
THEN
act1 : $now := now + 1$
END

1.5.3 Second Refinement: Threshold and Heart Blocks

This refinement introduces the functional properties of abnormal behavior of the heart, such as heart blocks, and *threshold* is used to detect an appropriate intrinsic activity for each chamber by the CRT pacemaker. The heart block introduces perturbation in the heart conduction network that generates some troubles in the electrical impulse propagation, and it influences the normal behavior of the heart. We use the landmark nodes to show the different types of heart blocks in Fig. 1.2. For modeling the heart blocks, we define an enumerated set *HeartBlockSets* to present different types of blocks of the heart.

$$axm1 : partition(HeartBlockSets, \{SA\_nodal\_blocks\}, \{AV\_nodal\_blocks\}, \{Infra\_Hisian\_blocks\}, \{LBBB\_blocks\}, \{RBBB\_blocks\}, \{None\})$$

The sensors play an important role for sensing an intrinsic activity of the heart. The CRT pacemaker actuator delivers a stimulation for a short period of time by observing the sensed values and required safety margins. Each chamber of the heart has different range of threshold values that can be specified by the physiologist by monitoring the detected intrinsic activities. We define three constants STA_THR_A, STA_THR_RV and STA_THR_LV to hold a range of standard threshold values for the right atrium, right ventricle and left ventricle, respectively in $axm1 - axm3$. We define a new constraint using an axiom ($axm4$) to show that the threshold of the atrium chamber is less than the threshold of the left and right ventricles.

$$axm1 : STA\_THR\_A \in \mathbb{N}1$$
$$axm2 : STA\_THR\_RV \in \mathbb{N}1$$
$$axm3 : STA\_THR\_LV \in \mathbb{N}1$$
$$axm4 : STA\_THR\_A < STA\_THR\_LV \land STA\_THR\_A < STA\_THR\_RV$$

For modeling the dynamic properties of the closed-loop system in this refinement, we declare some new variables. These variables are: *HeartBlocks* – to show the different types of heart blocks; *C_Thr* – to hold the current sensing value that is produced by the heart, and it can be used by the CRT pacemaker for detecting an intrinsic activity of the heart; *Thr_State_A* – a boolean state of the right atrium; *Thr_State_RV* – a boolean state of the right ventricle; and *Thr_State_LV* – a boolean state of the left ventricle. Three

state variables are defined for each chamber (RA, RV and LV) to synchronize and to maintain the order of sensing activities. We introduce three safety properties using invariants ($inv4$, $inv5$ and $inv6$) that state that the sensor of each chamber is OFF when the detected sensor value is greater than or equal to the standard threshold value and boolean state of the chamber is $TRUE$.

$inv1 : HeartBlocks \in HeartBlockSets$
$inv2 : C\_Thr \in \mathbb{N}$
$inv3 : Thr\_State\_A \in BOOL \land Thr\_State\_LV \in BOOL \land Thr\_State\_RV \in BOOL$
$inv4 : \forall i \cdot i \in \mathbb{N}_1 \land i \geq STA\_THR\_A \land Thr\_State\_A = TRUE$
$\Rightarrow PM\_Sensor\_A = OFF$
$inv5 : \forall i \cdot i \in \mathbb{N}_1 \land i \geq STA\_THR\_LV \land Thr\_State\_LV = TRUE$
$\Rightarrow PM\_Sensor\_LV = OFF$
$inv6 : \forall i \cdot i \in \mathbb{N}_1 \land i \geq STA\_THR\_RV \land Thr\_State\_RV = TRUE$
$\Rightarrow PM\_Sensor\_RV = OFF$

We introduce a set of events in this refinement to simulate the desired behavior of the heart block and detecting the intrinsic activities of the heart chambers using sensors of the CRT pacemaker. A sensor can detect an intrinsic activity when the threshold value of the detected signal is greater than or equal to the standard threshold constant. A set of events is introduced to model the possible behaviors of the heart blocks. A conduction disturbance in the heart is generated when an impulse produced from the sinus node (A) is blocked or delayed from depolarizing the atria known as SA block [15, 19]. To model the SA block, we introduce a new event *HeartConduction_Block_A_B_C* that is the refinement of the abstract event *HeartKO*. The guard of this event states that the landmark node (A or C) is not visited, the current impulse propagation time of the node (B or C) does not belong to the given range of the impulse propagation time, or the current impulse propagation speed of the path ($A \mapsto B$ or $A \mapsto C$) does not belong to the given range of impulse propagation speed. When the given guard satisfies then the actions show that the heart is in the abnormal state, and the heart has sinoatrial (SA) nodal block. Other heart block events are also the refinement of the abstract event *HeartKO*, which are modeled in the similar way.

EVENT HeartConduction_Block_A_B_C Refines HeartKO
WHEN
grd1 : $(ConductionNodeState(A) = FALSE) \lor$
$(ConductionNodeState(C) = FALSE) \lor$
$(CConductionTime(B) \notin ConductionTime(B)) \lor$
$(CConductionTime(C) \notin ConductionTime(C)) \lor$
$(CConductionSpeed(A \mapsto B) \notin ConductionSpeed(A \mapsto B)) \lor$
$(CConductionSpeed(A \mapsto C) \notin ConductionSpeed(A \mapsto C))$
THEN
act1 : $HeartState := FALSE$
act2 : $HeartBlocks := SA\_nodal\_blocks$
END

For modeling the threshold for detecting the appropriate intrinsic activities of the heart using the CRT pacemaker sensors, we do not introduce any new event in this refinement step. We introduce the functional behavior of threshold by strengthening the guards of the abstract events. The new added guards

are used to detect an intrinsic activity of the heart by comparing the value of sensed signal with the standard threshold value for the right atrium, right ventricle and left ventricle. For example, an event $PM\_Sensing\_Off\_A$ is the refinement of the abstract event $PM\_Sensing\_Off\_A$. In the refined event, we introduce a new variable Thr_A and some new guards ($grd8$, $grd9$). The guard $grd8$ declares the type of local variable (Thr_A), and the next guard $grd9$ compares the sensed value with the selected standard threshold value of the atria chamber and the sensed value is equivalent to the current sensing value that is produced by the heart model. The variable C_Thr plays an important role to synchronize the behavior of the CRT pacemaker and heart in the closed-loop model and to monitor an activity of the right atrium by comparing the sensed value with the selected threshold value. In this event, we introduce two extra actions ($act12$ and $act13$) to set $FALSE$ state to both the left and right ventricles. The rest of the guards and actions of this event are similar to the abstract event. We also modify other events related to the left and right ventricles to model the desired behavior of the sensors for synchronizing between the CRT pacemaker and heart models.

EVENT PM_Sensing_Off_A Refines
PM_Sensing_Off_A
ANY Thr_A
WHERE
grd1 : $PM\_Sensor\_A = ON$
grd2 : $PM\_Sensor\_RV = OFF$
grd3 : $PM\_Actuator\_A = OFF$
grd4 : $PM\_Actuator\_RV = OFF$
grd5 : $PM\_Sensor\_LV = OFF$
grd6 : $PM\_Actuator\_LV = OFF$
grd7 : $(now < AVI + VAI \wedge No\_Pace\_LV\_RV = 0 \wedge RV\_Delay\_AVI = 0 \wedge Delay\_Pace\_LV = 0 \wedge Immd\_Pace\_RV = 0 \wedge Immd\_Pace\_LV = 0) \vee (now < PSRecord + VAI)$
grd8 : $Thr\_A \in \mathbb{N}$
grd9 : $Thr\_A \geq STA\_THR\_A \wedge Thr\_A = C\_Thr$
THEN
act1 : $PM\_Sensor\_A := OFF$
act2 : $PSRecord := 0$
act3 : $now := 0$
act4 : $Immd\_Pace\_LV := 0$
act5 : $Delay\_Pace\_LV := 0$
act6 : $No\_Pace\_LV\_RV := 0$
act7 : $Immd\_Pace\_RV := 0$
act8 : $Pace\_LV := 0$
act9 : $Pace\_RV := 0$
act10 : $Pace\_A := 0$
act11 : $RV\_Delay\_AVI := 0$
act12 : $Thr\_State\_LV := FALSE$
act13 : $Thr\_State\_RV := FALSE$
END

1.5.4 Third Refinement: Refractory and Blanking Periods and Cellular Model

This is the last refinement of the closed-loop system that introduces cellular automata for modeling the concrete behavior of the heart, and the refractory

and blanking periods for modeling the concrete behavior of the CRT pacemaker. This final refinement of the heart model provides a simulation model that includes an impulse propagation at cellular level in the heart chambers. We define a set of constants and mathematical properties to formalize the desired behavior of the heart using cellular automata (see Fig. 1.3). Each biological cell can have one of the following states : *Active*, *Passive* and *Refractory*. To define the possible cell states, we declare an enumerated set *CellStates* in $axm1$. To simplify the modeling of cellular automata, we consider only two dimensional structure of the connected cells. In order to define the two dimensional structure, we declare a constant *NeighbouringCells* to specify a set of coordinated positions of the neighbouring cells using axioms ($axm2$ - $axm4$). A function $NEXT$ is declared to define the state of the neighbouring cells in $axm5$. Another new function *CellS* declares to map cell states to the neighbouring cells in $axm6$. A set of properties ($axm7$ - $axm9$) is introduced to model the desired functionalities of the cellular automata in two dimensions. The first property states that if the neighbouring cells are in the *Active* state then the next state of the neighbouring cells must be in the *Refractory* state. The second property states that if the neighbouring cells are in the $Refractory$ state then the next state of the neighbouring cells must be in the $Passive$ state. The last property states that if the neighbouring cells are in the $Passive$ state then the next state of the neighbouring cells must be either *Active* state or $Passive$ state.

$$
\begin{array}{l}
axm1 : partition(CellStates, \{PASSIVE\}, \{ACTIVE\}, \{REFRACTORY\}) \\
axm2 : x \in \mathbb{Z} \\
axm3 : y \in \mathbb{Z} \\
axm4 : NeighbouringCells = \\
\qquad \{\{x, y\}, \{x+1, y\}, \{x-1, y\}, \{x, y+1\}, \{x, y-1\}\} \\
axm5 : NEXT \in \mathbb{P}(NeighbouringCells) \rightarrow CellStates \\
axm6 : CellS \in NeighbouringCells \rightarrow CellStates \\
axm7 : (\forall\, param \cdot param \in \mathbb{P}(NeighbouringCells) \wedge (\forall\, m, n \cdot \{m, n\} \in param \wedge \\
\qquad CellS(\{m, n\}) = ACTIVE) \Rightarrow NEXT(param) = REFRACTORY) \\
axm8 : (\forall\, param \cdot param \in \mathbb{P}(NeighbouringCells) \wedge (\forall\, m, n \cdot \{m, n\} \in param \wedge \\
\qquad CellS(\{m, n\}) = REFRACTORY) \Rightarrow NEXT(param) = PASSIVE) \\
axm9 : (\forall\, param \cdot param \in \mathbb{P}(NeighbouringCells) \wedge (\forall\, m, n \cdot \{m, n\} \in param \wedge \\
\qquad CellS(\{m, n\}) = PASSIVE) \Rightarrow \\
\qquad NEXT(param) = ACTIVE) \vee NEXT(param) = PASSIVE
\end{array}
$$

For modeling the concrete behavior of actuators and sensors of the CRT pacemaker, we introduce the refractory and blanking periods[2] for the right atrium, right ventricle and left ventricle. These refractory and blanking periods play an important role to suppress device-generated artifacts and unwanted signal artifacts generated from intrinsic activities of the heart. Moreover, these periods also help to identify appropriate sensing events, and to prevent over sensing events in other chambers. To define the static behavior of the CRT pacemaker, we declare eight constants Atrial Refractory Period (ARP), Right Ventricular Refractory Period (RVRP), Left Ventricular Refractory Period (LVRP), Post Ventricular Atrial Refractory Period (PVARP),

[2]https://www.bostonscientific.com/content/dam/bostonscientific/quality/education-resources/english/ACL_Cross-Chamber_Blanking_20081219.pdf

Right Ventricular Blanking Period (RVBP), Left Ventricular Blanking Period (LVBP), A-Blank after Right Ventricular Activity (ABaRV), and A-Blank after Left Ventricular Activity (ABaLV) using axioms ($axm1$ - $axm8$).

$$
\begin{array}{l}
axm1 : ARP \in 30 \mathinner{..} 500 \\
axm2 : RVRP \in 20 \mathinner{..} 500 \\
axm3 : LVRP \in 20 \mathinner{..} 500 \\
axm4 : PVARP \in 50 \mathinner{..} 500 \\
axm5 : RVBP \in 5 \mathinner{..} 60 \\
axm6 : LVBP \in 5 \mathinner{..} 60 \\
axm7 : ABaRV \in 5 \mathinner{..} 150 \\
axm8 : ABaLV \in 5 \mathinner{..} 150
\end{array}
$$

The dynamic behavior of cell automata is defined by introducing four new variables. These four variables are m, n, *Transition* and *NextCellState*. The first two variables m and n are declared to model the coordinate positions in two dimensions of an active cell during an impulse propagation. The next variable *Transition* is defined as boolean type to enable transition between different states of the cells to model the behavior of a tissue. The last variable *NextCellState* is used to store the states of the neighbouring cells after each transition. In this refinement, we introduce six new safety properties. The first two safety properties state that the actuator and sensor of the right atrium become ON when the current clock counter elapses the refractory and blanking periods. To check the refractory period after pacing or sensing activity, we need to store the time of pacing or sensing activity of the ventricular chambers. In these safety properties, we use the variable *PSRecord* to record the time of previous occurrence of pacing or sensing event for initiating the refractory periods. The last four safety properties show that the actuators and sensors of the right and left ventricles become ON when the current clock counter elapses the refractory and blanking periods. It means that the sensing and pacing events always occur after elapsing the refractory and blanking periods.

$$
\begin{array}{l}
inv1 : m \in \mathbb{Z} \land n \in \mathbb{Z} \\
inv2 : Transition \in BOOL \\
inv3 : NextCellState \in CellStates \\
inv4 : PM\_Actuator\_A = ON \Rightarrow now \geq PSRecord + PVARP \land \\
\qquad now \geq PSRecord + RVRP \land now \geq PSRecord + LVRP \land \\
\qquad now \geq PSRecord + ABaLV \land now \geq PSRecord + ABaRV \\
inv5 : PM\_Sensor\_A = ON \Rightarrow now \geq PSRecord + PVARP \land \\
\qquad now \geq PSRecord + RVRP \land now \geq PSRecord + LVRP \\
\qquad \land now \geq PSRecord + ABaLV \land now \geq PSRecord + ABaRV \\
inv6 : PM\_Actuator\_RV = ON \Rightarrow now \geq ARP \land now \geq RVBP \\
inv7 : PM\_Actuator\_LV = ON \Rightarrow now \geq ARP \land now \geq LVBP \\
inv8 : PM\_Sensor\_RV = ON \Rightarrow now \geq ARP \land now \geq RVBP \\
inv9 : PM\_Sensor\_LV = ON \Rightarrow now \geq ARP \land now \geq LVBP
\end{array}
$$

For modeling an impulse propagation at cellular level in the heart for two dimensional structure, we define two events *HeartConduction_ Cellular* and *HeartConduction_ Next_ UpdateCell*. It should be noted that we define these events abstractly that can be further refined to formalize the concrete behavior of the cellular automata at tissue level corresponding to various states of

the cell. The event *HeartConduction_ Cellular* enables transition for switching state of the neighbouring cells of the conduction network by propagating electrical current. This event allows to set a boolean state as $TRUE$ of the *Transition*. The guards of this event state that there is a valid path between two landmark nodes that belongs to a set of pairs of the conduction network; the current impulse propagation speed and time flag is $TRUE$ that allows to synchronize and for preserving the desired behavior of the heart for capturing the current values of the impulse propagation time and impulse propagation speed for each node; a set of neighbouring cells is selected in the two dimensional structure; the current cell position (m,n) is in the passive state and it is equivalent to the next state of the neighbouring cells; and the current transition state is $FALSE$.

EVENT HeartConduction_Cellular
ANY $p, q, param$
WHERE
grd1 : $p \mapsto q \in ConductionPath$
grd2 : $CCSpeed\_CCTime\_Flag = TRUE$
grd3 : $param \in \overline{\mathbb{P}}(\{\{m,n\}, \overline{\{m+1,n\}}, \{m-1,n\}, \{m,n+1\}, \{m,n-1\}\})$
grd4 : $\{m,n\} \in dom(CellS) \wedge CellS(\{m,n\}) \in \{PASSIVE, ACTIVE, REFRACTORY\}$
grd5 : $NextCellState = CellS(\{m,n\})$
grd6 : $Transition = FALSE$
THEN
act1 : $Transition := TRUE$
END

The event *HeartConduction_ Next_ UpdateCell* is a new event for calculating the state of neighbouring cells and to update the position of the current cell (m, n). The guards of this event state that a set ($param$) of neighbouring cells is selected in two dimensional structure; the selected set is a domain of the function $NEXT$; and the current transition state is $TRUE$. The actions of this event calculates the state of neighbouring cells that is assigned to *NextCellState*, sets the current transition state as $FALSE$, and updates the current cell (m, n) nondeterministically to propagate an impulse in the conduction network.

EVENT HeartConduction_Next_UpdateCell
ANY $param$
WHERE
grd1 : $param \in \mathbb{P}(\{\{m,n\}, \{m+1,n\}, \{m-1,n\}, \{m,n+1\}, \{m,n-1\}\})$
grd2 : $param \in dom(NEXT)$
grd3 : $Transition = TRUE$
THEN
act1 : $NextCellState := NEXT(param)$
act2 : $Transition := FALSE$
act3 : $m :\in \{m-1, m, m+1\}$
act4 : $n :\in \{n-1, n, n+1\}$
END

For modeling the concrete temporal behavior of the CRT pacemaker considering the refractory and blanking periods, we introduce many guards in several events. For example, we introduce an event *PM_ Pacing_ On_ A* that

refines the abstract event $PM\_Pacing\_On\_A$ by strengthening the guards. The guards of this event state that the actuator of the right atrium is OFF; the current clock counter elapses the AVI + VAI or PSRecord + VAI interval considering the possible scenarios related to delay pacing, no pacing and immediate pacing for either one chamber (LV or RV) or both chambers (LV and RV); pacing state of the atrium chamber is 0; and the current clock counter is elapsed the refractory and blanking periods after detecting an activity of pacing or sensing in the heart. The actions of this event state that the actuator of the right atrium generates a small electrical current to pace, and to set the pacing state of the atrium chamber for synchronizing the pacing and sensing events of the other heart chambers.

EVENT PM_Pacing_On_A Refines PM_Pacing_On_A
WHEN
grd1 : $PM\_Actuator\_A = OFF$
grd2 : $(now = AVI + VAI \wedge No\_Pace\_LV\_RV = 0 \wedge Delay\_Pace\_LV = 0)$
$\vee$
$(now = PSRecord + VAI \wedge (Delay\_Pace\_LV = 2 \vee Delay\_Pace\_LV = 1$
$\vee No\_Pace\_LV\_RV = 1 \vee RV\_Delay\_AVI = 1 \vee Immd\_Pace\_RV = 1$
$\vee Immd\_Pace\_LV = 1))$
grd3 : $Pace\_A = 0$
grd4 : $now \geq PSRecord + PVARP$
grd5 : $now \geq PSRecord + RVRP \wedge now \geq PSRecord + LVRP$
grd6 : $now \geq PSRecord + ABaRV \wedge now \geq PSRecord + ABaLV$
THEN
act1 : $PM\_Actuator\_A := ON$
act2 : $Pace\_A := 1$
END

We also introduce a new guard in the event *tic* to describe the correct temporal behavior of the sensing and pacing events. The provided guard synchronizes the pacing and sensing activities of the CRT pacemaker according to the biventricular sensing and pacing (BiSP) requirements by monitoring the heart functionalities. The *tic* event contains a very large guard to represent all the possible timing requirements (see Section 1.4). We present below only a slice of the complete formalized timing requirements to show the progress of the event *tic* as the guard conditions according to Fig. 1.5(b).

EVENT tic
WHEN
grd1: $(now < AVI \wedge PM\_Sensor\_LV = OFF \wedge PM\_Sensor\_RV = OFF \wedge$
$No\_Pace\_LV\_RV = 1)$
$\vee$
$(now \geq AVI \wedge now < PSRecord + PVARP \wedge PM\_Sensor\_LV = OFF \wedge$
$PM\_Sensor\_RV = OFF \wedge No\_Pace\_LV\_RV = 1 \wedge$
$(Pace\_RV = 2 \vee Thr\_State\_RV = TRUE))$
$\vee$
$(now \geq PSRecord + PVARP \wedge now < PSRecord + VAI \wedge$
$PM\_Sensor\_LV = OFF \wedge PM\_Sensor\_RV = OFF \wedge No\_Pace\_LV\_RV = 1$
$\wedge PM\_Sensor\_A = ON)$
$\vee$
...
...
THEN
act1 : $now := now + 1$
END

1.5.5 Model Validation and Analysis

This section presents the proof statistics of the developed closed-loop model, and the validity of functional behavior of the CRT pacemaker and heart by simulating the proved formal model in ProB [21]. The Event-B language supports *consistency checking* and *model analysis*, in which the *consistency checking* shows that the defined events always preserve the given invariants and the refinement checking ensures that the concrete machine is a valid refinement of an abstract machine, and the *model analysis* is used to animate the formal specification to check the required functional behavior of the system. Model validation plays an important role for gaining confidence in the developed formal model in order to check the consistency with system requirements. In addition, the ProB tool also allows *automated consistency checking* and *constraint-based checking*. This tool helps to identify possible deadlocks and hidden properties that may be exposed by the generated proof obligations. In our work, the ProB tool is used to animate the closed-loop model of the CRT pacemaker and heart at each refinement level to check the required functional behavior by considering numerous scenarios for validating the developed formal models. This tool effectively assists us in finding potential problems and to improving the guard conditions in each layer of the refinement. To use the ProB model checker at each refinement level, we were able to animate all the abstract and a series of refined models to prove the absence of errors (no counter example). It should be noted that the ProB tool automatically imports static and dynamic properties, including safety properties, of the closed-loop system that are used for consistency checking and model checking to discover the violation of the given safety properties against the formalized system behavior.

Model	**Total number of POs**	**Automatic Proof**	**Interactive Proof**
Abstract Model	29	25(86%)	4(14%)
First Refinement	261	245(94%)	16(6%)
Second Refinement	35	26(74%)	9(26%)
Third Refinement	70	69(99%)	1(1%)
Total	**395**	**365(92%)**	**30(8%)**

TABLE 1.3: Proof Statistics

Table 1.3 shows the proof statistics of the closed-loop model developed in the RODIN tool [32]. This formal development generates 395(100%) proof obligations (POs), in which 365(92%) POs are proved automatically with the help of inbuilt RODIN provers, and the remaining 30(8%) POs are proved interactively by simplifying the predicates using the Rodin provers. It should be noted that the simplifying predicates are quite simple. An integration of the heart model and CRT pacemaker model generates some extra POs related to the joint behavior of the closed-loop system and by sharing some common variables by both the heart and CRT models. For example, the current clock counter variable (*now*) is shared, which has been used in the events of the CRT pacemaker and heart models. The CRT pacemaker shows functional properties

of pacing and sensing modes under the virtual biological environment of the heart. The heart model represents normal and abnormal states of the heart, which is estimated by the physiological analysis. A list of safety properties is introduced in the incremental refinements to guarantee the correctness of the functional requirements of the closed-loop model of the heart and CRT pacemaker.

1.6 DISCUSSION

This paper presents an approach for modeling the closed-loop trustworthy cyber physical systems (TCPS). The main goal of this work is to provide a novel modeling technique that helps to community to develop a closed-loop model of the trustworthy cyber physical system (TCPS) and virtual environment. The closed-loop modeling approach is used to analyze the system requirements of TCPS for verification, validating assumptions, identifying new emergent behavior, identifying undesired state of the system, strengthening the given system requirements, *what-if* analysis and to check inconsistencies in the given system. If any fault presents in the system then the formalized system does not behave appropriately as per the environmental requirements. In fact, we need precise knowledge of the error states for finding any effective heuristics. All the relevant states that lead to error/hazard states that can be exploited to discover the desired correct functionalities or improving the exiting system requirements. The combined closed-loop model of the TCPS and virtual environment generates several new POs that do not appear in both the TCPS model and environment model, independently. All the generated POs of the closed-loop system are produced according the required behavior of the TCPS and the defined environmental requirements. Moreover, the generated POs allow for strengthening the guards to make system deterministic to meet the stakeholder needs by describing the concrete behavior of the TCPS and by removing flaws in the developing model of the TCPS. This can be an effective approach to guarantee the correctness of the functional behavior and requirements of the TCPS. Moreover, the result of this approach can be viable to assist for meeting the requirement of the certification standards. In fact, the developed closed-loop model is not limited only for analyzing the system requirements, but we can use it for various purposes during the system development, such as automated code generation, automated test case generation, use as an evidence for safety assurance cases and to assist for evaluating the product for certification purpose.

In this chapter, we formalized the closed-loop system of the CRT pacemaker and heart using stepwise incremental refinements. The stepwise refinement approach is a well known technique for developing the dependable systems, and was championed by Harlan Mills in his work on Box-Structures [28]. For developing the closed-loop system, we used the formal development of the CRT pacemaker [34] and the formal development of the heart [26] as a basis for this work. In this work, we formalized the closed-loop model of the CRT

pacemaker and heart incrementally using the set theoretical notations to check the desired behavior of the CRT pacemaker, and to find the missing system requirements by analyzing the given system requirements and environmental requirements. The formalized environment shows an environmental conditions that present conditional properties that help to design a system whether the system provides an appropriate action or solution as a feedback according to the environmental situation. The closed-loop model of the CRT pacemaker and heart allows us to evaluate whether the CRT pacemaker provides an appropriate therapy for any arrhythmias. We have provided a list of safety properties in each refinement to verify the correctness of the defined system behavior. The given safety properties guarantee that all possible executions of the closed-loop system are safe, if the generated POs are successfully discharged. The results of this experiment show that the closed-loop model has potential to verify the TCPS requirements, to identifying new emergent behavior, checking requirements consistency and strengthening the given system requirements. Although the TCPS modeling and environment modeling can be formalized in many ways by using various techniques, the modeling methodologies for developing the closed-loop system and to analyze the system requirements described in this chapter would still remain the same.

1.7 RELATED WORK

Requirement analysis is an important and challenging phase in the software development lifecycle for eliciting, analyzing and recording the system requirements according to stakeholder needs. The recorded requirements must be precisely defined, unambiguous, formally verified, documented and traceable by covering the possible requirements. Most common errors related to software requirements are listed in [22] that presents a detailed survey on the types of errors and associated root causes for failing the critical systems. These root causes can be human error, process flaws and program faults. There are several popular techniques for requirement analysis, such as simulation and prototyping. A prototype is an early model of the a product that contains partial features of the system when the system requirements are unclear or indefinite [9]. An approach is proposed in [12] for constructing a prototype using an algebraic specification language and for executing the developed specification. A run-time technique for monitoring the system requirements is presented in [11] that allows to monitor violating properties of the system behavior and to adapt the new dynamic behaviors by satisfying the higher-level system goals.

Ian et al. [13] proposed an approach to specify the requirements and environment of the system, and then capture the assumptions on the physical components by recording rely-conditions, and later derive a specification of the computational part of the control system. Moreover, the proposed approach does not claim that the developed system can be perfectly safe, but it claims that the proposed approach will help to identify the assumptions

for physical components of the system and to ensure that the requirements are formally documented. Kishi et al. [30] proposed environment modeling approach for designing the embedded systems and to rectifying the possible bugs. A new language is proposed for modeling an environment and to specify the dynamic behavior for simulating the virtual environment in [18]. Several other papers [20, 8] also reported work on the environment modeling and simulation using different techniques.

An environment modeling is an essential approach that is not limited for simulation only, but it can be used for checking the desired system requirements and to use during the system testing. Auguston et al. [3] proposed environmental behavior model based on Attributed Event Grammar for testing the embedded systems. Heisel et al. [14] discussed the testing approach by using the specified requirements of the system and environment models in the UML state machines. Based on environmental constraints, a testing approach is presented in [10] for describing the behavior of synchronous reactive software using temporal logic.

Méry et al. [26] proposed the first heart model considering all the required normal and abnormal behaviors in the Event-B modeling language. Formal techniques based a closed-loop model of the cardiac pacemaker, for one-and two-electrode, and heart is presented in [33, 29]. This approach is based on formal modeling and verification of the cardiac pacemaker. We have adopted this approach for modeling the closed-loop system of the TCPS and environment, and we have used a case study, the CRT pacemaker and heart, as an example to demonstrate the results and benefits.

1.8 CONCLUSION

The trustworthy cyber-physical systems (TCPS) are dependable critical systems that play a major role in several industrial sectors, like avionic, transportation, medical, space and automotive domains. An increasing demand for new technology and growing interest, for developing the TCPS within a limited time period, allow to industrial sectors to adopt the commercial firmware and software. This rapid adoption of firmware and softwares for developing the TCPS increases vulnerabilities that may lead to devastating system failures, including loss of life and economical damage. Mostly, the product design and engineering flaws, including firmware problems, are identified as the main causes of the TCPS failures.

In this chapter, we have discussed an approach for requirement analysis using the closed-loop modeling. We have developed the closed-loop model, an integration of TCPS and environment, to identify new emergent behavior, missing system requirements, validating assumptions, identifying undesired state of the system, strengthening the given requirements, *what-if* analysis and to check inconsistencies in the given TCPS. In our work, for modeling both the TCPS and environment models by supporting the *correct-by-construction* approach, we use the Event-B modeling language and Rodin tools [32, 2]

for managing, developing, verifying and simulating the desired requirements under the given safety constraints.

To demonstrate the effectiveness of the closed-loop modeling approach, our goal is to exemplify by integrating the formal models of the CRT pacemakers and heart to model the closed-loop system for verifying the desired behavior under relevant safety properties, and to be able to guarantee the correctness of the functional behavior of the CRT pacemaker. Our experiment involves formalizing and reasoning about the behavior of a CRT pacemaker, allows to actuators to pace appropriately whenever required, under normal and abnormal heart conditions. A set of general and patient condition-specific temporal requirements is specified that is used to formalize an interactive and physiologically relevant closed-loop model for verifying the basic and complex operations of the CRT pacemaker. With the use of model checkers, we demonstrate that the proposed system is capable of testing the common and complex heart conditions across a variety of CRT pacemaker modes. This system is a step toward a modeling approach for medical cyber-physical systems with the patient-in-the-loop.

Applying the closed-loop approach for developing the TCPS has many benefits: the exposure of errors which might have not been detected without the environment model; to validate the given assumptions; to increase confidence and decrease the failure risks; and to promote the use of closed-loop modeling approach for identifying the emergent behavior and improving the system requirements for developing the quality TCPS. Moreover, this approach also allows to consider the feedback from domain experts by simulating the desired behavior. There are scientific and legal applications for using the closed-loop modeling approach for better understanding, identifying the desired functional behavior, improving the system requirements and to meet certification requirements for developing the dependable TCPS.

Bibliography

[1] IEEE Standard Glossary of Software Engineering Terminology. *IEEE Std 610.12-1990*, pages 1–84, Dec 1990.

[2] Jean-Raymond Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, New York, NY, USA, 1st edition, 2010.

[3] Mikhail Auguston, James Bret Michael, and Man-Tak Shing. Environment behavior models for automation of testing and assessment of system safety. *Information and Software Technology*, 48(10):971 – 980, 2006. Advances in Model-based Testing.

[4] S. Serge Barold, Roland X. Stroobandt, and Alfons F. Sinnaeve. *Cardiac Pacemakers Step by Step*. Futura Publishing, 2004. ISBN 1-4051-1647-1.

[5] V. N. Bayes de Luna A., Batcharov and M. Malik. *"The morphology of the Electrocardiogram" in The ESC Textbook of Cardiovascular Medicine*. Blackwell Publishing Ltd., 2006.

[6] Jr. Bubenko, J.A. Challenges in requirements engineering. In *Requirements Engineering, Proceedings of the Second IEEE International Symposium on*, pages 160 – 162, March 1995.

[7] M.C. Bujorianu and H. Barringer. An integrated specification logic for cyber-physical systems. In *Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on*, pages 291–300, June 2009.

[8] Keungsik Choi, Sungchul Jung, Hyunjung Kim, and Doo hwan Bae. Uml-based modeling and simulation method for mission-critical real-time embedded. In *System DevelopmentĂİ, IASTED Conf. on Software Engineering 2006, 2006, 160âĂŞ165. Mittal, Zeigler and De la Cruz*, 2006.

[9] Alan M. Davis. Operational prototyping: A new development approach. *IEEE Softw.*, 9:70–78, September 1992.

[10] L. du Bousquet, F. Ouabdesselam, J.-L. Richier, and N. Zuanon. Lutess: A specification-driven testing environment for synchronous software. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 267–276, New York, NY, USA, 1999. ACM.

[11] S. Fickas and M. S. Feather. Requirements monitoring in dynamic environments. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, RE '95, pages 140–147, Washington, DC, USA, 1995. IEEE Computer Society.

[12] Joseph Goguen and Jose Meseguer. Rapid prototyping: in the obj executable specification language. *SIGSOFT Softw. Eng. Notes*, 7:75–84, April 1982.

[13] IanJ. Hayes, MichaelA. Jackson, and CliffB. Jones. Determining the specification of a control system from that of its environment. In Keijiro Araki, Stefania Gnesi, and Dino Mandrioli, editors, *FME 2003: Formal Methods*, volume 2805 of *Lecture Notes in Computer Science*, pages 154–169. Springer Berlin Heidelberg, 2003.

[14] Maritta Heisel, Denis Hatebur, and Thomas Santen Dirk Seifert. Testing against requirements using uml environment models. In *in Fachgruppentreffen Requirements Engineering und Test, Analyse & Verifikation*, pages 28–31, 2008.

[15] Robert Plonsey Jaakko Malmivuo. *Bioelectromagnetism*. Oxford University Press, 1995. ISBN 0-19-505823-2.

[16] Societe franÃğaise cardiologie Jean-Yves Artigou, Jean-Jacques Monsuez. *Cardiologie et maladies vasculaires*. Elsevier Masson, 2006.

[17] John von Neumann. Theory of Self-Reproducing Automata. University of Illinois Press, 1966 Edited by Arthur W. Burks.

[18] Gabor Karsai, Sandeep Neema, and David Sharp. Model-driven architecture for embedded software: A synopsis and an example. *Sci. Comput. Program.*, 73(1):26–38, September 2008.

[19] M. Gabriel Khan. *Rapid ECG Interpretation*. Humana Press, 2008.

[20] C. Kreiner, C. Steger, and R. Weiss. Improvement of control software for automatic logistic systems using executable environment models. In *Euromicro Conference, 1998. Proceedings. 24th*, volume 2, pages 919–923 vol.2, Aug 1998.

[21] Michael Leuschel and Michael Butler. *ProB: A Model Checker for B*, pages 855–874. LNCS. Springer, 2003.

[22] R.R. Lutz. Analyzing software requirements errors in safety-critical, embedded systems. In *Requirements Engineering, 1993., Proceedings of IEEE International Symposium on*, pages 126–133, Jan 1993.

[23] William H. Maisel, Megan Moynahan, Bram D. Zuckerman, Thomas P. Gross, Oscar H. Tovar, Donna-Bea Tillman, and Daniel B. Schultz. Pacemaker and ICD Generator Malfunctions: Analysis of Food and Drug Administration Annual Reports. *JAMA*, 295(16):1901–1906, 2006.

[24] John McDermid. *Software Engineer's Reference Book*. CRC Press, Inc., Boca Raton, FL, USA, 1991.

[25] Dominique Méry and Neeraj Kumar Singh. Technical Report on Formalisation of the Heart using Analysis of Conduction Time and Velocity of the Electrocardiography and Cellular-Automata. Technical report, http://hal.inria.fr/inria-00600339/en/, 2011.

[26] Dominique Méry and Neeraj Kumar Singh. Formalization of heart models based on the conduction of electrical impulses and cellular automata. In Zhiming Liu and Alan Wassyng, editors, *Foundations of Health Informatics Engineering and Systems*, volume 7151 of *Lecture Notes in Computer Science*, pages 140–159. Springer Berlin Heidelberg, 2012.

[27] Dominique Méry and Neeraj Kumar Singh. Analyzing requirements using environment modelling. In *Digital Human Modeling. Applications in Health, Safety, Ergonomics and Risk Management - 5th International Conference, DHM 2015, Held as Part of HCI International*, 2015.

[28] Harlan D. Mills. Stepwise refinement and verification in box-structured systems. *IEEE Computer*, 21(6):23–36, 1988.

[29] Dominique MŐry and NeerajKumar Singh. Closed-loop modeling of cardiac pacemaker and heart. In Jens Weber and Isabelle Perseil, editors, *Foundations of Health Information Engineering and Systems*, volume 7789 of *Lecture Notes in Computer Science*, pages 151–166. Springer Berlin Heidelberg, 2013.

[30] N. Noda and T. Kishi. Aspect-oriented modeling for embedded software design. In *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*, pages 342–349, Dec 2007.

[31] International Standard Organization. ISO 26262: Road Vehicles – Functional Safety, 2011.

[32] Project RODIN. Rigorous open development environment for complex systems. http://rodin-b-sharp.sourceforge.net/, 2004.

[33] Neeraj Kumar Singh. *Using Event-B for Critical Device Software Systems*. Springer-Verlag GmbH, 2013.

[34] NeerajKumar Singh, Mark Lawford, ThomasS.E. Maibaum, and Alan Wassyng. Formalizing The Cardiac Pacemaker Resynchronization Therapy. In *Digital Human Modeling. Applications in Health, Safety, Ergonomics and Risk Management, HCII 2015*, LNCS. Springer International Publishing, 2015.

[35] Ian Sommerville and Pete Sawyer. *Requirements Engineering: A Good Practice Guide*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.