11-12-2015

High-Confidence Medical Device Software Development

Zhihao Jiang
University of Pennsylvania, zhihaoj@seas.upenn.edu

Rahul Mangharam
University of Pennsylvania, rahulm@seas.upenn.edu

Recommended Citation
Z. Jiang and R. Mangharam. High-Confidence Medical Device Software Development. Foundations and Trends in Electronic Design Automation, Vol. 9, No. 4 (2015) 309–391.

High-Confidence Medical Device Software Development

Abstract
The design of bug-free and safe medical device software is challenging, especially in complex implantable devices. This is due to the device's closed-loop interaction with the patient's organs, which are stochastic physical environments. The life-critical nature and the lack of existing industry standards to enforce software validation make this an ideal domain for exploring design automation challenges for integrated functional and formal modeling with closed-loop analysis. The primary goal of high-confidence medical device software is to guarantee the device will never drive the patient into an unsafe condition even though we do not have complete understanding of the physiological plant. There are two major differences between modeling physiology and modeling man-made systems: first, physiology is much more complex and less well-understood than man-made systems like cars and airplanes, and spans several scales from the molecular to the entire human body. Secondly, the variability between humans is orders of magnitude larger than that between two cars coming off the assembly line. Using the implantable cardiac pacemaker as an example of closed-loop device, and the heart as the organ to be modeled, we present several of the challenges and early results in model-based device validation. We begin with detailed timed automata model of the pacemaker, based on the specifications and algorithm descriptions from Boston Scientific. For closed-loop evaluation, a real-time Virtual Heart Model (VHM) has been developed to model the electrophysiological operation of the functioning and malfunctioning (i.e., during arrhythmia) hearts. By extracting the timing properties of the heart and pacemaker device, we present a methodology to construct timed-automata models for formal model checking and functional testing of the closed-loop system. The VHM's capability of generating clinically-relevant response has been validated for a variety of common arrhythmias. Based on a set of requirements, we describe a framework of Abstraction Trees that allows for interactive and physiologically relevant closed-loop model checking and testing for basic pacemaker device operations such as maintaining the heart rate, atrial-ventricle synchrony and complex conditions such as avoiding pacemaker-mediated tachycardia. Through automatic model translation of abstract models to simulation-based testing and code generation for platform-level testing, this model-based design approach ensures the closed-loop safety properties are retained through the design toolchain and facilitates the development of verified software from verified models. This system is a step toward a validation and testing approach for medical cyber-physical systems with the patient-in-the-loop.

Disciplines
Computer Engineering | Electrical and Computer Engineering

Foundations and Trends® in Electronic Design Automation
Vol. 9, No. 4 (2015) 309–391

DOI: 10.1561/1000000040

now
the essence of knowledge

High-Confidence Medical Device Software Development

Zhihao Jiang and Rahul Mangharam
University of Pennsylvania
United States

Contents

Abstract

The design of bug-free and safe medical device software is challenging, especially in complex implantable devices. This is due to the device's closed-loop interaction with the patient's organs, which are stochastic physical environments. The life-critical nature and the lack of existing industry standards to enforce software validation make this an ideal domain for exploring design automation challenges for integrated functional and formal modeling with closed-loop analysis. The primary goal of high-confidence medical device software is to guarantee the device will never drive the patient into an unsafe condition even though we do not have complete understanding of the physiological plant.

There are two major differences between modeling physiology and modeling man-made systems: first, physiology is much more complex and less well-understood than man-made systems like cars and airplanes, and spans several scales from the molecular to the entire human body. Secondly, the variability between humans is orders of magnitude larger than that between two cars coming off the assembly line.

Using the implantable cardiac pacemaker as an example of closed-loop device, and the heart as the organ to be modeled, we present several of the challenges and early results in model-based device validation. We begin with detailed timed automata model of the pacemaker, based on the specifications and algorithm descriptions from Boston Scientific. For closed-loop evaluation, a real-time Virtual Heart Model (VHM) has been developed to model the electrophysiological operation of the functioning and malfunctioning (i.e., during arrhythmia) hearts. By extracting the timing properties of the heart and pacemaker device, we present a methodology to construct timed-automata models for formal model checking and functional testing of the closed-loop system. The VHM's capability of generating clinically-relevant response has been validated for a variety of common arrhythmias. Based on a set of requirements, we describe a framework of Abstraction Trees that allows for interactive and physiologically relevant closed-loop model checking and testing for basic pacemaker device operations such as maintaining the heart rate, atrial-ventricle synchrony and complex conditions such as avoiding pacemaker-mediated tachycardia.

Through automatic model translation of abstract models to simulation-based testing and code generation for platform-level testing, this model-based design approach ensures the closed-loop safety properties are retained through the design toolchain and facilitates the development of verified software from verified models. This system is a step toward a validation and testing approach for medical cyber-physical systems with the patient-in-the-loop.

Z. Jiang and R. Mangharam. *High-Confidence Medical Device Software Development*. Foundations and Trends® in Electronic Design Automation, vol. 9, no. 4, pp. 309–391, 2015. DOI: 10.1561/1000000040.

1

Medical Devices: Current State and Challenges

The medical device market is worth $289 billion, of which $110 billion is from the US alone, with this number projected to reach $133 billion in 2016. Examples include everything from adhesive bandages, stents, artificial joints, drug infusion pumps to surgical robots, implantable cardiac pacemakers, and devices still undergoing basic research like the artificial pancreas. To take one example of the societal impact of medical devices, an estimated 3 million people worldwide have implanted cardiac pacemakers (a heart rate adjustment device), with 600,000 added annually. Clinical trials have presented evidence that patients implanted with cardiac defibrillators (another heart rate adjustment device) have a mortality rate reduced by up to 31%. Implanted cardiac pacemakers and defibrillators have approximately 80,000-100,000 lines of software code which essentially makes all sensing, control and actuation decisions autonomously within the human body, over the 5-7 year device lifetime [1]. With the increasing complexity of combining hardware and software in a large class of these life-saving technologies, there is an urgent need for approaches to rigorously validate the device and therapy to be safe and efficacious.

[1] Paul L. Jones. Senior Systems/Software Engineer, Office of Science and Engineering Laboratories, U. S. FDA. Personal communication, 2010.

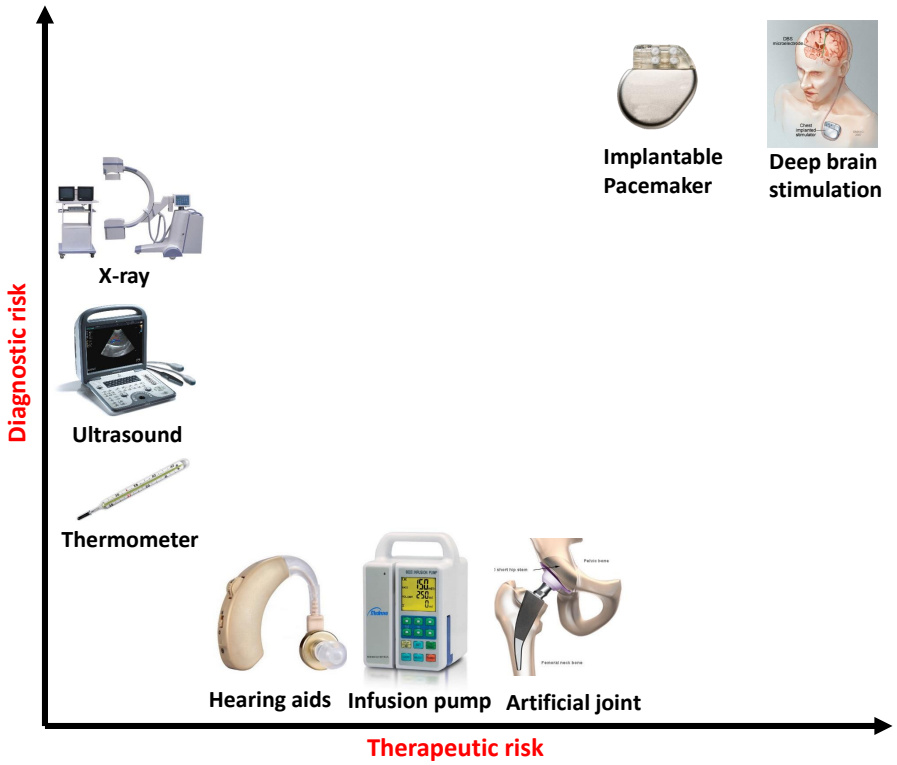


Figure 1.1: Current medical devices across a range of diagnostic and therapeutic risk. Implantable software-controlled devices such as the pacemaker and defibrillator which operate in a closed-loop of sensing, control and actuation are amongst the highest risk

The US Food and Drug Administration defines a medical device as an instrument, apparatus, implement, machine, or implant which is:

- intended for use in the diagnosis of disease or other conditions, or in the cure, mitigation, treatment, or prevention of disease, in humans or other animals, or
- intended to affect the structure or any function of the human body or other animals, and which does not achieve any of its primary intended purposes through chemical action and which is not dependent upon being metabolized for the achievement of any of its primary intended purposes."

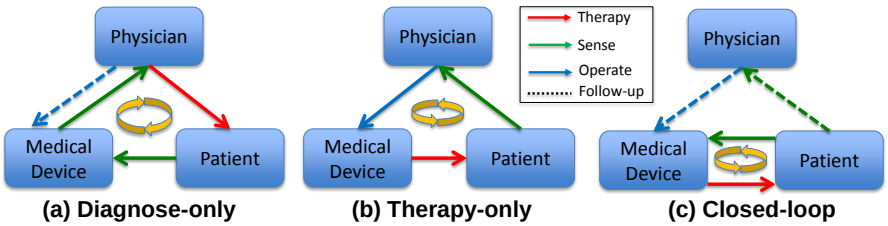


Figure 1.2: Diagnostic-only and therapy-only devices do not interact with the patient in direct closed-loop. The physician is responsible for the diagnostic and/or therapeutic decisions. However in closed-loop medical devices, the devices interact with the patient in closed-loop and have to make therapeutic decisions based on their own diagnosis.

In general, medical devices are categorized according to their risk factors - Class I, Class II and Class III, corresponding to low-risk, medium-risk and high-risk devices (Food and Administration [2014]). Fig. 1.1 gives an intuitive description of medical devices examples across a range of diagnostic and therapeutic risk.

1.1 Closing the Device-Patient Loop

Medical devices operate across a range of invasiveness and intervention with the patient in the loop. For diagnostic-only devices, like an X-ray machine, the physician operates the device to obtain patient data. Upon interpretation of the data, the physician performs diagnosis followed by delivery of proper therapy to the patient (Fig. 1.2.(a)). For therapy-only devices, e.g. a drug infusion pump, the physician configures the device infrequently based on prior diagnosis of the patient so the device executes the therapy on the patient (Fig. 1.2.(b)). We denote these devices as **Open-loop Medical Devices** as there is no direct feedback loop between the patient and the device. For open-loop devices, the device operates under the supervision of professionally-trained physicians. The device's safety is mostly determined by how accurately it provides information to the physicians or how faithfully it operates as instructed by the physicians.

There is a class of devices with both diagnostic and therapeutic functions, i.e. implantable cardiac devices to treat cardiac arrhythmia, deep brain stimulation devices (Coffey [2009]) to treat Parkinson's disease and artificial

pancreas to treat Type-1 diabetes. These devices capture and diagnose the patient's physiological conditions from sensory data, *and* deliver therapy in response (Fig. 1.2.(c)). These devices usually operate (semi-) autonomously with very little human intervention. Although therapies can be delivered more timely with these devices, malfunctions or inappropriate therapies from these devices also cannot be corrected timely, which can cause serious adverse effects on patients' health. Therefore, these devices are usually classified into the highest risk category and undergo the most stringent regulation. We denote them as **Closed-loop Medical Devices**.

There are multiple challenges to develop safe and effective closed-loop medical devices:

1.1.1 Closed-loop Interactions with Complex Physiology

When using open-loop medical devices, the diagnosis and therapy decisions are made by medical professionals, who have expert knowledge of human physiology. Therefore they are able to identify adverse health conditions and adjust the therapy accordingly. On the other hand, closed-loop medical devices have to make both the diagnosis and therapy decisions on their own. The domain expertise required to make those decisions has to be programmed into the device. It is impossible to encode all the knowledge of human physiology into the device. Therefore, for unanticipated physiological conditions, when the appropriate response has not been programmed into the device, the device may deliver inappropriate therapy which can have an adverse effect on patient's health.

Technological development of materials, sensors, embedded computing, energy storage, communications and packaging usher new closed-loop therapies (e.g. deep brain stimulation). While the spectrum of closed-loop interactions between the device and the human physiology may not be fully understood, the challenge is to ensure the device never drives the patient into an adverse state under all physiological conditions. Furthermore, the incremental addition of new therapies in legacy devices (e.g. cardiac rhythm therapy), may result in conflicted diagnostics and behavior of the device for well-understood behaviors and result in inappropriate and unsafe operations.

1.1.2 Limited Diagnostic and Therapeutic Functions

One fundamental rationale behind closed-loop medical devices is to enable patients to live their lives normally with limited explicit interaction with the device, and also with minimal physician supervision. In fact, a large number of closed-loop medical devices are autonomous implantable devices. As a result, the sensing and therapy capabilities of these devices are limited, in order to minimize power consumption, heat dissipation and invasiveness. Limited sensing capabilities, and hence limited observability, may cause misdiagnosis as the device may be unable to distinguish the source between two sensed signals from different conditions that now seem similar and result inappropriate therapy. Due to limited therapeutic capabilities, there exists sub-optimal physiological conditions that are untreatable. The device may even drive the body to a less optimal state by over-treating the patient by preempting the body's natural response. In later chapters, we will describe examples in which an untreatable condition is deteriorated into an adverse condition due to the device interaction.

1.1.3 Software-related Medical Device Recalls

Due to the complexity of the diagnostic and therapeutic functions of the closed-loop devices, these functions are mostly controlled by their software components. Software embedded in a medical device, unlike electrical and mechanical components, does not fail due to corrosion, fatigue or have statistical failures of subcomponents. Software failures are uniquely sourced in the design and development of the system. According to the US Food and Drug Administration, in 1996, 10% of all medical device recalls were caused by software-related issues (Maisel et al. [2001]). This percentage rose to an average of 15% of recalls from 2008 to 2012 (Fig. 1.3). Malfunctions of closed-loop medical devices usually have severe consequences, which will be categorized as *Class I*, meaning there is a "reasonable probability that use of these products will cause serious adverse health consequences or death." (Food and Administration [2006], Zhang et al. [2015], Sandler et al. [2010]).

	Software change control	Software Design	Software design manufacturing process	Sum	% of all CDRH recalls
2008	13	141	2	**156**	**18.3%**
2009	9	111	1	**121**	**15.4%**
2010	4	73	3	**80**	**8.9%**
2011	11	182	10	**203**	**15.8%**
2012	12	169	5	**186**	**15.5%**
Sum	**49**	**676**	**21**	**746**	**15.1%**

Figure 1.3: Medical device recalls due to software issues have risen from 10% in the 1990s to ˜15% in the past decade (Food and Administration [2012])

1.2 Medical Device Regulation Efforts and Challenges

The medical device industry is regulated to ensure the safety of the patients and the public. In the United States, the FDA is the primary regulatory authority responsible for assuring the safety, efficacy and security of patients using medical devices. Based on the rationale that 1) manufacturers know their devices better than the regulator, and 2) the variety of medical devices requires a variety of approaches, it is the device manufacturers' responsibility to demonstrate the safety and efficacy of the medical devices. Manufacturers are required to complete a pre-market submission before the devices can be released to the market. The level of requirements for the submission is determined by the safety classification of the devices. A set of general guidelines are recommended by the FDA (Food and Administration [1997, 2002, 2005]) which list the activities that need to be performed to ensure device safety.

In safety-critical industries such as automotive electronics, avionics and nuclear systems, international standards are enforced for software system development, evaluation, manufacturing and post-market changes (Fürst et al. [2009], Feiler et al. [2010]). This awareness is only beginning to enter the medical device industry as compliance with international standards are "recommended" in the aforementioned guidelines (Jetley et al. [2006]) but the burden of their interpretation and enforcement is on the device manufacturer. The basic rationale behind these standards is that: if all the risks/hazards of the device are identified and reasonably mitigated, and the device is developed with rigorous process, the device is *reasonably safe*.

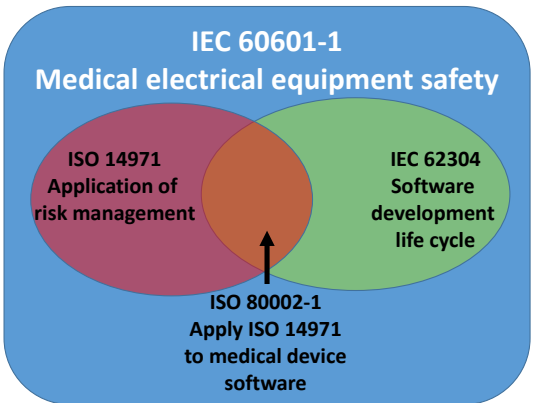


Figure 1.4: International standards for medical device safety. These standards define the required activities during the development process.

Fig. 1.4 describes the primary standards to ensure medical device safety and their relationships. The IEC 60601 Medical Electrical Equipment - General requirements for basic safety and essential performance is a product safety standard that all electronic medical devices must comply to. IEC 60324 specifies the processes and activities needed to perform during the software development life cycle to ensure software safety.

Risk management is a core activity throughout the software development life cycle. ISO 14971 is specified for the application of risk management to medical devices. In addition, for each risk management activity of ISO 14971, ISO 80002-1 provides additional guidelines for the software component, which highlights and explains approaches to assuring that software safety is adequately addressed.

1.2.1 Risk Management Challenges of Closed-loop Systems

While it is not normally possible to develop a device that is safe with a probability of 100% under all physiological and operating conditions, approaching the problem along the lines of risk analysis, risk evaluation and risk control helps better address a “designed-for-safety" mindset. Fault Tree Analysis (FTA) is a common tool in risk analysis in which hazards of the system are first identified and the possible causes of the hazards are analyzed until the initial faults are reached. Fig. 1.5.(a) demonstrate an example fault tree for

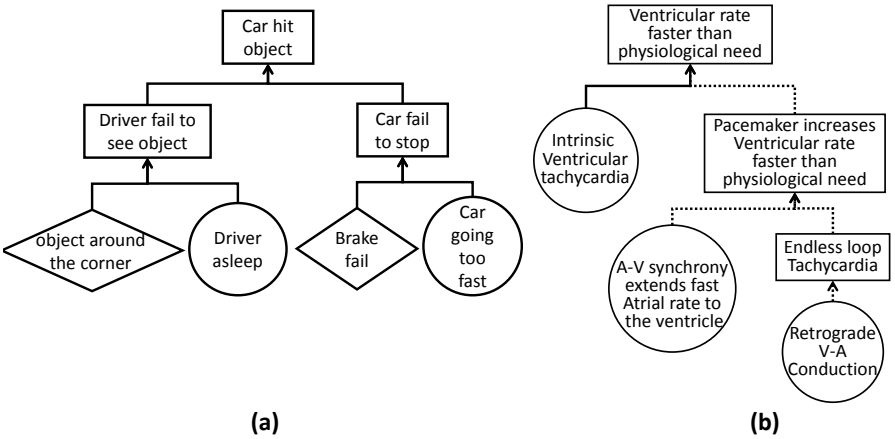


Figure 1.5: Fault Tree Analysis (FTA) Examples. (a) FTA for a hazard for a car; (b) FTA for a hazard in implantable pacemaker. The dashed line shows two mechanisms that were not identified during hazard analysis but discovered in post-market studies.

automobile. FTA is very good at showing how resistant a system is to single or multiple initiating faults. It is not good at finding all possible initiating faults since causes are conjectured and analyzed manually.

In closed-loop medical devices, there may exist interactions between the device and the patient that can cause certain hazard, but are unknown due to limits in physiological knowledge, behavior not captured in patient trials and the separation of the software development teams and the medical domain experts. Fig. 1.5.(b) describes an example fault tree for a hazard for an implantable pacemaker. There are several causes for undesirable fast ventricular rate. The well-understood cause is the intrinsic ventricular tachycardia (solid line). However, with pacemaker implanted, new mechanisms to cause hazard are introduced into the closed-loop system, as illustrated by the two branches with dotted lines. These two branches were not identified during the initial fault tree analysis, and were only identified after the devices have been released into the market, causing unnecessary adverse effects to the patients Furman and Fisher [1982]. Risks identified at this late stage are also more costly to fix, increasing the cost for device development.

After the fault tree has been constructed, probabilities for the initial faults are analyzed bottom up to calculate the probability of each hazard. The tech-

Probability of occurrence	Severity I Catastrophic (death, serious injury)	Severity II Significant (Reversible serious injury)	Severity III Marginal (inconvenience)	Severity IV Negeligible
Frequent	1	3	7	13
Probable	2	5	9	16
Occasional	4	6	11	18
Remote	8	10	14	19
Improbable	12	15	17	20

Hazard Risk Index	Acceptance Criteria
1 to 5	Unacceptable
6 to 9	Undesirable: Written and reviewed decision required
10 to 16	Acceptable upon completion of quality assurance review
17 to 20	Acceptable without review

Figure 1.6: Top table: Risk index according to occurrence and severity. Bottom table: Risk control using risk index

nique is called Failure Mode and Effects Analysis (FMEA). Then the risks are evaluated by assigning risk index to each hazard according to their occurrence and severity (Fig. 1.6). After the risks are evaluated, different activities are required to mitigate the risks according to the risk index. The risks are then be re-evaluated to calculate the residual risk and analyze the risk/benefit. This is part of the risk control process. FMEA is good at exhaustively cataloging initiating faults, and identifying their local effects. It is not good at examining multiple failures or their effects at a system level.

1.2.2 Pre-Market Evaluation with Clinical Trials

Regardless of how rigorous the risk management and the device development process are, the devices have to be able to achieve their design goal on the real patient, which can only be evaluated within its physiological environment. Devices that have high risk factors, including the closed-loop medical devices, are required to submit clinical evidence for their safety and efficacy, often in form of clinical trials. In clinical trials, the devices are used on a preselected population of patients following carefully-designed protocols. The goal of a medical trial, in part, is to obtain unambiguous results for the pri-

mary question of the trial which can support the safety and/or efficacy of the devices. However, conducting clinical trials is very time consuming and expensive, and risks found during clinical trials are very expensive to fix (U. S. Food and Drug Administration [2013]).

To address this **safety gap** between ensuring the device satisfies its therapeutic requirements with the patient-in-the-loop and testing its software specifications, new approaches for closed-loop validation of the device software within the physiological context are needed - this is the primary focus of this article.

1.3 Model-based design to improve medical device safety

With the deluge of software-based closed-loop medical devices in the coming years, relying on clinical trials as the only closed-loop evaluation method to identify risks rooted in device software is not scalable. Model-based design and virtual integration have been proposed and applied in other industries like automotive and avionics (Fürst et al. [2009], Feiler et al. [2010]), and can potentially help during the development process and provide extra confidence to the device before conducting clinical trials. However, unlike man-made systems like automobiles and aircrafts, physiological systems are less understood with larger variations for the type and degree of patient conditions. The lack of faithful models of physiological environment of the closed-loop medical devices is one of the reason that model-based design is not well-adopted in the medical device industry.

As computational models of human physiology are developed, they can be used to interact with closed-loop medical devices or their models. The FDA is starting to recognize in-silico modeling and simulation as regulatory-grade evidence for device safety and efficacy. For example, Ghorbani and Bogdan [2013] developed glucose-insulin models that can be used to evaluate control algorithms for artificial pancreas devices which can sense blood glucose and deliver insulin. Simulation results with the models have been recognized by FDA to replace animal trials, in part, which significantly reduced cost (B. P. Kovatchev and M. Breton and C. Dalla Man and C. Cobelli [2009]). With the increasing interest and recognition from the regulators, computer models and simulations are expected to play bigger role as as

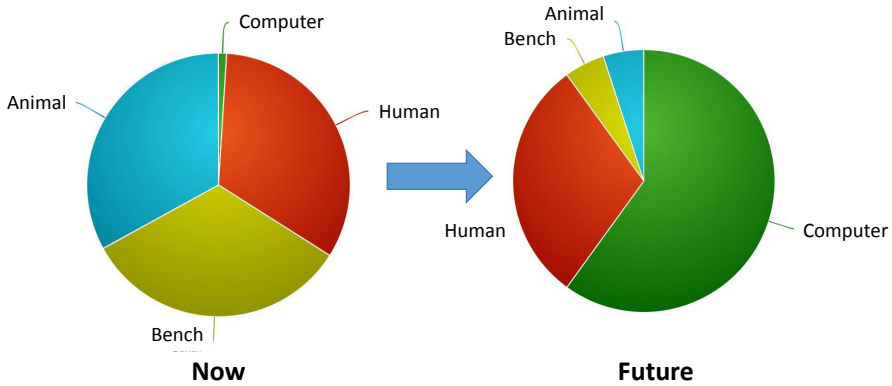


Figure 1.7: Percentage of computer simulation is expected to increase as safety and effectiveness evidence of medical devices

"regulatory-grade evidence" evidence in the development of future closed-loop medical devices (Fig. 1.7).

1.4 Contributions

In this article, we use an implantable cardiac pacemaker as a working example to demonstrate how model-based design can help improve the safety and efficacy medical device software. We demonstrate the application of model-based design in several design activities during the development process, from the perspective of the manufacturer's design validation team. We assume availability of design artifacts including pacemaker design and physiological requirements. By demonstrating the process of developing verified models to generate verified code, the results of our model-based closed-loop evaluation should be able to support the device's safety and efficacy requirements during the regulation process.

Our proposed model-driven design for closed-loop medical devices (Fig. 1.8) begins with developing heart models that can interact with real and modeled pacemakers (Jiang et al. [2012a]). In Chapter 2, we introduce our heart models for closed-loop *model checking* and *testing* of implantable cardiac devices, and the rationale for the difference between heart models used in these two applications. For closed-loop evaluation, the heart models have to be able to represent and respond under different physiological conditions.

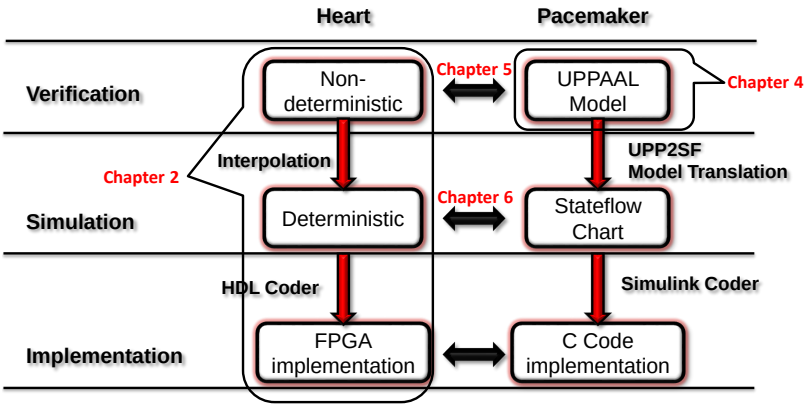


Figure 1.8: Model-driven design for verified models to verified code for the closed-loop heart and pacemaker system

The heart models are available in different formalisms to interact with the pacemaker design in closed-loop across different design stages. In Chapter 3, we validate the heart models and discuss how to identify model parameters from patient data so that the heart model can represent different physiological conditions.

In Chapter 4, we introduce the pacemaker software specification which is referenced from a dual chamber pacemaker design from Boston Scientific (Boston Scientific Corporation [2007b]). The software specification is converted to an abstract formalism called *Timed Automata* (Alur and Dill [1994]). The timed automata model of the pacemaker will be the starting point for our model-based analysis and implementation. In Chapter 5, we identify two basic hazards for pacemaker and use the UPPAAL model checker (Larsen et al. [1997]) to evaluate whether the hazards have been reasonably mitigated. With the help of heart models introduced in Chapter 2, we are able to cover the closed-loop behaviors of large variety of heart conditions so that we can evaluate whether there exists any known and even unknown mechanism to induce hazards (Jiang et al. [2014]). Pacemaker and heart models used in model-checking are abstract as model checkers do not scale well with increased model complexity. So complex dynamics of the heart and pacemaker are not captured at this stage.

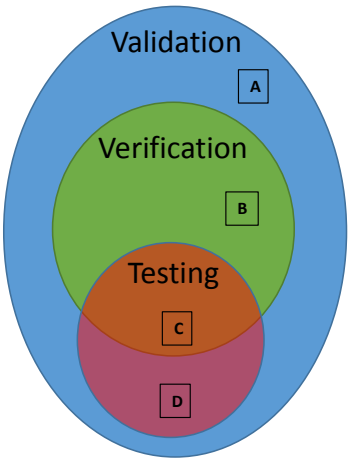
In Chapter 6, we describe the development of an automatic model translation procedure to translate models from UPPAAL to Stateflow (Inc. [2016]) to ensures that abstract models used for verification over-approximate the more detailed models used downstream (Pajic et al. [2012]). The Stateflow model of the pacemaker is then evaluated with heart models with relatively complex dynamics (Jiang et al. [2010], Jiang and Mangharam [2011], Jiang et al. [2011]). Once the detailed models pass simulation-based testing with closed-loop dynamics, they are automatically generated into code and are subject to platform-level integration testing (Jiang and Mangharam [2016]). This model-driven design approach ensures the closed-loop safety properties are retained through the design toolchain and facilitates the development of verified software from verified models.

1.5 Useful terminologies for often misinterpreted terms

Ensuring the safety of complex medical devices has drawn interest not only from stakeholders like regulators and industries, but also medical professionals and academia. Different communities have different interpretations over certain terminologies, often causing misunderstandings. In this paper we adopt the terminologies from the regulation perspective, so that the results we have fit into the regulation framework. Most of the definitions are referred from the FDA guideline document General Principles of Software Validation (Food and Administration [2002]). Below are several terminologies that we use throughout the paper which worth clarifying.

1.5.1 Requirements vs. Specifications

By the definition of FDA (Food and Administration [2005]), the requirements of a system describe **what** the system should achieve and the specifications of a system describe **how** the system is designed to satisfy the requirements. For example, a requirement for an autonomous car is "The car should not hit objects". The corresponding specification can be "brake if the speed of the car is greater than x and the distance to the object is less than y". We can see that a car satisfying its specification may not satisfy the requirement (e.g. when the car is driving too fast or the obstacle pops up right in front of the car). In this paper, we use the word requirement in particular to denote the intended uses of the medical devices to improve physiological conditions.



Validation activities	Zone
Planning	A
Requirements	A
Traceability	A,B
Change management	A
User site testing	D
Defect resolution	A,B
Risk management	A
Intended use	A
Evaluations	B
Design reviews	B
System testing	C
Regression testing	C,D

Figure 1.9: Validation activities during the software development life cycle (D A. Vogel [2011])

1.5.2 Validation vs. Verification vs. Testing

As defined in Food and Administration [2002], software validation is the confirmation by examination and provision of objective evidence that:

1. software specifications conform to user needs and intended uses, and
2. the particular requirements implemented through software can be consistently fulfilled

The first aspect ensures the device is safe and effective. The second aspect maintains the traceability of requirements throughout the development life cycle. Software verification fulfills the second aspect of software validation by "providing objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase. "

Testing is the technique that can be used for validation and/or verification. Fig. 1.9 illustrates the relationship between validation, verification and testing, and different activities during the software development life cycle to ensure the safety and effectiveness of the software.

1.5.3 Closed-loop vs. Open-loop Evaluation

In open-loop evaluation, i.e. open-loop testing, input sequences are send to the system and system outputs are compared with expected outputs. In open-loop testing, the system outputs do not affect the inputs afterward. In closed-loop evaluation, the environment of the system is taken into account. System outputs affect the state of the environment and thus affect the input sequences. For closed-loop medical devices, clinical trials are currently the most common closed-loop evaluation method. Enable closed-loop evaluation at model level requires models of the environment, which is human physiology for closed-loop medical devices.

Closed-loop evaluation accomplishes two goals in model-based design: 1) It enforces environmental constraints so that the test space is smaller and the test cases have physiological relevance. 2) Execution traces can be better interpreted as the physiological models encode domain knowledge.

2

Modeling the Physiological Environment

Closed-loop medical devices such as the implantable cardiac pacemaker and defibrillator are designed to operate autonomously and interact with the human body to maintain and improve the physiological condition of the patient. To evaluate the device within the closed-loop context of the human body, the knowledge of the physiological contexts (e.g. patient arrhythmia, physical activity) and the signals by which the device interacts with the organ(s) to manage the condition is essential. By constructing physiological models and encoding physiological requirements, our goal is to evaluate the safety and efficacy of the device therapy across a range of physiological conditions.

Consequently, it is important to model the physiological environment of the device such that details unrelated to the interaction between the device and the human are abstracted away, while essential information required to differentiate different patient conditions are maintained. As we will see, to validate the device operation across a range of physiological conditions and for a set of safety and efficacy properties, a family of models are needed which refine the closed-loop context to appropriately express the condition to be verified. In this chapter, we would therefore like to answer the following questions:

- How does the device interact with the physiological environment?
- What details must the physiological environment models capture?
- What are the different modeling philosophies when developing environment models for testing and for model checking?

Models, especially environment models of the human physiology, which span a large spectrum of scale and complexity, should be designed in accordance with their respective applications. Each application of the environment model has a different focus and has distinct modeling requirements which influence the model complexity and model identifiability.

Model complexity is generally measured in terms of the size of the state space and the computation complexity of state transitions, which affect the computation cost (memory and time) for closed-loop validation. Model complexity is largely influenced by the type of interactions with the device (e.g. analog signals, sensing, actuation) and the environment conditions specified by the physiological requirements.

Model Identifiability is a metric for the feasibility of identifying model parameters from data. There are two methods for model construction: non-parametric modeling in which no prior knowledge is assumed and the model construction is purely data-driven; and parametric modeling in which domain knowledge of the physiological conditions is taken into account. For example, to model the electrophysiological activity of the heart, there is abundant literature describing the phenomena of individual arrhythmia, which makes parametric modeling of the environment favorable.

In the following sections, we introduce the physiological contexts within which the implantable pacemaker operates, and proceed to construct heart models for closed-loop validation of the pacemaker. Note that for two different applications, i.e. model checking of the device model in the loop and functional testing of the actual device in the loop, models are constructed differently as we address their respective requirements for environment models.

2.1 Physiology Basis of the Heart and the Pacemaker

Before we get into the details of heart modeling, we use this small section to introduce the physiology basis of the heart. Readers with knowledge of this subject can skip to the modeling sections.

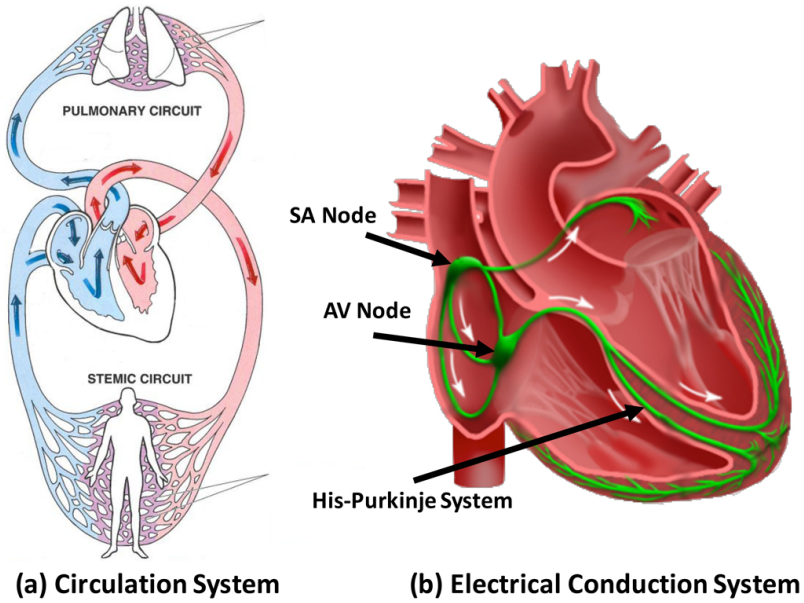


Figure 2.1: (a) The circulation system. (b) Electrical Conduction system of the heart

2.1.1 Blood Circulation System

The heart is the "motor" for blood circulation within our body. The heart has two ventricles which pump the blood out of the heart, and two atria which gather blood from the body and pump them into the ventricles. (Fig. 2.1.(a)) There are two circulations through the heart: the *Pulmonary circulation* and the *Stemic circulation*. In the pulmonary circulation, the right atrium collects oxygen-depleted blood from all over the body and pumps it into the right ventricle. The right ventricle then pumps low-oxygen blood to the lungs. The blood gets oxygenated in the lungs and gathers into the left ventricle. In the stemic circulation, the oxygenated blood in the left atrium is pumped into the left ventricle. The left ventricle pumps the blood to the rest of the body and the heart itself. After the body extracts the oxygen from the blood and injects carbon dioxide, the oxygen-depleted blood then flows back to the right atrium.

2.1.2 Electrical Conduction System of the Heart

The oxygen demand of the body changes during different activities. For example, the demand is higher while running and lower while sleeping. To satisfy these demands, the heart muscles in the atria and the ventricles have to contract with certain frequency and in accordance to optimize the *Cardiac Output*, which refers to the volume of blood pumped by the heart per minute (mL blood/min). The coordinated contractions of the heart muscles are governed by the electrical conduction system of the heart (Fig. 2.1.(b)) A *Normal Sinus Rhythm (NSR)* is the healthy heart rhythm which provides efficient blood flow. During a NSR, electrical signals are periodically generated by the *Sinoatrial (SA) node* in the upper right atrium, which acts as the intrinsic pacemaker of the heart. The signals conduct throughout both atria and trigger muscle contractions to push blood into the ventricles. After a long conduction delay at the *AV node* so that both ventricles are fully filled, the signals conduct through fast-conducting *His-Purkinje* system to trigger almost simultaneous contractions of the ventricles and pump blood out of the ventricles.

Derangement from NSR can result in insufficient cardiac output and thus insufficient oxygen supply to the body and/or the heart itself, which are referred to as *Arrythmia*. Arrhythmia impair the heart's ability to efficiently pump blood and compromise the patient's health. Arrhythmia are categorized into so-called *Tachycardia* and *Bradycardia*. Tachycardia features undesirable fast heart rate which can cause inefficient blood pumping. Bradycardia features slow heart rate which results in insufficient blood supply. Bradycardia are due to failure of impulse generation with anomalies in the SA node, or failure of impulse propagation where the conduction from atria to the ventricles is delayed or blocked.

2.1.3 Electrophysiology and Implantable Cardiac Devices

The electrical activities of the heart closely couple with the mechanical contractions thus the electrical activities of the heart can be monitored and used to diagnose arrhythmia. The most well-known method is Electrocardiogram (ECG), which measures the integration of electrical activities of the heart measured along different axis on the body surface. The electrical activities can also be directly measured by inserting electrodes through the vein into

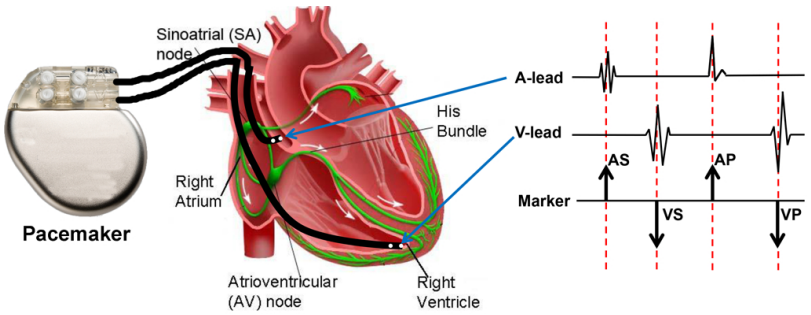


Figure 2.2: (a) Lead placement for a dual chamber pacemaker. (b) Electrogram (EGM) signals measured from pacemaker leads and corresponding internal pacemaker events

the heart. The electrodes are placed against the inside heart wall and localized electrical activities can be measured. Physicians can also deliver pacing sequence through the electrodes to explore the heart conditions. This procedure is referred to as Electrophysiological (EP) Testing (Josephson [2008]) and the signals are referred to as electrograms (EGMs) (Fig. 2.2.b). The timing and morphology of the ECG and EGM signals together are used to diagnose arrhythmia.

The implantable cardiac pacemakers are rhythm management devices designed to treat bradycardia. A typical dual chamber pacemaker has two leads inserted into the heart through the veins which can measure the local electrical activity of the right atrium and right ventricle respectively (Fig. 2.2.a). According to the timing between sensed impulses, the pacemaker may deliver electrical pacing to the corresponding chamber to maintain proper heart rhythm.

2.2 Physiological Models of the Heart

To study the mechanisms of heart diseases and their effects on cardiac output, different physiological models of the heart have been developed. Fig. 2.3 illustrates several aspects that these models capture. With the development of the imaging techniques like MRI, detailed anatomical structures of the heart can be modeled and studied (Schulte et al. [2001]). These models are fundamental in other modeling aspects as well, as the anatomy of the heart dic-

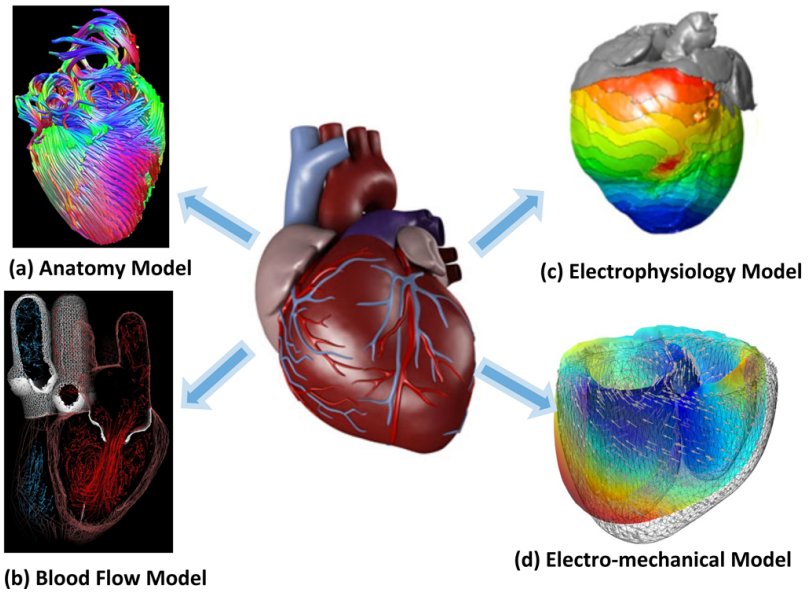


Figure 2.3: Physiological models of the heart from different perspectives

tates the electrical and mechanical behaviors of the heart. Fig. 2.3.(a) shows models for heart muscle fiber orientations by E.W. Hsu and C.S. Henriquez [2011]. With anatomy models the electrical and/ or mechanical properties of the heart can be studied. Fig. 2.3.(b) illustrate a model of blood flow within the ventricles (Peskin and McQueen [1989]). Electrical properties of the heart at cellular level has been modeled (Sachse et al. [2008]) and by combining these cellular models with the structural models, the electrical activities of the whole heart are studied, especially the mechanism of different arrhythmia (Trayanova and Boyle [2014], Grosu et al. [2011], Murthy et al. [2013]). Intrinsic heart rate variability has been modeled to synthesize optimal control of pacemaker pacing. (Bogdan et al. [2013]) Abstraction of the electrical cellular model has also been attempted by Islam et al. [2014] to reduce model complexity without sacrificing accuracy. The electrical properties and the mechanical properties of the heart are closely coupled. Models combining both of these aspects are also developed to study the effects of different arrhythmia on cardiac outputs (Trayanova and Boyle [2014], Rossi et al. [2011]).

2.3 Heart Models for Closed-loop Validation of Implantable Cardiac Devices

Models should be developed according to their applications. The aforementioned models of the heart are mostly used for understanding the mechanisms of different heart diseases. Physiological models developed for closed-loop evaluation of medical devices should have the following considerations:

C1. Interfacing with the device: The model should be able to generate physiological signals that the device sense from the real physiological entities. And the model should be able to take device output as input and change its states accordingly. Model complexity should also be adjusted according to the device interface to hide unnecessary details.

C2. Differentiate different physiological conditions: To evaluate the safety and effectiveness of the device, the device has to be evaluated under certain physiological conditions specified by the requirements. For example, the pacemaker is supposed to maintain proper heart rate during Bradycardia. The model should be expressive enough to be able to differentiate the physiological condition (Bradycardia in the example) from other conditions. Failing to do so may result in false-positives or false-negatives in the evaluation result.

C3. Physiological/logical interpretation of model states: In closed-loop evaluation we are checking the device safety and effectiveness against the physiological requirements. However, due to the limited interface (e..g two leads for a dual chamber pacemaker) it is always difficult to determine only from an execution trace that the therapy is safe and effective. Therefore, being able to provide physiological meanings to the states of the model also allows us to interpret the closed-loop execution more accurately, thus reducing the number of physiologically impossible executions during the evaluation. To satisfy these requirements, the model structure of these physiological models should base on physiological or clinical first principles so that states and state transitions of the closed-loop executions can be explained with physiological language.

C4. Available patient data: In closed-loop evaluation, physiological models are developed to represent certain physiological condition across a population of patients or even a particular patient. The model parameters must be identified so that the behaviors of the models match the behaviors of the patients (groups). Due to the limited sensing capability of closed-loop medical de-

vices, the obtained data is sparse. i.e. we can not put a sensor on every tissue region of the heart. Therefore the complexity of the model should be in accordance with the available data to avoid *over-fitting*, which occurs when a model has too many parameters relative to the number of observations, and this can introduce errors during prediction.

The electrophysiological models mentioned in the last section (Trayanova and Boyle [2014], Grosu et al. [2011]) satisfy C1-C3. However, the parameter space of these models are too large (10+ parameters for each cellular model multiplied by 10^5 of elements) which not only increase simulation complexity, but also impossible to identify due to lack of data. As introduced in Section 2.1.3, the pacemaker has only two leads at fixed locations and only use timing between local activation events for diagnosis. These models with high spatial fidelity possess details that can be abstracted without sacrificing the three considerations.

Electrophysiology testing (EP testing) has been an active clinical field to diagnose and treat arrhythmia with minimal-invasive procedures. During an EP testing procedure, the physicians diagnose heart conditions by examining the patterns and intervals of local electrical activations (temporal) measured from electrodes placed into different locations of the heart (spatial). EP testing is the perfect modeling level for closed-loop evaluation of implantable cardiac devices because: 1) it is the basis of implantable cardiac devices (C1), 2) physicians can use EP testing to diagnose most arrhythmia thus distinguish them (C2,C3), 3) there are abundant patient data available (C4).

In the remaining chapter we will introduce our heart modeling efforts based on EP testing, and model adaptation for two different applications of closed-loop evaluation of implantable cardiac devices.

2.4 Heart Models for Closed-loop Testing

During closed-loop testing, the devices interact with the environment (or its models) under different environmental conditions. The closed-loop executions are monitored and violations of safety and efficacy requirements are reported. In model-based closed-loop testing, the environment models are expected to mimic the behaviors of actual environment and its interaction with the device. Thus, the environment models are in general deterministic so that

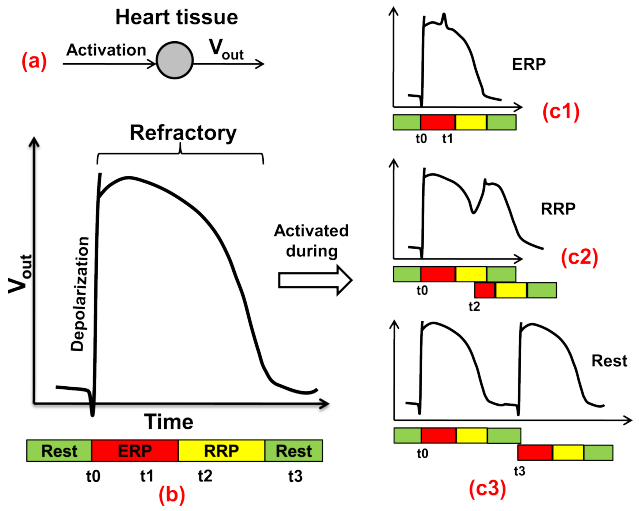


Figure 2.4: (a) The generation of Action potential; (b) Action potential; (c1) The second activation arrived during ERP; (c2) Arrived during RRP; (c3) Arrived after refractory.

the execution traces are reproducible and are able to mimic different arrhythmia. Complex dynamics during state transitions also need to be captured to validate violations within longer executions traces.

2.4.1 Modeling Philosophy

As the pacemaker can only sense and actuate from two locations within the heart, only structures and parameters that affect inputs to the pacemaker are needed. Since the two leads are fixed, the accurate spatial locations of different heart anatomical structures are not necessary. Instead, the topology of the electrical conduction system of the heart is more important.

2.4.2 Timing Behaviors of Cellular Electrophysiology

The contraction of heart muscles is triggered by external voltage applied to the tissue. After the activation, a transmembrane voltage change over time can be sensed due to ion channel activities, which is referred to as an Action Potential (Fig. 2.4(a)). The upstroke of the action potential is called depolarization, during which the muscle will contract. The voltage change caused by the depolarization will depolarize the tissue nearby, which causes an ac-

tivation wave across the heart. After the depolarization there is a refractory period during which the tissue recovers to the pre-excitation state and the voltage drops down to the resting potential. The refractory period can be divided into *Effective Refractory Period (ERP)* and *Relative Refractory Period (RRP)* (Fig. 2.4(b)). During ERP, the tissue cannot be depolarized due to the lack of charge. As a result, the activation wave will be "blocked" at the tissue during ERP (Fig. 2.4(c1)). During RRP, the tissue is partially recovered and the tissue can be depolarized. However, the new action potential generated by the depolarization will have different morphology (e.g. attenuated in magnitude and duration), thus affecting the refractory periods of the tissue and conduction delay of the activation wave (Fig. 2.4(c2)). Fig. 2.4(c1)-(c3) show the action potential shape change and corresponding timing change in refractory periods when the tissue is activated at time stamp $t1$, $t2$, $t3$ after the initial activation $t0$.

2.4.3 Heart Model Components

We introduce the model components that can be used to configure heart models corresponding to different heart conditions. As discussed earlier, the action potential of a heart tissue has 3 timing periods during which the tissue responds to external electrical stimuli differently. We use an extended timed-automata formulation (Alur and Dill [1994]) to model the timing behaviors of a heart tissue during each cycle.

Node Automata: We refer to the tissue model as *node automaton* and Fig. 2.5.(a) shows the structure of a node automaton i. 3 states correspond to the timing periods of the action potential. From Rest state, the node can either self-activate or get activated by external stimuli (Act_node) and go to ERP state. During ERP state the node does not respond to external stimuli (blocked). During RRP state, the node can still be activated and go to ERP state, however the ERP period and the conduction delay of the tissue are affected by the "earliness" of the activation arrived during the RRP period, which is tracked by a shared variable $C(i)$. The new ERP period is determined by a function over clock value $g(f(t))$ which mimics the beat-to-beat dynamics described in Josephson [2008]. The function g and f are given by:

$$f(t) = 1 - t/Trrp \tag{2.1}$$

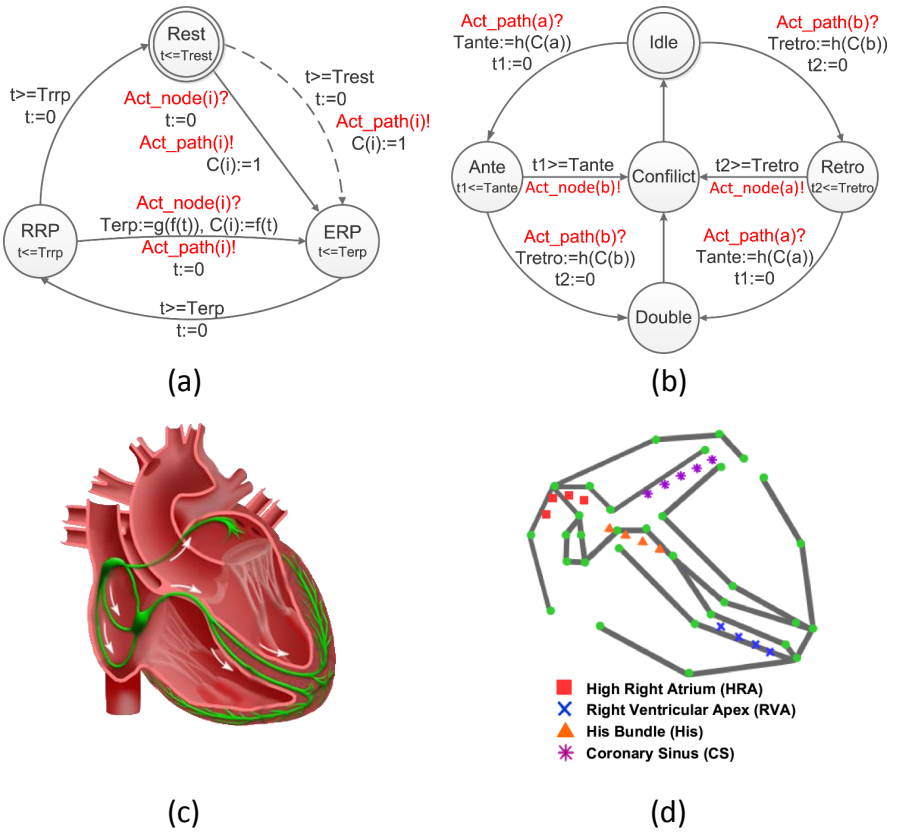


Figure 2.5: (a) Node automaton: The dotted transition is only valid for tissue (like SA node) that can be activated by an external trigger; (b) Path automaton modeling the electric conduction and propagation between two node automata; (c) Electrical conduction system of the heart; (d) Model of the electrical conduction system of the heart using a network of node & path automata Jiang et al. [2012a].

and

$$g(x) = \begin{cases} T_{min} + (1 - (1-x)^3) \cdot (T_{max} - T_{min}), i = AV \\ T_{min} + (1 - x^3) \cdot (T_{max} - T_{min}), i \neq AV \end{cases} \tag{2.2}$$

where T_{min} and T_{max} are the minimum and maximum value for *Terp* of the tissue.

Due to the limited number of observable points within the heart, modeling the electrophysiological behavior of every tissue of the heart and its

full anatomy is unnecessary and unfeasible. In our heart models, only self-activating tissue and key hubs of the electrical conduction system are modeled as node automata.

Path Automata: The electrical conduction through the tissue between nodes are abstracted using *path automata*. The path automata can be used to represent structural or topological (functional) electrical connections between nodes. Fig. 2.5.(b) shows a path automaton connecting node a and b.

The initial state of a path automaton is *Idle*, which corresponds to no conduction. The states corresponding to the two conduction directions are named after the physiological terms: Antegrade (Ante) and Retrograde (Retro). These states can be intuitively described as forward and backward conductions. If path actuation *Act_path* event is received from one of the nodes connected to it, there is a transition to *Ante* or *Retro* state based on the activation source in the path automaton. At the same time, the clock invariant of the state is modified according to the shared variable *C(a/b)*. This corresponds to the change of the conduction delay that is caused by the early activation. Similar to node automaton, the changing trend is extracted from clinical data and the function h is defined as:

$$h(c) = \begin{cases} path\_len/v \cdot (1 + 3c), i = AV \\ path\_len/v \cdot (1 + 3c^2), i \neq AV \end{cases} \tag{2.3}$$

where $path\_len$ denotes the length of the path and v is the conduction velocity.

After *Tante* or *Tretro* time expires, the path automaton sends out *Act_node(b)* or *Act_node(a)* respectively. A transition to *Conflict* state occurs followed by the transition to *Idle* state. The intermediate state *Conflict* is designed to prevent back-flow, where the path is activated by the node *b* it has just activated. If during *Ante* or *Retro* state another *Act_path* event is received from the other node connected to the path automaton, a transition to *Double* state will occur, corresponding to the two-way conduction. In this case, the activation signals eventually cancel each other and the transition to *Idle* state is taken.

2.4.4 Modeling the Heart's Electrical Conduction System

The node and path automata are the basic building blocks for heart modeling. Hearts with different conditions are modeled by using different conduction

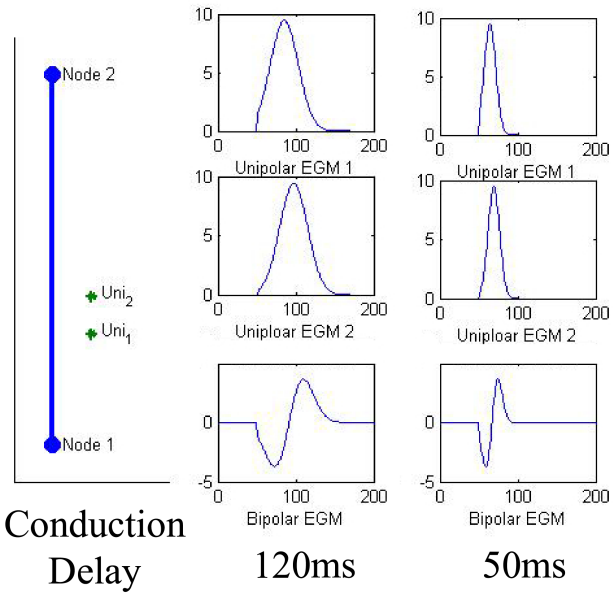


Figure 2.6: The influence of conduction velocity and probe configuration on the EGM morphology. The left columns show the placement of probes in relation to the path; the right columns show the functional EGM.

topologies with appropriate timing parameters for each node and path automata. Fig. 2.5.(d) shows one such topology of a network of node and path automata.

2.4.5 Interaction with the Heart Model

In EP testing and during pacemaker implantation, the local electrical activities, measured as electrogram (EGM) signals, are used to diagnose heart conditions. During heart model construction, we can assign a node automaton at electrode locations and the transitions to the ERP state can be used to represent the local activation events. In a more general setup where electrodes are assigned anywhere within the heart model, a probe model is designed to generate synthetic EGM signals using spatio-temporal information from the proximity to the network of node and path automata. Fig. 2.6 shows the morphology of EGM signal changes with different conduction velocity and probe configurations. A detailed description of the probe model can be found

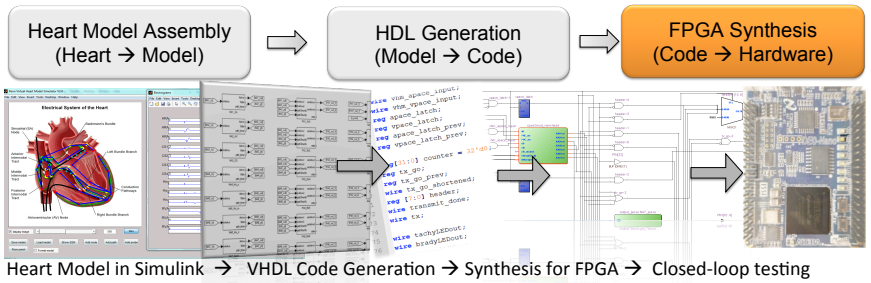


Figure 2.7: The heart model was developed in Matlab/Simulink and code was automatically generated to operate on an FPGA platform for platform-level testing.

in Jiang and Mangharam [2011].

2.4.6 Heart-on-a-Chip Platform

Platform testing remains the primary means to verify and validate device software. Currently testing is done by feeding recorded open-loop heart signals to the device and evaluating the device output. Consequently, the change in the state of the heart condition, in response to device output, is not taken into account. Thus, device malfunctions involving state changes due to multiple closed-loop interactions will not be captured during open-loop testing.

To this effect, the heart model described above is also implemented on hardware platform (Fig. 2.7) for closed-loop testing. Since each heart model is a network of node and path automata running concurrently, we implemented the heart model on an FPGA, so that increasing in the number of nodes and paths would not affect real-time constraints. The second generation heart model implementation has been implemented on a lower cost fast micro-controller platform. The fast clock ensures that executions of all nodes and paths can be finished within 1ms. The Heart-on-a-Chip platform includes a heart model implementation which is able to represent common heart conditions such as bradycardia, tachycardia, heart block, etc (for mode details refer to Jiang et al. [2012a]). The parameters of the heart model can be changed at run-time by either switching among pre-defined parameter sets, or sending values directly to the model through a user interface in Matlab. A monitoring system observes logical interactions between heart model and the pacemaker and checks them against safety invariants at run-time.

As shown in (Fig. 2.8), with an analog interface the heart model can interact with a commercial pacemaker in real time. Our analog interface uses an optical isolation circuit to separate the pacemaker circuit and the heart implementation. Signals generated from the heart are attenuated to the appropriate level to interact with a Boston Scientific pacemaker and analog pacing signals are converted to pacing events received by the heart model.

2.5 Heart Models for Closed-loop Model Checking

During closed-loop model checking, the whole closed-loop state space of the device and the environment is mathematically explored against physiological requirements. The ideal environment models should be: (1) simple enough to avoid the state space explosion problem (Clarke et al. [1994]), (2) general enough to cover possible physiological operation parameters, and (3) expressive enough to represent specific physiological conditions in-depth. It is obvious that no single model can achieve all three properties. A rigorous framework should be adapted so that models with the appropriate level of

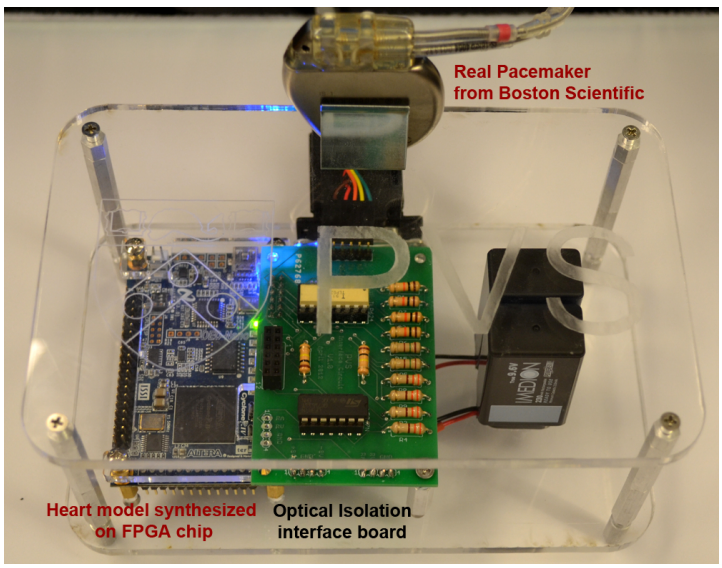


Figure 2.8: Heart-on-a-Chip testbed for real-time closed-loop testing of the pacemaker or model of the pacemaker with the heart model on the hardware platform

refinement, from a family of models, are selected based on the requirement type. For example, for a simple property that the pacemaker must maintain the heart rate above a desired level, a relatively abstract model will suffice. For a more complex property such as the termination of Pacemaker Mediated Tachycardia (Josephson [2008]), an appropriately refined model which expresses the physiological condition will be selected.

2.5.1 Modeling Philosophy

Model Formalism: Choosing an appropriate formalism for the physiological models is important as the formalism determines the closed-loop state space and hence the feasibility to do model checking. The pacemaker utilizes the timing of local electrical events to diagnose heart conditions. It is therefore natural to use timed-automata models of the heart. With this formalism, we use the UPPAAL model checking tool (Behrmann et al. [2004]) such that the whole closed-loop state space is explored symbolically.

Model Coverage: Pacemakers are designed to treat bradycardia by maintaining the appropriate heart rate when the intrinsic rate is low. However, at the same time, a pacemaker should not adversely affect other heart conditions such as supraventricular tachycardia (Zhang et al. [2015]). Even for the same patient, the heart condition changes over time and must be addressable by the modeling effort. In order to achieve safety across all possible heart conditions, the heart models used during closed-loop verification should be able to cover all possible heart behaviors, more precisely, their mapping to pacemaker inputs. Over-approximation (Clarke et al. [2003]) with non-determinism can be used to simplify model structure while covering larger variety of environmental behaviors. Techniques like model-checking can then be used to examine the whole closed-loop state space for property violations.

Ambiguity due to Limited Sensing Capability: The spatial sensing resolution of pacemakers is low, in terms of the number of sensing location (2-3 leads), as well as the information obtained from each sensing location (binary events with no analog morphology). Through the process of abstraction, if different heart conditions are able to generate exactly the same input

sequence to the pacemaker, there will be ambiguities in concretizing abstract closed-loop executions. For certain conditional requirements, it is important to differentiate all possible concrete executions corresponding to an abstract execution. As the result, the heart model(s) should have the capability to differentiate these heart conditions when verifying certain properties. Thus, a single heart model will not suffice and a family of heart models are required.

Information Loss during Abstraction: While over-approximation achieves simplicity and coverage, it also inevitably introduces invalid behaviors (e.g. not clinically feasible or relevant) into the model which can cause false-negatives and false-positives during model checking. To solve this problem, refined models of the heart should be available which can resolve spurious counterexamples and eliminate invalid executions when necessary to avoid false-positives.

The most challenging aspect during closed-loop model checking is the abstraction and refinement of the environment model. In Jiang et al. [2014] we developed a series of heart model abstractions at various abstraction levels. The models are abstracted using abstraction rules derived from physiological knowledge, thus ensuring that each abstraction step covers more physiological conditions. The models in adjacent abstraction levels also satisfy timed-simulation relationship (Yamane [2006]) to ensure complete coverage in the more abstract model. In the remainder of this section, we briefly discuss this multi-scale modeling process and the domain knowledge used. The detailed abstraction and proof for simulation relationship can be found in Jiang et al. [2014].

2.5.2 Timed Automata and Timed Simulation

Timed automata (Alur and Dill [1994]) is an extension of a finite automaton with a finite set of real-valued clocks. It has been used for modeling and verifying systems which are triggered by events and have timing constraints between events. UPPAAL is a standard tool for modeling and verification of real-time systems, based on networks of timed automata. The graphical and text-based interface makes modeling more intuitive. Requirements can be specified using Computational Tree Logic (CTL), as described in Clarke and Emerson [1982], and violations can be visualized in the simulation

environment.

The rest of this section describes the process of model abstraction and refinement where more abstract models cover more of the state space but can only prove simpler properties. As we refine models, they cover lesser state space but can prove more complex properties. The reader looking for a more intuitive understanding of the area may skip the detailed description in the rest of this chapter, during the first reading.

Syntax of Timed Automata

A timed automaton **G** is a tuple $\langle S, S_0, \Sigma, X, inv, E \rangle$, where

- S is a finite set of locations.
- $S_0 \in S$ is the set of initial locations.
- Σ is the set of events.
- X is the set of clocks.
- inv is the set of invariants for clock constraints at each location.
- E is the set of edges. Each edge is a tuple $\langle s, \sigma, \Psi, \lambda, s' \rangle$ which consists of a source location s, an event $\sigma \in \Sigma$, clock constraints Ψ, λ as a set of clocks to be reset and the target location s'.

For the clock variables X, the clock constraints $\Psi \in \Psi^X$ can be inductively defined by $\Psi := x \perp c \| \Psi_1 \wedge \Psi_2$, where $\perp \in \{\leq, =, \geq\}$, and $c \in \mathbb{N}$.

Semantics of Timed Automata

A state of a timed automaton is a pair $\langle s, v \rangle$ which contains the location $s \in S$ and the valuation v for all clocks. The set of all states is Ω. For all $\lambda \in X$, $v[\lambda := 0]$ denotes the valuation which sets all clocks $x \in \lambda$ as zero and the rest of the clocks unchanged. For all $t \in \mathbf{R}$, $v + t$ denotes the valuation which increase all the clock value by t. There are two kinds of transitions between states. The discrete transition happens when the condition of an edge has been met. So we have:

$$\langle s, \sigma, \Psi, \lambda, s' \rangle \in E, v \models \Psi, v[\lambda := 0] \models inv(s')$$

$$\Rightarrow (s, v) \xrightarrow{\sigma} (s', v[\lambda := 0])$$

The timed transition happens when the timed automaton can stay in the same location for certain amount of time. We have:

$$\delta \in \boldsymbol{R}, \forall \delta' \leq \delta, v + \delta' \models inv(s)$$

$$\Rightarrow (s, v) \xrightarrow{\delta} (s, v + \delta)$$

Timed Simulation

For two timed automata $T^1 = \langle S^1, S_0^1, \Sigma^1, X^1, inv^1, E^1 \rangle$ and $T^2 = \langle S^2, S_0^2, \Sigma^2, X^2, inv^2, E^2 \rangle$, a timed simulation relation is a binary relation sim $\subseteq \Omega^1 \times \Omega^2$ where Ω^1 and Ω^2 are sets of states of T^1 and T^2. We say T^2 time simulates T^1 ($T^1 \preceq_t T^2$) if the following conditions holds:

- Initial states correspondence: $(\langle s_0^1, \mathbf{0} \rangle, \langle s_0^2, \mathbf{0} \rangle) \in$ sim
- Timed transition: For every $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in$ sim, if $\langle s_1, v_1 \rangle \xrightarrow{\delta} \langle s_1, v_1 + \delta \rangle$, there exists $\langle s_2, v_2 + \delta \rangle$ such that $\langle s_2, v_2 \rangle \xrightarrow{\delta} \langle s_2, v_2 + \delta \rangle$ and
  $(\langle s_1, v_1 + \delta \rangle, \langle s_2, v_2 + \delta \rangle) \in$ sim.
- Discrete transition: For every $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in$ sim, if $\langle s_1, v_1 \rangle \xrightarrow{\sigma} \langle s_1', v_1' \rangle$, there exists $\langle s_2', v_2' \rangle$ such that $\langle s_2, v_2 \rangle \xrightarrow{\sigma} \langle s_2', v_2' \rangle$ and $(\langle s_1', v_1' \rangle, \langle s_2', v_2' \rangle) \in$ sim.

Certain properties are preserved for timed simulation relation. For $\varphi \in ATCTL$, if $M \preceq_t M'$, we have $M' \models \varphi \Rightarrow M \models \varphi$.Yamane [2006] However, $M' \not\models \varphi \Rightarrow M \not\models \varphi$ does not hold. Violations of $ATCTL$ yield counter-examples and the validity of which need to be checked on more refined model.

It is known that timed simulation relation is also closed under composition Yamane [2006]. So when we have two heart models $H_1 \preceq_t H_2$ we will have $H_1 \| P \preceq_t H_2 \| P$ where P is the timed-automata model of the pacemaker. For $\varphi \in ATCTL$, we have $H_2 \| P \models \varphi \Rightarrow H_1 \| P \models \varphi$. With this property we can verify the pacemaker model with abstract heart model. In the rest of the section, we will describe how we develop our initial heart model from the physiological perspective and abstract the model step by step so

that the complexity of the model is reduced for verification. Given two heart models H_1, H_2 and a timed simulation mapping sim=$\Omega_1 \times \Omega_2$, there are no automated methods to check $H_1 \preceq_t H_2$. In the Appendix, we show the manual proof for the timed simulation relation between two heart models H_2 and H_3. Other timed simulation relations can be proved similarly.

2.5.3 Abstraction 0: Initial Abstraction

For the initial heart model, we assume the heart tissue is modeled with all its spatial detail. For the temporal aspect, we model each tissue region as an automaton $N0$ shown in Fig. 2.9(b). The beat-to-beat dynamics of heart tissue, modeled by the deterministic model, is abstracted using non-determinism. For example, the ERP period and conduction delay of the tissue are non-deterministically chosen from ranges instead of deterministic functions. The whole heart can be modeled by composing tissue models with different parameters $H_0 = N_0^1 \| N_0^2 \cdots N_0^n$. However, for a real heart, the number n is very large and the connectivity and parameter values are difficult, and perhaps impossible, to determine, which makes verification with H_0 infeasible.

2.5.4 Abstraction 1: Abstract Conduction Delays With Paths

At the first abstraction step, we separate the conduction delay (modeled by the *cond* state in $N0$) from the new node automata $N1$ and model the conduction between two nodes using path automaton $P1$. Since the beat-to-beat dynamics are abstracted with non-determinism, the RRP state is merged with the Rest state in $N1$. The procedure gives us the intuition to abstract the heart as a conduction network as shown in Fig. 2.9 (c).

2.5.5 Abstraction 2: Merging Equivalent States

The heart model H_1' still has equivalent locations. In Abstraction 2 we further abstract the node and path automata by merging equivalent states. In the new node automaton N_2 we merge the RRP state with the Rest state with:

$$N_2.Trest\_min = N_1.Trest\_min + N_1.Trrp\_min$$

$$N_2.Trest\_max = N_1.Trest\_max + N_1.Trrp\_max$$

Under the assumption that the ERP period of a node automaton is much

longer than the conduction delay of a path automaton, the Double and Conflict location of the path automaton is merged with the Idle state. The heart can be modeled as $H_2 = N_2^1 \| P_2^1 \| N_2^2 \| P_2^2 \| N_2^3$.

2.5.6 Abstraction 3: Replacing Blocking with Non-deterministic Conduction

In Abstraction 3, we replace the blocking behavior of the ERP location of the node with non-deterministic conduction in the path automaton. There exists a transition

$$RE \| ID \| RE \xrightarrow[Act\_path\_1!]{Act\_node\_1?} \xrightarrow{Act\_path\_1?} RE \| ID \| RE$$

in $N_3^1 \| P_3^1 \| N_3^2$ to replace transition

$$ER \| ID \| ER \xrightarrow{Act\_node\_1?} ER \| ID \| ER$$

in $N_2^1 \| P_2^1 \| N_2^2$

Without the ERP constraint the AV node is no longer needed and the heart can be modeled as $H_3 = N_3^1 \| P_3 \| N_3^2$. The detailed proof for this timed simulation relation can be found in Jiang et al. [2014].

2.5.7 Abstraction 4: Random Heart Model (RHM)

In Abstraction 4, we further simplify the heart model by removing the conduction path between two nodes. By setting $Trest\_min$ for both nodes to 0 the new heart model $H_4 = N_3^1 \| N_3^2$ covers all possible behavior of H_3. This random heart model with two nodes is the most abstract model that will be used at the beginning of the closed-loop model checking process.

Eventually we have a series of heart models with:

$N_0^1 \| N_0^2 \cdots N_0^n$
$\preceq_t N_1^1 \| P_1^1 \| N_1^2 \cdots P_1^m \| N_1^n$
$\preceq_t N_1^1 \| P_1^1 \| N_1^2 \| P_1^2 \| N_1^3$
$\preceq_t N_2^1 \| P_2^1 \| N_2^2 \| P_2^2 \| N_2^2$
$\preceq_t N_3^1 \| P_3 \| N_3^2$
$\preceq_t N_4^1 \| N_4^2$

Using this technique, we systematically create a family of heart models with different levels of complexity and expressiveness. In Chapter 5 we de-

scribe how to balance between complexity and expressiveness and use these heart models for closed-loop model checking of implantable pacemaker.

2.5.8 Discussion

In this chapter, we use heart modeling as example to demonstrate how to develop physiological models for closed-loop evaluation of closed-loop medical devices. We emphasized that models should be developed according to their applications. We developed heart models based on clinical Electrophysiological Testing, and modeled the topological and temporal behaviors of the heart with timed-automata formulation. In the next chapter we will demonstrate the identification and validation of the heart models.

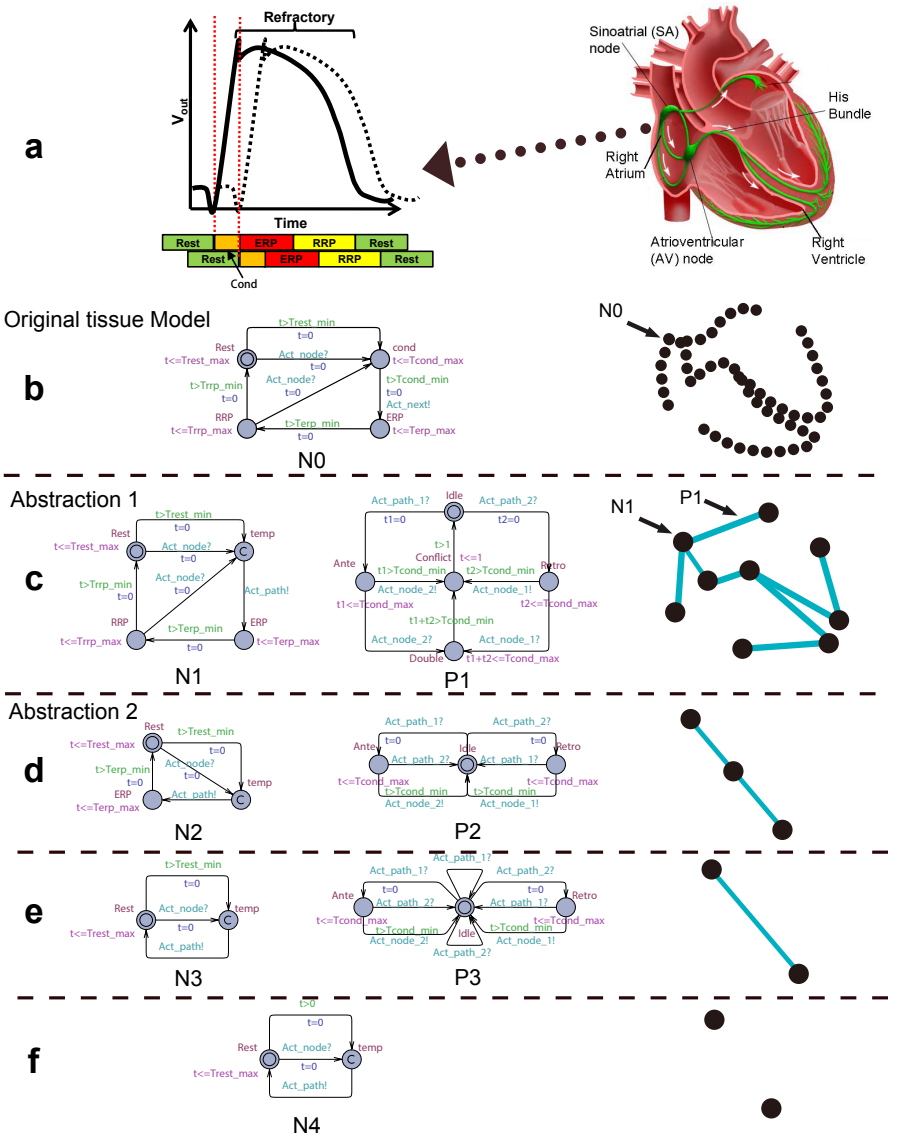


Figure 2.9: (a) Electrical voltage change measured on the heart tissue and its adjacent tissue region (dotted) upon activation. The whole process is divided into timing periods with different behaviors. (b) The original tissue model which captures the refactory timing behaviors of the heart tissue. (c) The conduction property is separated to the path automaton and the heart can be modeled as conduction network. (d) Equivalent locations are merged. (e) The blocking property of the ERP location is replaced by a non-deterministic conduction in the path automaton. (f) Conduction between nodes are replaced by self-activations of the nodes. More details in Jiang et al. [2014]

3

Identifying and Validating the Environment Model

Physiological models are developed to represent certain clinical conditions common across a population of patients, or the conditions of a specific patient. Consequently, the structure of the model and corresponding parameters have to be identified. This information can be obtained from electrogram data collected during medical procedures and from physiological literature in which population data has been analyzed and summarized. Due to limited interactions with the patient (e.g. during a device implantation procedure or an ablation procedure), currently the quality and quantity of patient-specific physiological data is sparse as there is generally not enough information to identify all the parameters in the heart model. A model with the spatio-temporal structure that is similar to the conduction patterns in the heart helps simplify the process of identifying the model parameters. A rigorous procedure for the model identification step is an important contributor to the model validation step. In this chapter, we first aim to answer the following questions:

- What is the importance of model identification for closed-loop simulation and model checking?
- How are models identified from patient data and patient population parameters?

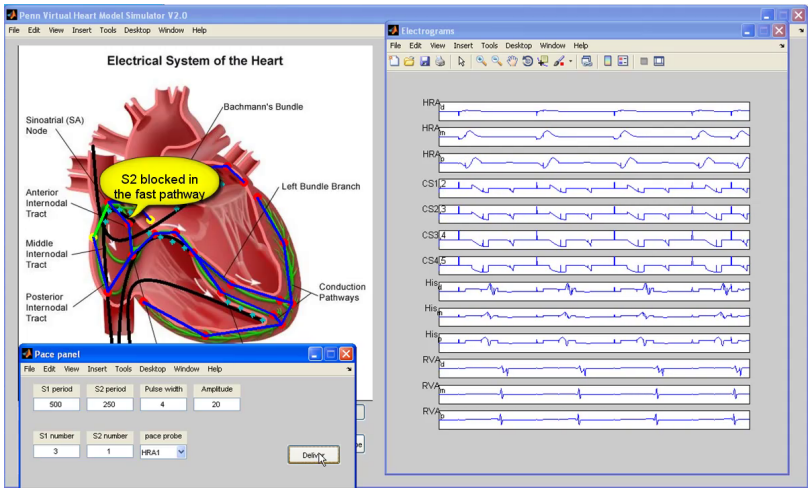


Figure 3.1: Simulation model of the heart showing the conduction pathways (left) with electrogram signals from different probe locations (right) and an interactive pacing panel (bottom left). In this case, the heart was paced four times at an interval of 500ms, followed by a pacing at a shorter (250ms) interval. This EP Testing procedure is employed to trigger conduction along alternative pathways and check for the existence of a reentry circuit.

In the following section, we briefly discuss our model identification effort for heart models used in two closed-loop verification applications, and their corresponding challenges. This is followed by the procedures to validate the heart models before they are used for closed-loop verification and testing of the pacemaker.

3.1 Heart Model Identification for Closed-loop Testing

In closed-loop simulation, a deterministic heart model should be identified to represent a specific patient under a certain heart condition. The constraints for model parameters can be obtained from patient data with *Electrophysiological (EP) Testing*. During EP testing, the physician delivers electrical pacing sequences from electrodes placed inside the patient's heart to instigate responses along fast and slow conduction pathways (Fig. 3.1). The observed patterns and timing of electrical events are used to extract conduction and propagation properties of different tissue regions across the myocardium. Since the goal for any EP testing procedure is not to determine all the timing

parameters for a patient, the number of parameters that can be identified from the patient data is limited.

Fig. 3.2 illustrates how timing parameters can be extracted during an EP testing procedure. Fig. 3.2(a) shows a setup with two electrodes placed in the right atrium and right ventricle of the heart respectively. EGM signals can be measured from these two electrodes (Fig. 3.2(b)). The physician delivers a series of long interval pacing sequences followed by one or more short interval pacing through the electrodes. This may trigger different responses along primary and alternate conduction pathways from the patient's heart. Fig. 3.2(c) shows a heart model structure with unknown parameter values. By analyzing the *timing* and *pattern* of the EGM signals we extract constraints on the heart model parameters. In EGM sequence 1, the interval between two intrinsic activations $a1$ and $a2$ in EGM A is 700ms, so we have:

$$ERP1 + RRP1 + Rest = 700ms$$

The interval between $a1$ and $v1$ is 150ms, so we have:

$$Td1 + Td2 = 150ms$$

In EGM sequence 2, the pacing interval from Electrode A is 300ms. By observing that the interval between $p1 - v1$ is less than the interval between $p2 - v2$, we know that $p2$ arrives during the RRP period of the AV node. So we have:

$$ERP1 + RRP1 \leq 300ms$$

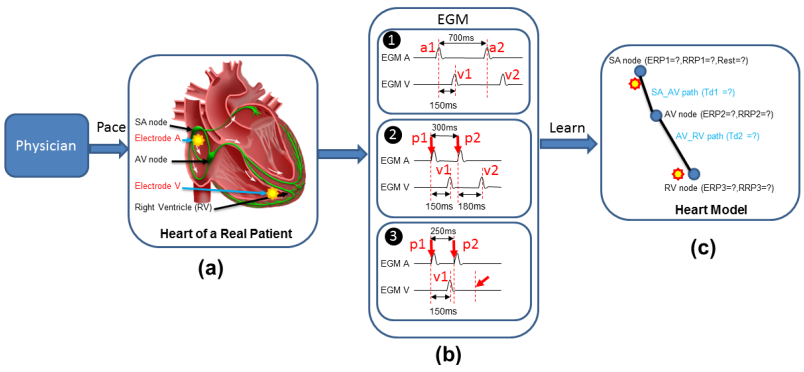


Figure 3.2: (a) The illustration of the probe locations. (b) Multiple pacing sequences with different timing outcomes. (c) The heart model with undecided parameters

TABLE 2-1 Normal Conduction Intervals in Adults

Laboratory	P-A	A-H	H-V	H
Narula (2,5)	25–60	50–120	35–45	25
Damato (1,3,18,28)	24–45	60–140	30–55	10–15
Castellanos (6)	20–50	50–120	25–55	
Schuilenburg (23,24)	85–150	35–55		
Peuch (4,14)	30–55	45–100	35–55	
Bekheit (25,26)	10–50	50–125	35–45	15–25
Rosen (27)	9–45	54–130	31–55	
Author	60–125	35–55	10–25	

Figure 3.3: Timing intervals measured during clinical studies Josephson [2008]

In EGM sequence 3, the pacing interval is further reduced to 250ms. There is no $v2$ corresponding to $p2$, indicating $p2$ arrives during the ERP period of the AV node. So we have:

$$ERP1 \leq 250ms$$

Each experiment provides additional time constraints for model parameters. By systematically conducting experiments certain model parameters can be uniquely identified within a relatively tight range. However, even with simplified model structure like the one in the example, not all model parameters can be uniquely identified due to limited number of electrodes and limited number of experiments during a real procedure.

3.1.1 Heart Model Identification in Closed-loop Model Checking

In model checking, the heart models have simpler structure and fewer parameters due to non-deterministic abstraction. The placement and connectivity of nodes and paths in the heart models are developed to be consistent with EP practice. This way, each node and path automata and their timing parameters have physiological correspondence to parameters found in literature (Fig. 3.3). The range for non-deterministic parameters directly corresponds to the range for possible values of the respective physiological parameters. Therefore, model identification for model checking is much simpler and requires less EP testing data. It is important to note here that model checking of abstract models of the closed-loop system and testing of the device in the loop are complementary approaches for validating the safety and efficacy of the overall system.

3.2 Validating the Environment Model

Since models are approximations of the actual environment, there are always discrepancies between the model and the actual patient (group). The challenge then is to evaluate the confidence in the safety guarantees that model-based closed-loop verification can provide. The metrics and process to validate the environment model is different for the two applications of heart modeling: in closed-loop model checking, the model's **coverage** on environmental behaviors is more important, while in closed-loop simulation, the **accuracy** of the model is more important.

In this chapter, we aim to answer the following questions and use our heart models as examples to demonstrate different validation procedures which improve the fidelity of the environment model.

- What are the different methods to validate physiological models?
- How much confidence is sufficient from the model validation process?

3.2.1 Validating Models for Closed-loop Simulation

A physiological model is considered valid for closed-loop simulation if (a) it is capable of generating the same output as the patient, for the same input; and (b) it is general enough to represent other patients with similar conditions by adjusting its parameters. The second point is to ensure that the model successfully captures the underlying mechanism instead of over-fitting the data. In the following example we validate the capability of our heart models to represent certain heart conditions according to the mechanisms described in physiological literature, and output the correct responses across a range of inputs.

Quantitative Heart Model Validation: During an EP testing procedure, the physician places catheters inside the patient's heart to observe local electrical activity from different locations of the heart. The His bundle catheter (HBE) is particularly important when evaluating the atria-to-ventricle conduction path (Fig. 3.4). For each A to V conduction there are 3 impulses which correspond to atrial contraction (A), His bundle activation (H) and ventricular activation (V). In this case study, two pacing signals $a1$ and $a2$ are delivered to the heart from the high right atrial catheter (HRA). By gradually decreas-

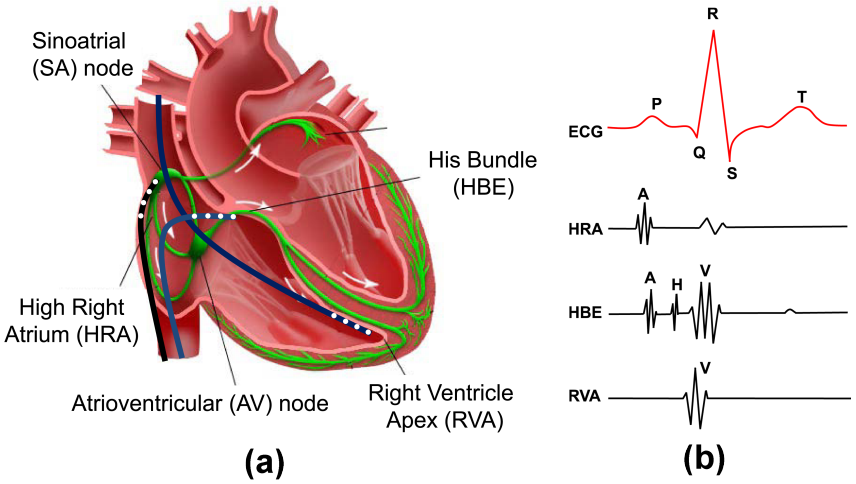
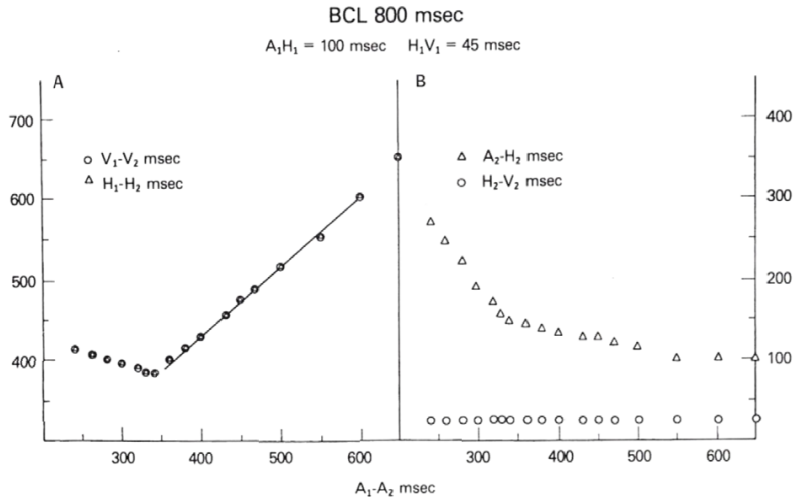
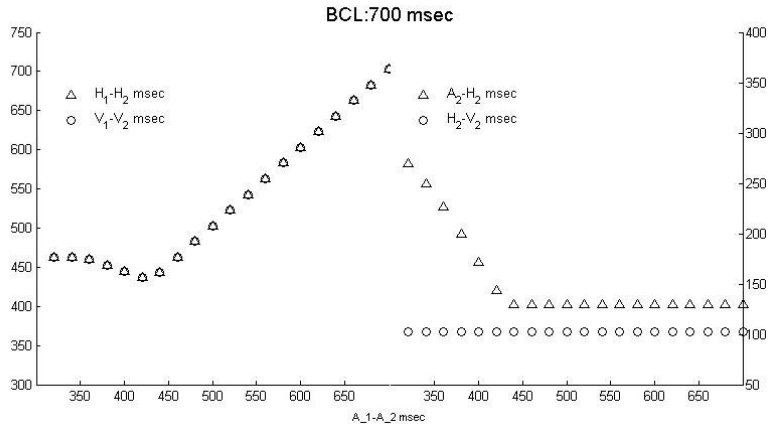


Figure 3.4: (a) Probe locations for a general EP testing procedure. (b) EGM signals measured from the probes at the high right atrial (HRA), His bundle (HBE) and right ventricular apex (RVA) standard catheter positions

ing the pacing interval in each test, certain tissue along the A-V conduction path will be activated during its refractory period, thus affecting the conduction delay further down the conduction path and change the intervals between the impulses. Fig. 3.5(a) shows the relation between pacing interval ($a1$-$a2$) and corresponding intervals between A, H and V impulses. On the left side it shows that interval $H_1 - H_2$ and $V_1 - V_2$ decrease but remain equal as the pacing interval decreases, indicating the tissue with the longest refractory period along the path is not between the His Bundle and the ventricles. When the pacing interval decreases to 350ms both intervals increases, indicating that the RRP of certain tissue has been reached and the tissue is between the atria and the His bundle. On the right it shows that the $A_2 - H_2$ interval increases as the pacing interval decreases, which further proves the hypothesis that the AV node, which is between the atria and the His bundle, has the longest refractory period along the A-V conduction path. We configured our heart model such that the AV node has the longest refractory period and performed the same study by decreasing the pacing interval. The heart model shows the same trend as that of the real patient (Fig. 3.5(b)).



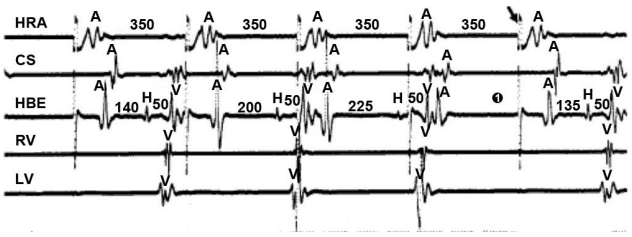
(a)



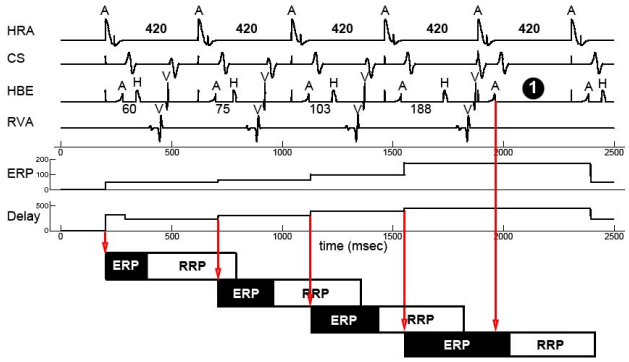
(b)

Figure 3.5: Key interval values when the coupling interval shortens for (a) a real patient (Josephson [2008]) and (b) in heart model simulation (Jiang et al. [2010]).

Validation by comparison to real patients: This heart condition can also show Wenckebach type A-V nodal response. In this case, a sequence of pacing signals with a short coupling interval ($A_1 - A_2 <= AV.Terp + AV.Trrp$)



(a) Real patient's electrograms



(b) Heart model's electrograms

Figure 3.6: (a) Electrograms of induced Wenckebach block in a patient. (b) Electrograms of induced Wenckebach block in the heart model with a basic cycle length of 420 msec. The heart model also displays lengthening in the A-H interval and block in A-V node (Marker 1). Rows 5 and 6 show the increase in the ERP and conduction delay of the A-V node.

is delivered in the atrium. This results in a gradual increase in the AV nodal conduction delay and then a dropped beat occurs in the ventricle due to the increased ERP period of the AV node. The EGMs for a real patient with Wenckebach type A-V nodal response are shown in Fig. 3.6(a). With the VHM, we observe similar behavior, and the gradually increasing ERP and conduction delay are visualized in Fig. 3.6(b).

3.2.2 Validating Models for Closed-loop Model Checking

In model checking, a lot of complex dynamics of the environment are abstracted so that the environment model covers a larger number of environmental behaviors using non-determinism. The validity of the model is obtained by a valid initial model and a rigorous abstraction processes. In Jiang et al.

[2014], we started with a valid detailed deterministic model (as described above) and by applying different abstraction steps we were able to generate a series of non-deterministic heart models. Between each abstraction step, the heart models satisfy a timed simulation relationship (Yamane [2006]) which is described below. The timed simulation guarantees all behaviors are covered in the more abstract model.

For two timed automata $T^1 = \langle S^1, S_0^1, \Sigma^1, X^1, inv^1, E^1 \rangle$ and $T^2 = \langle S^2, S_0^2, \Sigma^2, X^2, inv^2, E^2 \rangle$, a timed simulation relation is a binary relation $\mathsf{sim} \subseteq \Omega^1 \times \Omega^2$ where Ω^1 and Ω^2 are sets of states of T^1 and T^2. We say T^2 time simulates T^1 ($T^1 \preceq_t T^2$) if the following conditions holds:

- Initial states correspondence: $(\langle s_0^1, \mathbf{0} \rangle, \langle s_0^2, \mathbf{0} \rangle) \in \mathsf{sim}$
- Timed transition: For every $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in \mathsf{sim}$, if $\langle s_1, v_1 \rangle \xrightarrow{\delta} \langle s_1, v_1 + \delta \rangle$, there exists $\langle s_2, v_2 + \delta \rangle$ such that $\langle s_2, v_2 \rangle \xrightarrow{\delta} \langle s_2, v_2 + \delta \rangle$ and
  $(\langle s_1, v_1 + \delta \rangle, \langle s_2, v_2 + \delta \rangle) \in \mathsf{sim}$.
- Discrete transition: For every $(\langle s_1, v_1 \rangle, \langle s_2, v_2 \rangle) \in \mathsf{sim}$, if $\langle s_1, v_1 \rangle \xrightarrow{\sigma} \langle s_1', v_1' \rangle$, there exists $\langle s_2', v_2' \rangle$ such that $\langle s_2, v_2 \rangle \xrightarrow{\sigma} \langle s_2', v_2' \rangle$ and $(\langle s_1', v_1' \rangle, \langle s_2', v_2' \rangle) \in \mathsf{sim}$.

As shown in the later chapters, these validated heart models can then be used for closed-loop verification of implantable pacemaker. Both the identification and validation of the heart models can be used to provide more confidence to the verification results, which would be helpful during the medical device certification process.

4

A Dual Chamber Pacemaker Specification

As part of the model-based design, it is important to have a functional and formal model of the device software for testing and formal verification respectively. In our study, we focus on the implantable pacemaker since it is one of the simpler implantable cardiac devices as its functionality is based only on timing and does not consider signal morphology. This serves as a good base case to demonstrate the proposed methodology. In this chapter, we describe the basic specification and formal implementation of a dual chamber pacemaker, as well as a more advanced function on mode switching. The specifications are based on the algorithm descriptions from Boston Scientific manuals (Boston Scientific Corporation [2007b]) and the functional description released as part of the Pacemaker Challenge (Boston Scientific Corporation [2007a]). We aim to answer the following questions here:

- How are the pacemaker's timers specified to maintain the appropriate heart rhythm?
- What happens if new functionality is added to the basic model?

The artificial pacemaker is designed for patients with bradycardia (i.e. slow heart rate). Two leads, one in the right atrium and one in the right ventricle, are inserted into the heart and fixed onto the inner wall of the heart. These two leads monitors the local activation of the atria and the ventricles,

and generate corresponding sensed events (AS, VS) to its software. The software determines the heart condition by measuring time difference between events and delivers pacing events (AP, VP) to the analog circuit when necessary. The analog circuit then delivers pacing signals to the heart to maintain heart rate and A-V synchrony. In order to deal with different heart condition, pacemakers are able to operate in different modes. The modes are labeled using a three character system (e.g. xyz). The first position describes the pacing locations, the second location describes the sensing locations, and the third position describes how the pacemaker software responds to sensing. Here we introduce the widely used DDD mode pacemaker which is a dual chamber mode with sensing and pacing in both atrium and ventricle.

4.1 Basic Specifications of a DDD Pacemaker

The DDD pacemaker has five basic timing cycles triggered by external and internal events, as shown in Fig. 4.1. We decomposed our pacemaker model into five components which correspond to the five timers. $P = LRI \| AVI \| URI \| PVARP \| VRP$. These components synchronize with each other using broadcast channels and shared variables (as shown in Fig. 4.2).

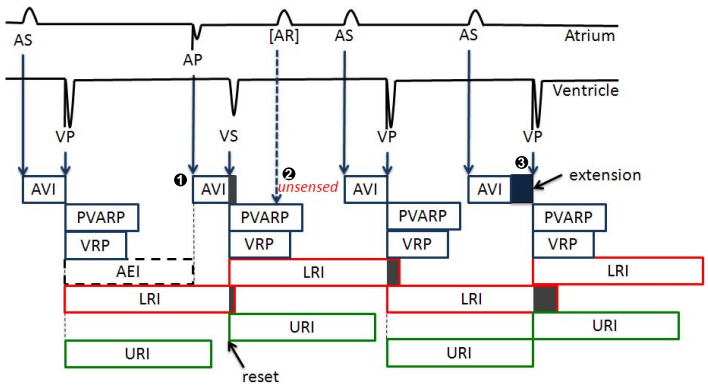


Figure 4.1: Basic 5 timing cycles for a dual chamber pacemaker which include the Lower Rate Interval (LRI), Atrio-Ventricular Interval (AVI), and Upper Rate Interval (URI). Also included are the blanking intervals, Post Ventricular Atrial Refractory Period (PVARP) and Ventricular Refractory Period (VRP), to inhabit action by the pacemaker.

4.1.1 Lower Rate Interval (LRI)

The Lower Rate Interval (LRI) component is shown in Fig. 4.2(a). This component defines the longest interval allowed between two ventricular events, thus keeping the heart rate above a minimum value. In DDD mode, the LRI interval is divided into a V-A interval (TLRI-TAVI) and a A-V interval (TAVI). The LRI component maintains a maximum V-A delay while the AVI component maintains a maximum A-V delay so together they maintain the maximum V-V delay. In the LRI component, the clock is reset when a ventricular event (VS, VP) is received. If no atrial event has been sensed (AS), the component will deliver atrial pacing (AP) after TLRI-TAVI.

4.1.2 Atrio-Ventricular Interval (AVI) and Upper Rate Interval (URI)

The function of the AVI component defines the longest interval between an atrial event and a ventricular event. If there is no ventricular event (VS) within TAVI after an atrial event (AS, AP), and the time since the last ventricular event (VS, VP) is longer than TURI, the component will deliver ventricular pacing (VP). The URI limits the ventricular pacing rate by enforcing a lower bound on the times between consecutive ventricle events. The UPPAAL design of AVI and URI component is shown in Fig. 4.2(b) and (c).

4.1.3 Post Ventricular Atrial Refractory Period (PVARP) and Post Ventricular Atrial Blanking (PVAB)

Ventricular events, especially Ventricular Pace (VP) are sometimes so strong that the atrial lead can sense the activation as well. This signal may be falsely recognized as an atrial event and disrupt normal pacemaker function. This scenario is called crosstalk and was discussed in our previous work (Jiang and Mangharam [2011]). In order to prevent this undesired behavior, and filter potential noises, there is a blanking period (PVAB) followed by a refractory period (PVARP) for the atrial events after each ventricular event (VS, VP). Atrial events during PVAB are ignored and atrial events during PVARP trigger AR! events which can be used in some advanced diagnostic algorithms. The UPPAAL design of PVARP component is shown in Fig. 4.2(d).

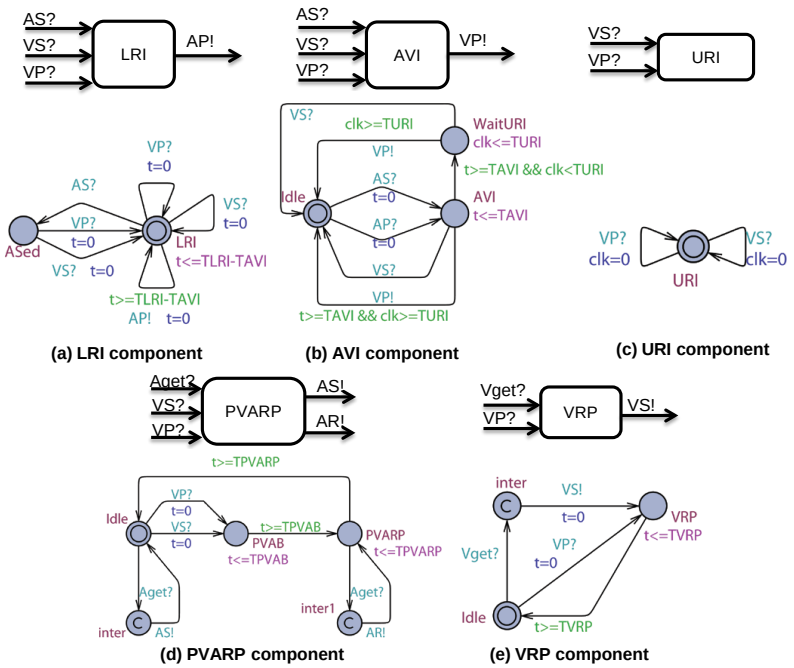


Figure 4.2: Five basic timing cycles for a dual chamber pacemaker, which include the Lower Rate Interval (LRI), Atrio-Ventricular Interval (AVI), and Upper Rate Interval (URI). Also included are the blanking intervals, Post Ventricular Atrial Refractory Period (PVARP) and Ventricular Refractory Period (VRP), to inhabit action by the pacemaker.

4.1.4 Ventricular Refractory Period (VRP)

The VRP follows each ventricular event (VP, VS) to filter noise and early events in the ventricular channel which could otherwise cause undesired pacemaker behavior. Fig. 4.2(e) shows the UPPAAL design of VRP component.

4.2 Mode Switch Operation: Atrial Tachycardia Response

Supraventricular Tachycardia (SVT) is an arrhythmia with an abnormally fast atrial rate. Typically, in a heart without pacemaker, the AV node, which has a long refractory period, can filter most of the fast atrial activations during SVT, thus the ventricular rate remains relatively normal. Fig. 4.3(a) demonstrates a

pacemaker event trace during SVT, with a pacemaker in ODO mode, which just sensing in both channels. As there is no pacing in ODO mode, the heart is in open-loop with the pacemaker. In this particular case, every 3 atrial events (AS) correspond to 1 ventricular event (VS) during SVT. As an arrhythmia, SVT is still considered a safe heart condition since the ventricles operate under normal rate and still maintain adequate cardiac output.

However, in the closed loop case with the DDD pacemaker, the AVI component of a dual chamber pacemaker is equivalent to a virtual pathway in parallel to the intrinsic conduction pathway between the atria and the ventricles. The pacemaker tries to maintain 1:1 A-V conduction and thus increases the ventricular rate inappropriately to match the atrial rate. This is known as Pacemaker Mediated Tachycardia (PMT) as the heart would have been safe without the pacemaker and its virtual pathway. Fig. 4.3(b) shows the pacemaker trace of the same SVT case with DDD pacemaker. Although half of the fast atrial events are filtered by the PVARP period ([AR]s), the DDD pacemaker still drives the closed-loop system into 2:1 A-V conduction with faster ventricular rate. Maintaining A-V delay is less important than maintaining an appropriate ventricular rate. The DDD pacemaker violates a higher priority requirement in order to satisfy a lower priority requirement, which is inappropriate.

Pacemaker manufacturers have designed algorithms to detect and terminate these behaviors. Intuitively, the mode-switch algorithm first detects SVT. After confirmed detection, it switches the pacemaker from a dual-chamber mode to a single-chamber mode. During the single-chamber mode, the A-V synchrony function of the pacemaker is deactivated thus the ventricular rate is decoupled from the fast atrial rate. After the algorithm determines the end of SVT, it will switch the pacemaker back to the dual chamber mode. The mode-switch algorithm (also known as atrial tachycardia response) specification we use is similar to the one described in the Boston Scientific pacemakers' manual (Boston Scientific Corporation [2007b]). The algorithm first measures the interval between atrial events outside the blanking period (AS, AR). The interval is considered as *fast* if it is above a threshold (*Trigger Rate*) and *slow* otherwise. In our UPPAAL model we model it as INT (see Fig. 4.5 (1)). A counter CNT increments for *fast* events and decrements for *slow* events (see Fig. 4.5 (2)). After the counter value reaches the *Entry Count*, the algorithm

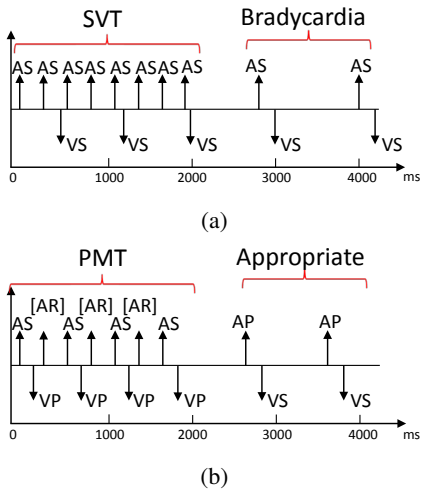


Figure 4.3: Benign open loop case: SVT without a pacemaker or with a pacemaker in sense-only mode (ODO) (b) Dangerous closed-loop-case SVT with DDD pacemaker which tries to match the fast atrial rate with a corresponding (and dangerous) fast ventricular rate.

will start a *Duration* (DUR) ,which is a time interval used to confirm the detection of SVT (see Fig. 4.5 (3)). In the *Duration*, the counter keeps counting. If the counter value is still positive after the *Duration*, the pacemaker will switch to the VDI mode (*Fallback mode*). In the VDI mode, the pacemaker only senses and paces the ventricle. At any time if the counter reaches zero, the *Duration* will terminate and the pacemaker is switched back to DDD mode. In our UPPAAL model of the mode-switch algorithm, we use nominal parameter values from the clinical setting. We define *trigger rate* at 170bpm (350ms), *entry count* at 8, *duration* for 8 ventricular events and *fallback mode* as VDI.

In order to model both DDD and VDI modes and the switching between them, we made modifications to the AVI and LRI components. In each component two copies for both modes are modeled, and switch between each other when switching events (DDD, VDI) are received. During VDI mode, VP is delivered by the LRI component instead of the AVI component. The clock values are shared between both copies in order to preserve essential intervals even after switching. The modified AVI (AVI') and LRI (LRI')components are shown in Fig. 4.4.

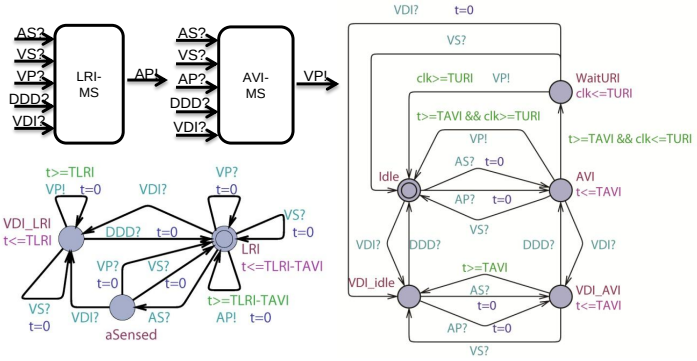


Figure 4.4: (a) After switching to VDI mode, the new LRI component LRI' maintains a minimum V-V interval; (b) After switching to VDI mode, the new AVI component AVI' keeps track of the time after each atrial events.

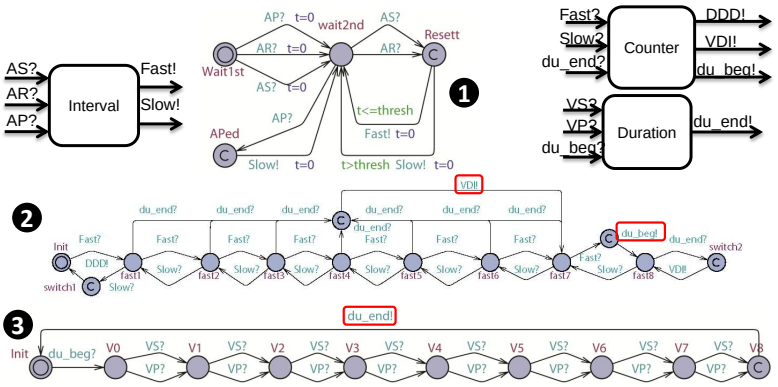


Figure 4.5: (1) Component INT: An atrial event (AS,AR) arrives before thresh after the previous atrial event is regarded as a fast event. Atrial event arrives after thresh and AP are regarded as slow event; (2) Component CNT: After 8 fast event the algorithm will start a duration by sending du_beg and will switch to VDI mode when the duration ends (du_end); (3) Component DUR :The duration length is 8 ventricular events (VS,VP)

5

Closed-loop Model Checking

Model checking is a technique in which the state space of the model under investigation is automatically and exhaustively explored to identify executions or states that violate specified properties. Violations of the properties are returned by the model checkers as *counter-examples*, which can be used by designers to revise the design. In the application of verification of properties in medical devices like implantable pacemaker, model checking can be used to identify known and unknown mechanisms for inducing hazards. This is extended to checking the heart-pacemaker closed-loop models against physiological hazards (e.g. when the pacemaker provides inappropriate therapy which drives the heart to an unsafe state).

Due to the curse of dimensionality as models get more complex, and hence the large computational cost, there are usually restrictions on the formalism and the complexity of the models under investigation. Using abstract models of the actual system adds the responsibility of proving the conformance between the abstract model and the real system. Assumptions made during abstractions may also introduce false-positives and/or false-negatives into the model checking results. Back in Chapter 2, we introduced a set of heart models with different abstraction levels, that can be used to cover behaviors of physiological conditions using non-determinism. In Chapter 4, we

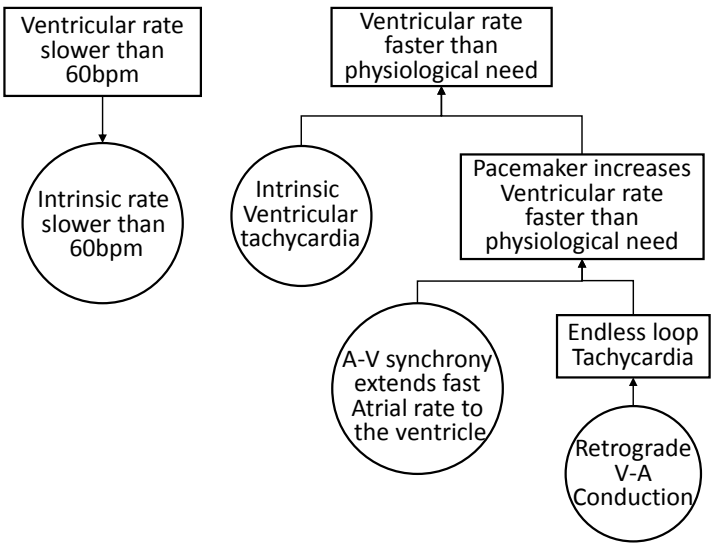


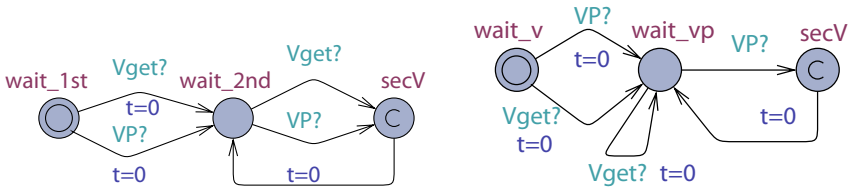
Figure 5.1: Sample Fault Tree Analysis of the physiological conditions leading to the lower rate limit and upper rate limits

then modeled a dual chamber pacemaker algorithm using timed-automata without doing abstractions. In this chapter, we use model checker UPPAAL to evaluate the pacemaker model against safety properties under different heart conditions captured by the heart models. Techniques to eliminate false-positives by refining the heart models are also discussed. We will try to answer the following questions:

- How do model checking results fit into the regulation framework?
- How do we find the appropriate abstraction level of the environment model for each physiological requirement?
- How do we interpret abstract counter-examples returned by model checker?

5.1 Risk Analysis for Implantable Pacemaker

Implantable pacemakers are designed to treat bradycardia by increasing the heart rate with external pacing. Therefore the heart rate should not only be increased to the minimum physiological need, but also should not be increased



(a) Monitor PLRI_test (b) Monitor PURI_test

Figure 5.2: (a) Monitor for LRL: Interval between two ventricular events should be less than TLRI, (b) Monitor for URL: Interval between a ventricular event and a VP should be longer than TURI

beyond physiological need. Fig. 5.1 demonstrates two Fault Tree Analysis (FTA) for these two top level hazards. In the remaining chapter we first specify hazards as properties, and use model checking to evaluate whether these hazards have been mitigated by the pacemaker. Then for one of the mitigation algorithm, we examine the mitigation effectiveness and the residue hazard.

5.2 Mitigating Top-level Hazards

The most essential function for the pacemaker is to treat bradycardia by maintaining the ventricular rate above a certain threshold. We define the region where the ventricular rate is slow, as unsafe. The monitor PLRI_test is designed to measure intervals between ventricular events and is shown in Fig. 5.2(a). For property

$$\varphi_{LRI} = \text{A[] (PLRI\_test.secV imply PLRI\_test.t} \leq \text{TLRI)}$$

we have a closed-loop system with heart model H_d (described in at the end of Chapter 2):

$$H_4 \| P \| PLRI\_test \models \varphi_{LRI}$$

The pacemaker is not designed to treat tachycardia so it can only pace the heart to increase its rate and cannot slow it down. To mitigate the hazard that the pacemaker may increase the heart rate above physiological need, an Upper Rate Interval (URI) is specified such that the pacemaker can increase the ventricular rate up to this limit.

We require that a ventricle pace (VP) can only occur at least $TURI$ after a ventricle event (VS, VP). The monitor PURI_test is shown in Fig. 5.2(b).

For the property

$$\varphi_{URI} = \text{A[] (PURI\_test.secV imply PURI\_test.t} \geq \text{TURI)}$$

we have:

$$H_d \| P \| PURI\_test \models \varphi_{URI}$$

5.3 Evaluate the Mitigation

As described in Chapter 4.2, the mode switch algorithm has been designed to mitigate the hazard that the A-V synchrony function of DDD pacemaker extends fast atrial rate to the ventricle. It is important to ensure the effectiveness of the algorithm without inducing other top-level hazards. In this section we first show the existence of the hazard in a pacemaker without the mode-switch algorithm. If the algorithm if effective the hazard will not exist after introducing the algorithm.

5.3.1 Existence of Pacemaker Mediated Tachycardia during SVT

The monitor Pv_v is designed to show existence of PMT during SVT. It goes to the error state if the ventricular rate drops below the Upper Rate Limit (Fig. 5.3).

We specify $\varphi_{MS} = E[](notPv\_v.err)$
which verifies the existence of PMT. The heart model H_e in Fig. 2.9 is not suitable for this property since the non-deterministic conduction of component P_3 does not capture the blocking property of the AV node, which is the key in PMT. We use a more refined model H_d which has AV node modeled. To identify the PMT scenario, we first set $H_d.N^1.Trest\_min < 100$ so that the atrial rate can be high and $H_d.N^2.Trest\_min > TURI$ so that the intrinsic heart rate is less than TURI. The property is first verified on pacemaker

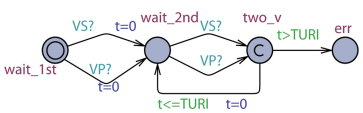


Figure 5.3: Monitor Pv_v for SVT: There exists an endless sequence in which interval between ventricular events is at most TURI

without the mode-switch algorithm. We have $H_d \| P \| Pv\_v \models \varphi_{MS}$ and the evidence returned by the model checker illustrates the PMT scenario.

5.3.2 Verification against fundamental safety properties

For a pacemaker with the mode switch algorithm:

P_2=LRI'‖AVI'‖URI‖PVARP‖VRP‖INT‖CNT‖DUR,

we verify the same fundamental safety properties on the pacemaker model with mode-switch algorithm. We have:

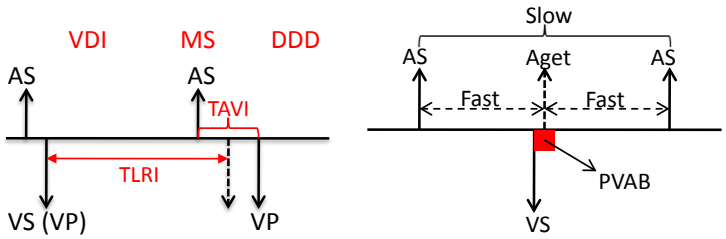
$$H_d \| P_2 \| PURI\_test \models \varphi_{URI}$$

$$H_d \| P_2 \| PLRI\_test \not\models \varphi_{LRI}$$

The Upper Rate Limit property still holds, but the Lower Rate Limit property is violated. The counterexample is proved to be valid after checking the trace of more refined heart models. By analyzing the trace we found that when the pacemaker is switching from VDI mode to DDD mode, the responsibility to deliver VP switched from LRI component to AVI component. Since the clock reference is different (Ventricular events in LRI component and Atrial events in AVI component), the clock value for delivering the next VP is not preserved. As a result, if an atrial event which triggered the mode-switch from VDI to DDD happens within [TLRI-TAVI, TLRI) after the last ventricular event, the next ventricular pacing will be delayed by at most TAVI time, which violates the Lower Rate Limit property (Fig. 5.4(a)).

5.3.3 Verification of the Mode-Switch Algorithm

After implementing the mode-switch algorithm, we verified the model against the same existence property. We expect the violation of this property, since during VDI mode the ventricular rate of the heart model is less than the Upper Rate Limit and will not trigger ventricular pacing. However, this property is still satisfied, indicating the mode-switch algorithm failed to eliminate the PMT scenario. The evidence trace returned by UPPAAL shows that a subset of atrial events fall into the blanking period after a ventricular event (see Fig. 5.4(b)). As a result, two fast events are reduced to one slow event and mode switch may never happen. This scenario does exist in all our refined heart models, we conclude that the trace is physiologically feasible.



(a) (b)

Figure 5.4: (a) Safety Violation: VP is delayed due to the reset of timer during mode-switch, (b) Correctness Violation: The blocking period may block some atrial events, turning two *Fast* events to one *Slow* event (Jiang et al. [2012b]

)

The mode-switch algorithm in our pacemaker model can not terminate all PMT behaviors as specified as certain mild PMT events are admissible.

5.4 Abstraction Tree for Environment Modeling

In the previous two sections, abstractions and selections of the heart models are performed manually, which require knowledge of both electrophysiology and model checking. Counter-examples returned from abstract models can be difficult to interpret by domain experts. One abstract counter-example could be produced by multiple physiologically valid conditions, which causes ambiguity. Thus, a rigorous framework is necessary to balance the need to cover a wide range of environmental conditions and the need to provide counter-examples to the physicians within their physiological context. The framework must also allow non-domain experts to perform verification, and establish 'hand-off' points where the results of verification can be handed back to the experts for interpretation.

In this section, we use a set of domain-specific abstraction rules based on physiological knowledge to ensure the physiological relevance of the behaviors introduced into the abstract models. The rules are applied to an initial set of physiological models to obtain an abstraction tree, which will be used for closed-loop model checking of the pacemaker. A straightforward search procedure is then used to conduct model checking using suitable heart models and return the most concrete and unambiguous counter-examples to the physicians for analysis. In this framework, physiological knowledge is only

needed when constructing the initial model set and when analyzing counter-examples. The application of the physiological abstraction rules and the verification procedure can be automated. The proposed method can potentially be generalized to other domains in which the device operates in a large variety of environmental conditions. More information regarding this research can be found in Jiang et al. [2015].

Step 1: Abstraction Tree construction

A set of heart models corresponding to different heart conditions are first developed. The list can be expanded as new heart conditions are discovered. Because we start from a set of initial models, and each one may be abstracted using a number of abstraction rules, we have a choice of which rules to apply to which models, and the order in which to apply them. Depending on which rule is applied when, we end up with different abstract models. Thus an *abstraction tree* T_{HM} for the heart is created, as shown in Fig. 5.5.

Step 2: Requirement encoding

The following requirement is designed to prevent the pacemaker from pacing too fast: "If the intervals between self-activations of the atria are between 300ms to 1000ms (60bpm - 200bpm), the intervals between ventricular paces should be no shorter than 500ms." Self-activation of the atria can be expressed using the location and clock of node automaton N_A. The requirement can be formalized using the monitor $M_{sing}(VP, 500, \infty)$:

$$Req1 : N_A.loc = Rest \wedge N_A.t \in [300, 1000] \Rightarrow \neg M_{sing}.loc == Err$$

Step 3: Choosing appropriate heart models for the requirement

To verify the closed-loop system with pacemaker model PM and abstraction tree T_{HM} (Fig. 5.5) against requirement $Req1$, the most abstract appropriate models are selected from the tree. The single event monitor M_{sing} from Fig. 5.6(a) with variables $Var(M_{sing}) = \{M_{sing}.t, M_{sing}.loc\}$ is used for this requirement. Model checking is performed on the closed-loop system including the heart model M_H, the pacemaker model M_P, and the monitor M. The requirement φ_P can be then represented with TCTL formula:

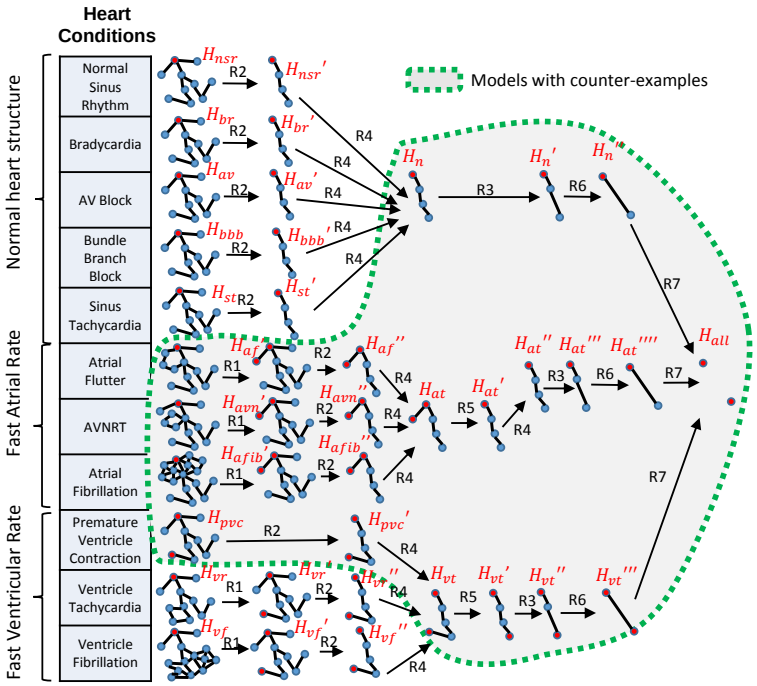


Figure 5.5: Heart Model Abstraction Tree with arrows showing the direction of the abstraction process starting from detailed models of different heart conditions. Model refinement is in the opposite direction.

A[] (not M.Err)

The variables in the requirement are:

$$Var(Req1) = \{N_A.t, N_A.loc, M_{sing}.loc\}$$

At the root of the tree H_{all}, we have $\{N_A.t, N_A.loc\} \not\subset Var(H_{all}) \cup Var(M_{sing})$. So H_{all} is not appropriate for $Req1$. All the children of H_{all}:

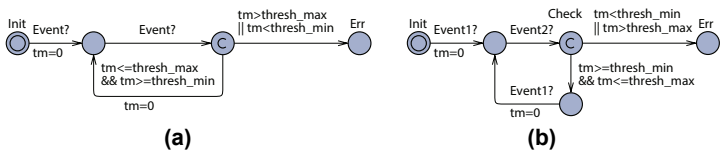


Figure 5.6: (a) M_{sing} for single event; (b) M_{doub} for two events

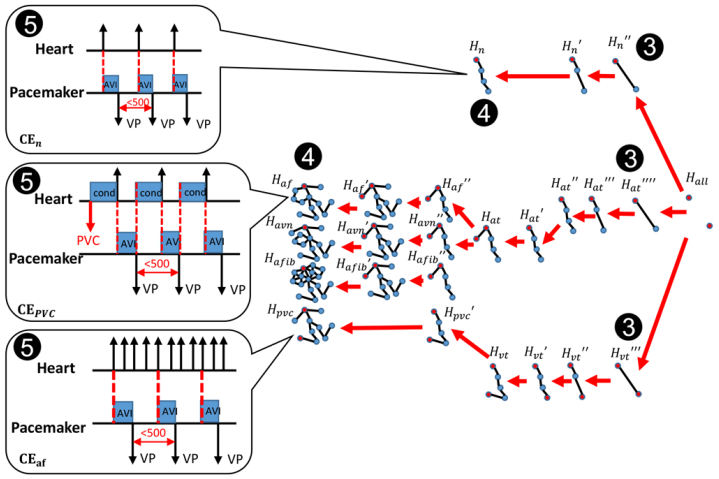


Figure 5.7: Model refinement: Finding the most concrete counter-examples using the abstraction tree

$H_n'', H_{at}'''', H_{vt}'''$ are appropriate for $Req1$, thus these three heart models are output as the most abstract models that are appropriate for $Req1$.

Step 4: Return the most concrete counter-examples

After the appropriate models for $Req1$ are selected, we have the initial set $HM = \{H_n'', H_{at}'''', H_{vt}'''\}$. By model checking on all three initial models in UPPAAL we have:

$$H_n''||PM \not\models Req1;\ H_{at}''''||PM \not\models Req1;\ H_{vt}'''||PM \not\models Req1$$

The abstraction tree is then further explored. The heart models with counter-examples are illustrated in Fig. 5.7, and the most refined heart models with counter-examples are: $H_n; H_{pvc}; H_{af}; H_{avn}; H_{afib}$.

Step 5: Analysis of the counter-examples

The counter-examples are then shared with physicians for analysis. In Fig. 5.7 we highlight three counter-examples. In the first counter-example, the intrinsic heart signals over time with up arrows as atrial activations and down arrows as ventricular activations. The signal for the second counter-example shows the pacemaker outputs with up arrows as atrial pacing and down arrows as ventricular pacing.

Counter-example CE_n is returned by H_n and none of its children models violate the requirement. By careful analysis we found that CE_n features the combination of fast intrinsic atrial rate and prolonged A-V conduction delay, which is the combination of heart conditions H_{st} and H_{av}. This scenario shows that the abstraction rules can introduce physiological heart conditions that were not explicitly modeled in the initial model set. The pacemaker improved the open-loop heart condition by pacing the ventricles AVI after each atrial event, which is a correct operation of the pacemaker despite the requirement violation.

Counter-example CE_{pvc} has a very similar execution to CE_n. However, the activations of the atrial node are triggered by retrograde conduction from ventricle to the atrium initialized by ventricular paces (marker cond). The atrial activations trigger another ventricular pace after AVI, which will trigger another retrograde conduction. In this case, the heart rate is inappropriately high, which corresponds to a dangerous closed-loop behavior referred to as *Endless Loop Tachycardia*.

In counter-example CE_{af}, the atrial rate is very high, which is also a suboptimal but not dangerous heart condition. However, the ventricular rate can stay normal due to the blocking property of the AV node. Despite the filters in the pacemaker, the pacemaker still paces the ventricle for every 3 atrial activations, which extends fast atrial rate to more dangerous fast ventricular rate. This scenario is referred to as Atrial Tachycardia Response of a pacemaker.

From the analysis, pacemaker operations in CE_{pvc} and CE_{af} must be revised. However, the revision should not affect the behavior in CE_n. This example demonstrates that counter-examples from refined models provide more physiological context of the requirement violations, and distinguish the physiological conditions that can trigger the violations. The information is helpful for debugging and improving the algorithm. The physicians can also improve the physiological requirement so that these heart conditions can be then considered case by case.

Discussion:
Model checking is not widely use in industry, in part, due to scalability issues and also because domain expertise must be a skill possessed by the verification engineer. However, with rigorous abstraction of the system and its envi-

ronment, model checking can be used to identify known and even unknown mechanisms to induce hazards. In this chapter, we use a model of a dual chamber pacemaker as an example to demonstrate the use of model checking during risk analysis. During the process we identified the need to refine the heart models to eliminate false-positives introduced during the abstraction, and demonstrated the difficulty to do so manually. The abstraction tree approach is then proposed to reduce the effort needed for both the developers and the domain experts, which makes model checking a viable approach for providing safety and effectiveness evidence.

6

Closed-loop Model Simulation and Testing

Model checking is performed on abstract models of the system, which is at an early stage in the development process. The verified system model during model checking is then translated into a Stateflow model, which is a step towards simulation-based testing and subsequently to code generation. Closed-loop simulation/testing are performed on more refined deterministic models, and on the actual system, complement model checking in terms of resolving ambiguities within abstract counterexamples. We aim to answer the following questions here:

- How are abstract models translated to deterministic models for simulation-based testing?
- What system level issues are best tested at the platform level?

In this chapter, we first describe an approach to automatically translate formal models that are verified in UPPAAL to Stateflow charts for simulation-based testing, and then code generated to run on an embedded platform. Following this, we demonstrate two examples that cannot be explicitly modeled using abstract semantics and must be tested.

6.1 UPPAAL to Stateflow Automated Model Translation

A model translation tool, UPP2SF (Pajic et al. [2014]) was developed to translate UPPAAL models to Stateflow (see Fig. 6.1(a)). Consider an UPPAAL model with automata $P_1, ..., P_n$. After UPP2SF translation, a two-level Stateflow chart is generated as in Fig. 6.1(b). The chart consists of parallel states $P_1, ..., P_n$ (referred to as the *parent* states) derived from the automata, parallel states $Gc\_x_1, ..., Gc\_x_m$ (referred to as *clock states*) that model all global clocks $x_1, ..., x_m$ from the UPPAAL model, and the state Eng that is used as the chart's control execution engine. Moreover, the chart has predefined global data variables (and constants) with appropriate ranges and initial values derived from the UPPAAL model. Since all automata in UPPAAL are active simultaneously, the obtained Stateflow chart is a collection of parallel states with unique execution orders. Also, in every UPPAAL automaton exactly one location is active at a time. Thus, each of the parent states is a collection of exclusive states, extracted from locations in the corresponding UPPAAL automaton.

In Pajic et al. [2014], we showed that for a large class of UPPAAL models, the generated Stateflow models generated by UPP2SF preserve behaviors of the initial UPPAAL models. The translation tool can be used to estimate

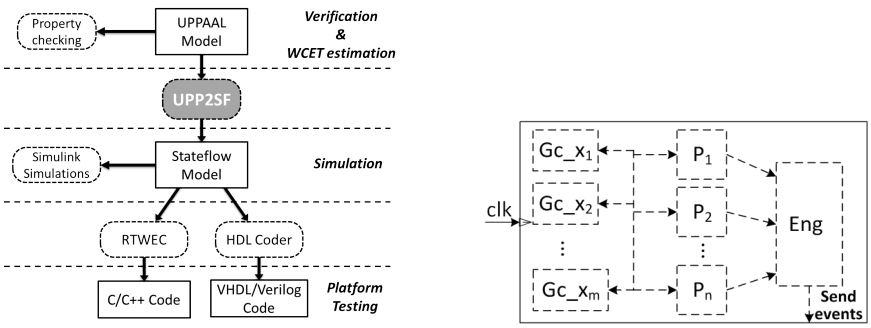


Figure 6.1: (a) Model Driven Design framework: From UPPAAL to Stateflow to generated code – covering model verification, simulation-based testing and platform testing. (b) Structure of Stateflow charts of the pacemaker's five basic timing cycles (from Fig. 4.2) derived by the UPP2SF model translator. Parent states $P_1, ..., P_n$ are derived from automata, while the *clock* states $Gc\_x_1, ..., Gc\_x_m$ model all global clocks $x_1, ..., x_m$ from the UPPAAL model. The state Eng is used to control execution of the chart.

the worst case execution time (WCET) during modeling and model checking stage in UPPAAL, and facilitates development of modular code from timed-automata based models.

Fig. 6.2 demonstrates the Stateflow chart generated from the UPPAAL model of the DDD pacemaker model in Fig. 4.2 using the UPP2SF tool. We generated C code from the pacemaker Stateflow chart using the Simulink Coder. The code structure is shown in Fig. 6.3. The code was generated for the general embedded real-time target and as a result we obtained the main procedure, `rt_OneStep`, which processes the three input events, $VinB$, $AinB$ and clk. To ensure that the model semantics are preserved (modulo the execution time), clk input events should be created every 1ms, followed by the procedure's activation. This makes it suitable for implementation on top of a real-time operating system (RTOS).

The pacemaker code generated by the Simulink Real-Time Workshop's Embedded Coder was executed on nanoRK (Nano-RK [2007]), a fixed-priority preemptive RTOS that runs on a variety of resource constrained platforms. We tested the implementation on the TI MSP-EXP430F5438 Experimenter Board interfaced with a signal generator that provides inputs for the pacemaker code (Fig. 6.4). More details regarding UPP2SF translation and platform testing can be found in Pajic et al. [2014].

6.2 Pacemaker Oversensing and Crosstalk

Oversensing is a general term for inappropriate sensing caused by noise or far-field signals. It's very common among pacemaker malfunctions and it may result in failure to pace (Beaumont et al. [1995], Fuertes and Toquero [2003]), competitive pacing and inappropriate therapy. Crosstalk is a special case for oversensing which occurs when the pacemaker stimulus in one chamber is sensed in the other chamber. It happens when two leads are close to each other or pacing signal in the other chamber is too strong. It is common that the ventricular lead is placed in the right ventricle outflow tract, which is close to the atrium (Saxonhouse et al. [2005]). Fig. 6.5(a) shows simulated EGMs from a patient with bradycardia and complete heart block. During atrial pacing (AP), the pacing signal is sensed by the ventricular lead 53 ms after the AP. (Marker 1) It is treated as ventricular sense (VS) signal

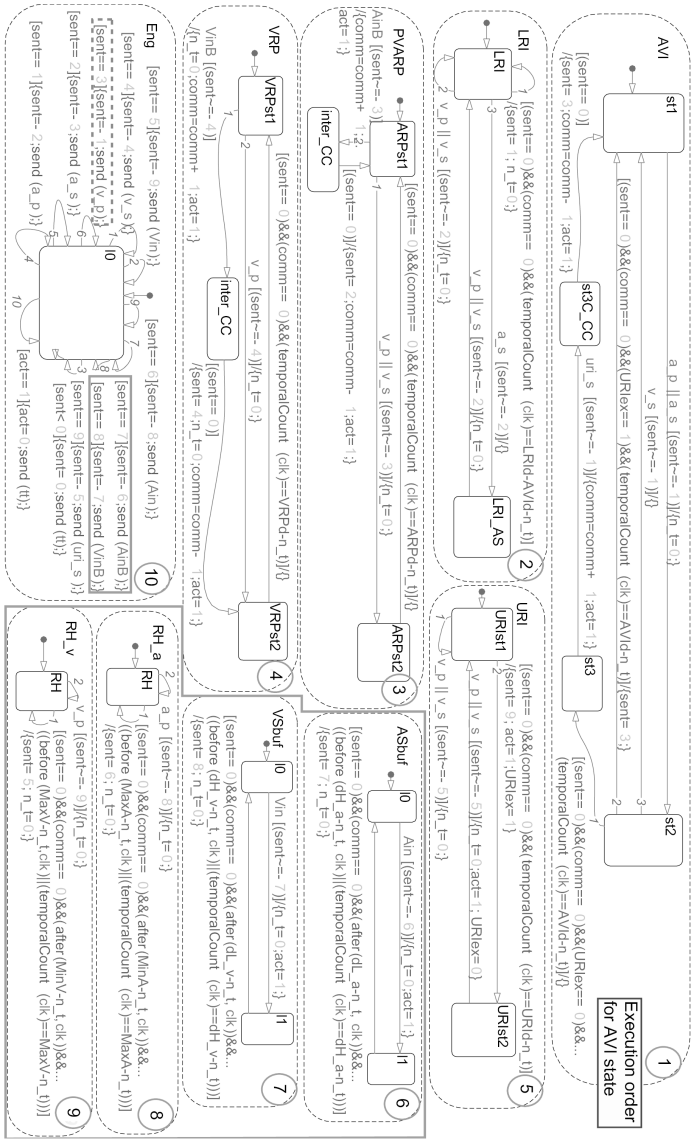


Figure 6.2: Pacemaker Stateflow chart converted from the UPPAAL model in Fig. 4.2 using UPP2SF; the heart and buffer models are highlighted.

and thus inhibits the subsequent ventricular pacing (VP). This is indicated by no QRS-wave in the ECG channel. (Marker 2) For a patient with complete

Listing 1. bitsForTID0 definition

```
struct {
   uint_T is_AVI:3;
   uint_T is_LRI:2;
   uint_T is_PVARP:2;
   uint_T is_VRP:2;
   uint_T is_URI:2;
   uint_T is_active_AVI:1;
   uint_T is_active_LRI:1;
   uint_T is_active_PVARP:1;
   uint_T is_active_VRP:1;
   uint_T is_active_URI:1;
   uint_T is_active_Eng:1;
   uint_T is_Eng:1;
   uint_T URI_ex:1;
 } bitsForTID0;
```

Listing 2. Rt_OneStep procedure

```
detect active inputs;
for each of the input events {
  if EventName is active  {
    sf_previousEvent = _sfEvent_;
    _sfEvent_ = EventName;
   c1_ChartName()
    _sfEvent_ = sf_previousEvent;
  }
}
update the outputs;
update the input events states;
```

Listing 3. c1_ChartName() procedure

```
increase counters for _sfEvent_;
for each parallel state {
 processState();
}
```

Listing 4. processState() procedure

```
if (rtDWork.bitsForTID0.is_active_NAME != 0) {
 switch (rtDWork.bitsForTID0.is_NAME) {
  case SubStateName1:
   /* the loop below is - checkTrans();*/
   for all transitions in ex. order {
    if transition enabled {
     execution transition actions;
     reset corresponding temporal counters;
     update rtDWork.bitsForTID0.is_NAME;
     break for
   }  }
   break;
  case SubStateName2:
   checkTrans();
   break;
  ...
  default:
   rtDWork.bitsForTID0.is_NAME=NoActiveChild;
   break;
 }  }
```

Listing 5. broadcast_tt() procedure

```
static void broadcast_tt(void) {
  int16_T sf_previousEvent;
  sf_previousEvent = _sfEvent_;
  _sfEvent_ = event_tt;
 c1_ChartName();
  _sfEvent_ = sf_previousEvent;
}
```

Figure 6.3: Structure of the pacemaker code obtained from the Stateflow chart shown in Fig. 6.2.

heart block this will cause dangerous ventricular asystole, meaning a long time without ventricular events.

Increasing the sensing threshold of the ventricular channel can prevent false sensing. In Fig. 6.5(b), the small signals in ventricular EGM are ignored and ventricular pacing are successfully delivered.

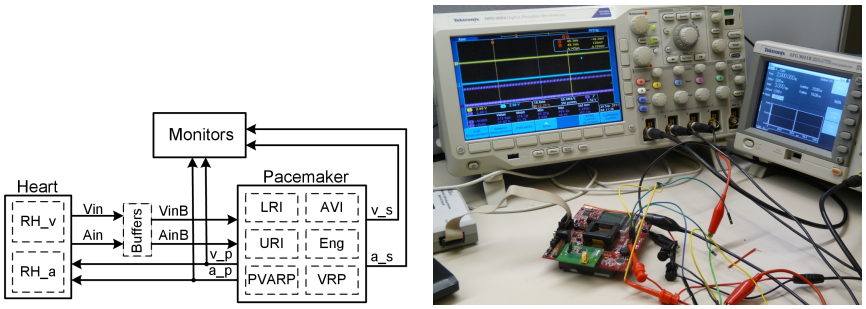


Figure 6.4: (a) Structure of the pacemaker model in UPPAAL and Stateflow, including the interaction between the pacemaker and heart, and the monitors used for verification. (b) Hardware setup with MSP430F5438 experimenters board.

6.3 Lead Displacement

Lead displacement affects many patients and can result in inappropriate or ineffective therapy. Fig. 6.6. (b) shows the simulation result for the pacemaker function when the leads are in their designated location. From the figure we can observe: 1) Each P-wave is initialized by an Atrial Pace signal. 2) Each QRS complex is initiated by a ventricular pacing signal. 3) The interval between AP and VP is 150 ms, which matches the programed AVI period.

One common case for lead dislodge is shown in Fig. 6.6.(a), where the atrial lead has fallen into the right ventricle outflow tract. In this case the atrial lead senses from the ventricle rather than atrium and atrial pacing will initiate a ventricular event. Fig. 6.6.(c) shows the simulated EGMs in this case. The figure reveals several facts: 1) No P wave is sensed or tracked (Marker 1). 2) Atrial Pace initiates an abnormal, wide QRS which is then sensed by the ventricle lead (Marker 2). 3) Intermittent appearance of VP on QRS 110 ms after the AP. The ventricular lead can receive signal from: 1) pacing signal sent from the atrial lead, 2) the intrinsic A-V conduction path. The two paths are shown in Fig. 6.6.(a) and form a timing race condition. When the signal from the atrial lead arrives the ventricular lead first, it will trigger VS. If the intrinsic signal arrives the ventricular lead during the VSP sensing window (defined in previous section), it will trigger VSP. Although the pacing is 'safe' because the pacing is early enough to avoid the vulnerable refractory period, the damage caused by pacing on depolarized

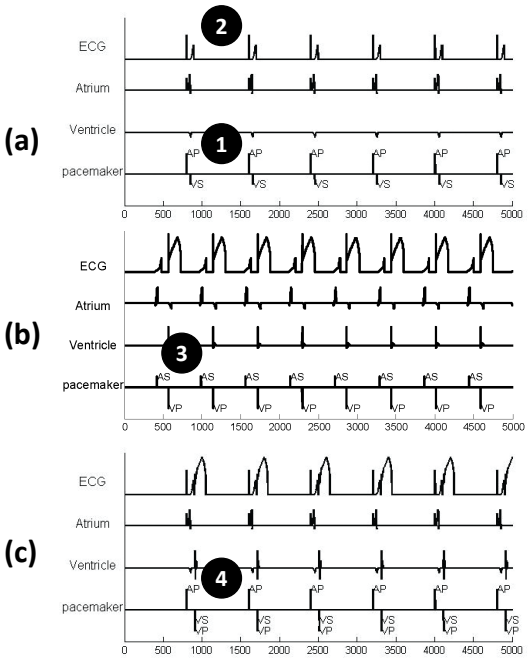


Figure 6.5: Crosstalk between pacemaker leads with high sensitivity in the ventricle, adjusted sensitivity and ventricular safety pacing

tissue is currently a matter of much investigation.

6.3.1 Summary:

In this chapter, we first introduced a model translation method from UPPAAL timed automata models to Stateflow charts. Combined with Simulink coder, the tool chain provides rigorous evidence of traceability from physiological requirements to the C code implementation. Oversensing, crosstalk and lead displacement are three examples in which the cause of the problem is not modeled in the abstract interface in the timed-automata model. In such cases, closed-loop testing with the more refined heart models and EGM interface can be used to provide safety evidence.

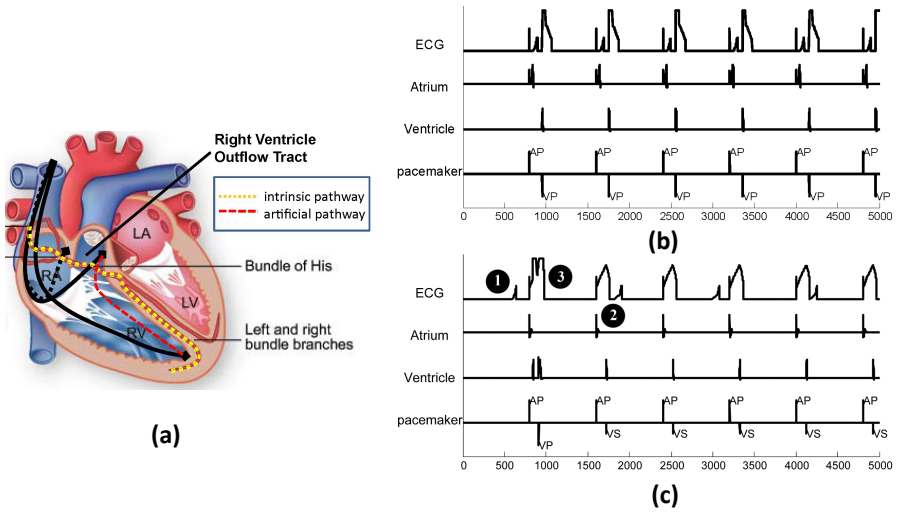


Figure 6.6: (a) Dotted line shows the location where the atrial lead should be (b) Pacemaker function before lead dislodge. (b) Pacemaker function after lead dislodge

7

Discussion and Open Challenges

Closed-loop medical devices like implantable cardiac devices have both diagnostic and therapeutic capabilities and interact with the patient autonomously in closed-loop. Their autonomy makes them among the highest risk devices which require the most stringent regulation. Currently, clinical trials are the primary means to identify risks associated with the closed-loop interaction between the devices and the patient. While such clinical trials are a necessity they are expensive and ineffective for verification of the safety and efficacy of medical device software.

Model-based design can potentially enable closed-loop evaluation earlier in the design process. This approach requires validated physiological models that represents the closed-loop interaction of different physiological conditions from the device's perspective. In this effort, we use an implantable pacemaker as an example to demonstrate the application of model-based design in providing safety and effectiveness towards "regulatory grade evidence" of the device and describe how these activities align into the regulatory process.

We developed heart models that capture the electrical behaviors across a range of heart conditions, and tailored the heart models for closed-loop model checking and closed-loop testing, which have different requirements. In closed-loop model checking, an abstract model of the pacemaker was val-

idated against physiological requirements. We identified the need for heart models at different abstraction levels and demonstrated an automated approach to select the most appropriate heart model for specific safety requirements. The abstract model of the pacemaker was then automatically translated to Stateflow chart using a model translation tool and then generated into C code implementation. With this model-driven design, we are able to retain the safety properties of the modeled device from verified models to verified code. In closed-loop testing, we use more refined heart models to capture mechanisms that were not captured in the abstract heart models and evaluated pacemaker algorithms.

Eventually we aim to conduct *Model-based Clinical Trials* with automated approaches to capture and tune patient-specific electrophysiological heart models using data acquired from ablation procedures. By applying parametric and sensitivity analysis to a small sampling of real patient heart models, across a select set of cardiac conditions, to derive a statistically significant, and physiologically relevant, population of patient models. This allows us to explore a wider range of heart behaviors and expose more corner cases to isolate software safety issues prior to an actual clinical trial. Using certified heart models, a model-based clinical trial provides additional confidence in the closed-loop safety and efficacy of medical devices prior to randomized controlled clinical trials. Model-based clinical trials for medical device software have the potential to complement the current regulatory approach by reducing the cost, scope and probability of failure of a traditional clinical trials. The area of Medical Cyber-Physical Systems is in its early days with several exciting and urgent fundamental challenges in modeling, control, verification and testing for higher confidence life-critical systems.

Acknowledgements

The authors would like to thank Houssam Abbas, Rajeev Alur, George M. Chen, Allison Connolly, Sanjay Dixit, Insup Lee, Pieter Mosterman, Miroslav Pajic, Oleg Sokolsky and Larisa G. Tereshchenko for fruitful discussions during the preparation of this manuscript. This research was supported in part by NSF CNS-0720518, NSF CNS-1035715, NSF MRI-0923518, NSF CPS Frontier 1446664 and NSF CAREER-1253842 grants. This work was also supported in part by STARnet - a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

References

R. Alur and D. L. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126:183–235, 1994.

B. P. Kovatchev and M. Breton and C. Dalla Man and C. Cobelli. In Silico Preclinical Trials: A Proof of Concept in Closed-Loop Control of Type 1 Diabetes. *Journal of Diabetes Science and Technology*, 3, 2009.

J. Beaumont, D. C. Michaels, M. Delmar, J. Davidenko, and J. Jalife. A Model Study of Changes in Excitability of Ventricular Muscle Cells. *American Journal of Physiology. 268*, 1995.

G. Behrmann, A. David, and K. G. Larsen. A Tutorial on UPPAAL. *Formal Methods for the Design of Real-Time Systems, Lecture Notes in Computer Science*, pages 200–236, 2004.

P. Bogdan, S. Jain, and R. Marculescu. Pacemaker control of heart rate variability: A cyber physical system perspective. *ACM Transactions on Embedded Computing Systems*, 12(1s):50:1–50:22, 2013.

Boston Scientific Corporation. PACEMAKER System Specification. Boston Scientific. *Device Documentation*, 2007a.

Boston Scientific Corporation. The Compass - Technical Guide to Boston Scientific Cardiac Rhythm Management Products. *Device Documentation*, 2007b.

E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counter Example-Guided Abstraction Refinement for Symbolic Model Checking. *Journal of the ACM*, 50 (5):752–794, 2003.

E. M. Clarke and E. A. Emerson. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs, Workshop*, pages 52–71, 1982.

E. M. Clarke, O. Grumberg, and D. E. Long. Model Checking and Abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

R J. Coffey. Deep brain stimulation devices: A brief technical history and review. *Artificial Organs*, 33(3):208–220, 2009.

D A. Vogel. *Medical Devices Software: Verification, Validation and Compliance*. Artech House, 2011.

E.W. Hsu and C.S. Henriquez. Myocardial fiber orientation mapping using reduced-encoding diffusion tensor imaging. *Journal of Cardiovascular Magnetic Resonance*, 3(4):339–347, 2011.

P. Feiler, L. Wrage, and J. Hansson. System architecture virtual integration: A case study. *Embedded Real-time Software and Systems Conference*, 2010.

U. S. Food and Drug Administration. Design Control Guidance For Medical Device Manufacturers. *Center for Devices and Radiological Health*, 1997.

U. S. Food and Drug Administration. General principles of software validation; final guidance for industry and fda staff. *Center for Devices and Radiological Health*, 2002.

U. S. Food and Drug Administration. Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices. *Center for Devices and Radiological Health*, 2005.

U. S. Food and Drug Administration. Ensuring the Safety of Marketed Medical Devices: CDRH's Medical Device Postmarket Safety Program. *Center for Devices and Radiological Health*, Jan 2006.

U. S. Food and Drug Administration. Medical device recall report - fy2003 to fy2012. *Center for Devices and Radiological Health*, 2012.

U. S. Food and Drug Administration. Classification of medical devices. *US FDA documents*, 2014.

B. Fuertes and J. Toquero. Pacemaker Lead Displacement: Mechanisms And Management. *Indian Pacing Electrophysiology Journal*, 2003.

S. Furman and J. D. Fisher. Endless loop tachycardia in an av universal (ddd) pacemaker. *Pacing and Clinical Electrophysiology*, 5(4):486–489, 1982.

S. Fürst, J. Mössinger, S. Bunzel, T. Weber, F. Kirschke-Biller, P. Heitkämper, G. Kinkelin, K. Nishikawa, and K. Lange. Autosar–a worldwide standard is on the road. In *14th International VDI Congress Electronic Systems for Vehicles*, volume 62, 2009.

M. Ghorbani and P. Bogdan. A cyber-physical system approach to artificial pancreas design. In *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, 2013.

R. Grosu, G. Batt, F. H. Fenton, J. Glimm, C. Le Guernic, S.A. Smolka, and E. Bartocci. From cardiac cells to genetic regulatory networks. In *Computer Aided Verification*, volume 6806 of *Lecture Notes in Computer Science*, pages 396–411. Springer Berlin Heidelberg, 2011.

Mathworks Inc. Matlab R2011a Stateflow Documentation. http://www.mathworks.com/help/toolbox/stateflow, 2016.

Md. A. Islam, A. Murthy, A. Girard, S. A. Smolka, and R. Grosu. Compositionality results for cardiac cell dynamics. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, (HSCC '14), pages 243–252, 2014.

R. Jetley, S. P. Iyer, and P. L. Jones. A Formal Methods Approach to Medical Device Review. *IEEE Computer*, 39:61–67, 2006.

Z. Jiang and R. Mangharam. Modeling Cardiac Pacemaker Malfunctions with the Virtual Heart Model. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 263 –266, Sept 2011.

Z. Jiang and R. Mangharam. Virtual Heart Model website - http://medcps.org, 2016.

Z. Jiang, M. Pajic, A. Connolly, S. Dixit, and R. Mangharam. Real-time heart model for implantable cardiac device validation and verification. In *2010 22nd Euromicro Conference on Real-Time Systems (ECRTS)*, pages 239 –248, July 2010.

Z. Jiang, M. Pajic, and R. Mangharam. Model-based Closed-loop Testing of Implantable Pacemakers. In *ACM/IEEE Second International Conference on Cyber-Physical Systems (ICCP'11)*, 2011.

Z. Jiang, M. Pajic, and R. Mangharam. Cyber-Physical Modeling of Implantable Cardiac Medical Devices. *Proceedings of the IEEE*, 100(1):122 –137, Jan. 2012a.

Z. Jiang, M. Pajic, S. Moarref, R. Alur, and R. Mangharam. Modeling and Verification of a Dual Chamber Implantable Pacemaker. *Tools and Algorithms for the Construction and Analysis of Systems*, 7214:188–203, 2012b.

Z. Jiang, M. Pajic, R. Alur, and R. Mangharam. Closed-loop verification of medical devices with model abstraction and refinement. *International Journal on Software Tools for Technology Transfer*, 16(2):191–213, 2014.

Z. Jiang, H. Abbas, PJ. Mosterman, and R. Mangharam. Tech Report: Abstraction-Tree For Closed-loop Model Checking of Medical Devices. *http://repository.upenn.edu/mlab_papers/73*, 2015.

M. E. Josephson. *Clinical Cardiac Electrophysiology*. Lippincot Williams and Wilkins, 2008.

K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, pages 134–152, 1997.

W. H. Maisel, M. O. Sweeney, W. G. Stevenson, K. E. Ellison, and L. M. Epstein. Recalls and Safety Alerts involving Pacemakers and Implantable Cardioverter-Defibrillator Generators. *JAMA*, 286(7), 2001.

A. Murthy, E. Bartocci, F. H. Fenton, J. Glimm, R. A. Gray, E. M. Cherry, S. A. Smolka, and R. Grosu. Curvature analysis of cardiac excitation wavefronts. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(2): 323–336, 2013.

Nano-RK. Nano-RK Sensor RTOS, Carnegie Mellon University. *http://nanork.org*, 2007.

M. Pajic, Z. Jiang, I. Lee, O. Sokolsky, and R. Mangharam. From Verification to Implementation: A Model Translation Tool and a Pacemaker Case Study. In *Proceedings of the 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium*, RTAS '12, pages 173–184, 2012.

M. Pajic, Z. Jiang, I. Lee, O. Sokolsky, and R. Mangharam. Safety-critical medical device development using the upp2sf model translation tool. *ACM Transactions on Embedded Computing Systems*, 13(4s):127:1–127:26, 2014.

C. S. Peskin and D. M. McQueen. A three-dimensional computational method for blood flow in the heart. 1. immersed elastic fibers in a viscous incompressible fluid. *Journal of Computer Physics*, 81(2):372–405, 1989.

S. Rossi, R. Ruiz-Baier, L. F. Pavarino, and A. Quarteroni. Active strain and activation models in cardiac electromechanics. *Proceedings in Applied Mathematics and Mechanics (PAMM)*, 11(1):119–120, 2011.

F. B. Sachse, A. P. Moreno, and J. A. Abildskov. Electrophysiological modeling of fibroblasts and their interaction with myocytes. *Annals of Biomedical Engineering*, 36(1):41–56, 2008.

K. Sandler, L. Ohrstrom, L. Moy, and R. McVay. Killed by Code: Software Transparency in Implantable Medical Devices. *Software Freedom Law Center*, 2010.

S. J. Saxonhouse, J. B. Conti, and A. B. Curis. Current of injury predicts adequate active lead fixation in permanent pacemaker/defibrillation leads. *Journal of the American college of Cardiology*, 2005.

R. Schulte, G. Sands, F. Sachse, O. Dossel, and A. Pullan. Creation of a Human Heart, Model and its Customisation using Ultrasound Images. *Biomedizinische Technik/Biomedical Engineering*, 46:26–28, 2001.

N. A. Trayanova and P. M. Boyle. Advances in modeling ventricular arrhythmias: from mechanisms to the clinic. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6(2):209–224, 2014.

U. S. Food and Drug Administration. Human Subject Protection; Acceptance of Data from Clinical Studies for Medical Devices; Proposed Rule. *Docket No. FDA-2013-N-0080*, 2013.

S. Yamane. Timed Weak Simulation Verification and its Application to Stepwise Refinement of Real Time Software. *International Journal of Computer Science and Network Security*, 6, 2006.

S. Zhang, C. Kriza, S. Schaller, and P. L. Kolominsky-Rabas. Recalls of cardiac implants in the last decade: what lessons can we learn? *PLoS ONE 10(5): e0125987.*, 2015. .