The Semantics of Refinement Chart

Dominique Méry[1] and Neeraj Kumar Singh[2]

[1] Université de Lorraine, LORIA, BP 239, Nancy, France
Dominique.Mery@loria.fr
[2] McMaster Centre for Software Certification, McMaster University, Hamilton, ON, Canada
singhn10@mcmaster.ca

Abstract. Refinement techniques play a major role to build a complex system incrementally. Refinement is supported by several modelling techniques in the area of system designing. These modelling techniques are either in textual notation or in graphical notation. This paper focuses on refinement chart (RC) that is based on graphical notations. The refinement chart is a graphical representation of a complex system using layering approach, where functional blocks are divided into multiple simpler blocks in a new refinement level, without changing the original behaviour of the system. The main contribution is to provide a formal semantical description of the refinement chart. The refinement chart offers a clear view of assistance in "system" integration that models complex critical medical systems. Moreover, it also sketches a clear view of different operating modes and their associated components. To realize the effectiveness of this approach, we apply this refinement based graphical modelling technique to model the grand challenge: *cardiac pacemaker*.

Keywords: Refinement, modelling, semantics, verification.

1 Introduction

Highly complex systems related to the medical domain are susceptible to error due to complex nature of the system operability and complexity of the system [1,2]. Software or hardware failure of a medical device can lead to injuries and loss of life. Design errors are considered as a main source of defects that can be introduced during the development process. There are several techniques available in the area of testing and formal verification to verify the correctness of system. However, existing techniques are not sufficient to prevent from the failure cases. Some additional techniques are required to handle the complexity of critical medical systems, and to make sure that the developed system is safe. The common notation of an existing technique is either textual or graphical. We know that a picture speaks louder than text, therefore graphical notations are much popular than textual notations. Nowadays graphical modelling techniques like Simulink, LabView and UML are main applications for developing the complex systems.

Despite all the efforts of the community, critical medical system designers still need a new way for modelling the systems, and to analyse the correctness of system behaviour. A set of requirements is given below that is mandatory for an efficient modelling solution:

– Integration of various modules of the medical critical systems and formal analysis of the inter-operations among the modules.
– Introspection features for identifying anomalies.
– Incremental model-based development.
– System integration of the critical infrastructure.
– Formalization of operating modes and associated components.
– Decomposition of a complex medical system into multiple subsystems.

This paper draws the attention towards a refinement-based graphical technique known as refinement chat for modelling the complex critical medical systems. This graphical technique provides an easily manageable representation for different refinement subsystems and offers a clear view of assistance in system integration. This graphical modelling technique may help for simplifying specification, synthesis, and validating the system requirements. Moreover, this technique enables an efficient creation/customization of the critical systems at low-cost and development time.

The main contribution of this paper is the semantical description of refinement chart [3,4], which provides automata of refinement chart including mode transitions, and operational semantics for a system refinement to carry out the sound and rigorous reasoning for developing the critical systems. Automata and operational semantics assist a designer to understand the mode transition and correct system behaviour of the different subsystems. The proposed approach of refinement chart captures all the above enumerated requirements. Moreover, this technique is not limited to medical systems only, but can be used to model highly critical systems like avionics, and automotive systems to identify the incorrect transitions or abnormal system behaviour.

This paper is organized as follows. Section 2 presents related work. Section 3 describes the refinement chart, and Section 4 presents the semantics of refinement chart. Section 5 presents assessment of the proposed technique using cardiac pacemaker case study. Finally, Section 6 concludes this paper along with future works.

2 Related Work

A *modal system* is a system characterized by *operation modes*, which coordinates system operations. Many systems are *modal systems*, for instance, space and avionic systems [5,1], steam boiler control [2], transportation and so on. Operation modes explore the actual system behaviour through observation of a system functioning under the multiple conditions according to the system environment. In this approach, a system is considered as a set of operating modes, where each operating mode is categorised according to the system functionality over different operating conditions. Each operating mode expresses the different functionality of the system.

Modecharts [6] is a graphical technique, which is used to handle mode and mode switching behavior of a system. This paper addresses the state space partition, and various working conditions of a system in order to define the controlling behaviour of the large state machines. However, the modecharts has lack of adequate support for specifying and reasoning about the functional properties.

Few papers [7,8] have also addressed the problem of mode changing in the real-time systems. Jorge et al. [7] present a survey on mode changing protocol for the real-time systems and propose several new protocols for schedulability analysis and configuration methods. The changing mode is based on delay and initiation of a new mode with consistently sharing the resources during mode change. Fohler et al. [8] discuss the issues of handling mode changes and requirements for their application for a real-time system, where they explore the specification of mode changes, sechedubility for modes and transitions, and run time execution of the mode changes including decomposition of the system into disjoint modes.

Dotti et al. [9] have proposed both formalization and a refinement notion for a *modal system*, using existing support for the construction of *modal system*. The paper presents the requirements for an Event-B model to realise a modal system specification using an industrial case study.

According to our literature survey, none of the existing approaches discuss a refinement-based technique for handling the complexity of a system. We propose a technique of refinement chart for presenting various operating modes for different subsystems. Each subsystem represents an independent function according to the operating modes. This refinement chart helps to design a complex system, system integration through code structuring, and to establish a relationship between two subsystems using operating modes.

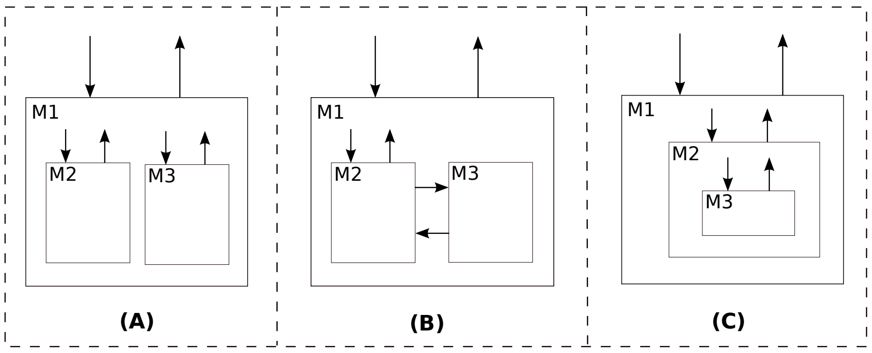
3 Refinement Chart

3.1 Motivation

The development of embedded software for a critical medical system requires significant lower level manual interaction for organizing and assembling the different parts of system. This is inherently error-prone, time-consuming and platform-dependent. To detect the failure cases in a software is not an easy task. Manually reviewing the source code is the only way to trace the cause of a failure. Due to the technological advancement and modern complexity of the critical medical system software, this is an impossible task for any third party investigator without any prior knowledge of the software. Consequently, we propose a simple methodology of system decomposition and integration using the refinement chart, that seeks to minimize the effort and overhead.

Refinement chart can be used during the decomposition and integration phases of a complex medical system that can help to analyse the complex behaviors. The purpose of refinement chart is to provide an easily manageable representation of a complex medical system in multiple refinements. The refinement chart offers a clear view of assistance in system integration. This is an important issue not only for being able to derive the system-level performance and correctness guarantees, but also for being able to assemble components in a cost-effective manner. Moreover, It can also help to a code designer to improve the code structures, code optimization, and code generation techniques. Every incremental refinement presents additional functionalities. This refinement-based structure may improve the safety, hardware integration and guidelines to develop the critical medical systems. This approach also helps in code integration and to test the different subsystems of a system independently.

A refinement-based system development has a different cost structure than the traditional development life-cycle. The cost of building models and related other required



$$\boxed{\begin{array}{ll} (A) & M1 \sqsubseteq (M2 \parallel M3) \\ (B) & M1 \sqsubseteq (M2 \succ M3) \\ (C) & (M1 \sqsubseteq M2,\ M2 \sqsubseteq M3) \end{array}}$$

Fig. 1. Refinement charts

design knowledge may be higher for producing the first system. However, these costs are amortized when reuse these models and designs for developing the other systems in future. Thus, the cost of producing first program may be higher, but the cost of development for reproducing advanced version of the products and reuse same codes in other products should be less than the conventional programming [10,11]. The cost of handling of proof obligations of specifications and refinements should be less than the cost of analyzing the final product.

3.2 Overview

Main objective of the refinement chart [3] is to specify the modal system requirements in a form that is easily and effectively implementable. During the modelling of modal system, several styles of specification are usually adopted for handling the complex operating modes. Functional blocks are divided into multiple simpler blocks in a new refinement level, without changing the original system behaviour. The final goal is to obtain a specification that is detailed enough to be effectively implemented, but also to describe correctly the requirements of a given system.

As the nature of critical medical systems is often characterizable as *modal systems*, we follow a state-based approach to propose suitable abstractions. We consider that the state of a model is detailed enough to allow one to distinguish its different operating conditions and also to characterize required mode functionality and possible mode switching in terms of state transitions.

Each subsystem that forms the specification is represented into a block diagram as a refinement chart. Fig. 1 presents the diagrams of the most abstract modal system. The diagrams use a visual notation loosely based on Statechart [12]. A mode is represented by a box with a mode name; a mode transition is an arrow connecting two modes.

The direction of an arrow indicates the previous and next modes in a transition. Refinement is expressed by nesting boxes. A refined diagram with an outgoing arrow from an abstract mode is equivalent to the outgoing arrows from each of the concrete modes. It is also similar to the ingoing arrows. In a refinement, nesting box can be arranged hierarchically and can be represented by basic rules of our refinement chart (see Fig. 2). The basic rules of refinements are: parallel refinement $[M1 \sqsubseteq (M2 \parallel M3 \parallel ...... \parallel M_{n-1} \parallel M_n)]$, sequential refinement $[M1 \sqsubseteq (M2 \succ M3 \succ ...... \succ M_{n-1} \succ M_n)]$ and nested refinement $[(M1 \sqsubseteq M2,\ M2 \sqsubseteq M3, ... , M_{n-1} \sqsubseteq M_n)]$. Furthermore, refinement charts, which appear in the hierarchical form can be expressed by any one rule, or several rules. A complex system can be represented by using refinement laws iteratively, means each subsystem can be refined by any rule (sequential, parallel or nesting) iteratively until to get the final model [3,4].

$$\boxed{\begin{array}{l} M1 \sqsubseteq (M2 \parallel M3 \parallel ...... \parallel M_{n-1} \parallel M_n) \\ M1 \sqsubseteq (M2 \succ M3 \succ ...... \succ M_{n-1} \succ M_n) \\ (M1 \sqsubseteq M2,\ M2 \sqsubseteq M3, ... , M_{n-1} \sqsubseteq M_n) \end{array}}$$

Fig. 2. Basic rules of Refinement Chart

Fig. 1 presents for only three modes (M1, M2 and M3) with different kinds of refinements. The parallel relationship among several refinement boxes states that a system operates simultaneously in the subsystems. For instance, Fig. 1(A) represents an abstract mode $M1$ and two parallel refinements are represented by nesting mode boxes $M2$ and $M3$. Transition between these two refinements $M2$ and $M3$ are not allowed. Entry into a parallel refined subsystem requires entry into all of its immediate child refinement. A transition out of one refinement requires an exit out of all the refined subsystems in parallel to it. The sequential relationship among several refinement boxes states that the system operates in at most one of these subsystems at any time. For example, Fig. 1(B) represents an abstract mode $M1$ and two sequential refinements are presented by the nesting mode boxes $M2$ and $M3$ in two levels of hierarchy, where $M2$ and $M3$ are embedded in $M1$. The transitions between $M2$ and $M3$ allows the system to go from one refinement to another refinement according to the operating modes. The nesting relationship among several refinement boxes states that the system operates in any subsystems. For example, Fig. 1(C) represents an abstract mode $M1$ and the subsystems refinement by a nesting box $M2$ and the subsystem $M2$ is refined by a nesting box $M3$ in three levels of hierarchy, where $M2$ is embedded in $M1$ and $M3$ is embedded in $M2$. A transition is allowed to next level of refined subsystem. A transition out of one refinement requires an exit out of all the refined sub level of refined subsystems.

3.3 Semantics of Refinement Chart (RC)

Automata RC. RC automata is similar to the classical automata, except that its modes can be of any RC type, and its transition function can refer to the concrete modes of automaton. Let $Automaton \triangleq (\Sigma, M, \delta, \theta, M_0)$ is a set of RC of type automaton. We have the following typing constraints on each elements of the automaton.

- Σ is a list of alphabets.
- M is a set of automaton modes.
- $\delta \subseteq \langle \mu, \sigma \rangle$ is an input transition relation, where:
  —μ denotes an input arrow.
  —σ an event of the transition.
- $\theta \subseteq \langle \nu, \sigma \rangle$ is an output transition relation, where:
  —ν denotes an output arrow.
  —σ an event of the transition.
- M_0 is an initial mode.

Operational Semantics. There are eight inference rules, written in the usual form $\frac{premiss}{conclusion}$. The first rule describes a transition between local modes using an input arrow. $\delta((m_1, m_2), \sigma)$ presents an input transition relation, where σ expresses an event of transition between two modes m_1 and m_2.

$$\frac{\delta((m_1,m_2),\sigma)}{m_1 \xrightarrow{\sigma} m_2} in$$

The second rule describes a transition between local modes using an output arrow. Similar to the input transition relation. $\theta((m_1, m_2), \sigma)$ presents an output transition relation, where σ expresses an event of transition between two modes m_1 and m_2.

$$\frac{\theta((m_1,m_2),\sigma)}{m_1 \xrightarrow{\sigma} m_2} out$$

Rule three provides the parallel operational semantics of an input transition relation, where parallel input transitions are presented by input arrows. $\delta((m_1, m_2, \cdots, m_n),$ $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1}))$ presents an input transition relation, where $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1})$ expresses a set of transition events, and $(m_1, m_2, \cdots, m_n)$ presents a set of modes of the refinement chart.

$$\frac{\delta((m_1,m_2,\cdots,m_n),(\sigma_1,\sigma_2,\cdots,\sigma_{n-1}))}{m_1 \xrightarrow{\sigma_1} m_2 || \cdots || m_1 \xrightarrow{\sigma_{n-1}} m_n} ||_{in}$$

Rule four shows the parallel operational semantics of an output transition relation, where parallel output transitions are presented by output arrows. $\theta((m_1, m_2, \cdots, m_n),$ $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1}))$ presents an output transition relation, where $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1})$ expresses a set of transition events, and $(m_1, m_2, \cdots, m_n)$ presents a set of modes of the refinement chart.

$$\frac{\theta((m_1,m_2,\cdots,m_n),(\sigma_1,\sigma_2,\cdots,\sigma_{n-1}))}{m_1 \xrightarrow{\sigma_1} m_2 || \cdots || m_1 \xrightarrow{\sigma_{n-1}} m_n} ||_{out}$$

Rule five shows the sequential operational semantics of an input transition relation, where sequential input transitions are presented by input arrows. $\delta((m_1, m_2, \cdots, m_n),$ $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1}))$ presents an input transition relation, where $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1})$ expresses a set of transition events, and $(m_1, m_2, \cdots, m_n)$ presents a set of modes of the refinement chart.

$$\frac{\delta((m_1,m_2,\cdots,m_n),(\sigma_1,\sigma_2,\cdots,\sigma_{n-1}))}{m_1\xrightarrow{\sigma_1}m_2\succ\cdots\succ m_{n-1}\xrightarrow{\sigma_{n-1}}m_n}\succ_{in}$$

Rule six shows the sequential operational semantics of an output transition relation, where sequential output transitions are presented by output arrows. $\delta((m_1, m_2, \cdots, m_n), (\sigma_1, \sigma_2, \cdots, \sigma_{n-1}))$ presents an output transition relation, where $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1})$ expresses a set of transition events, and $(m_1, m_2, \cdots, m_n)$ presents a set of modes of the refinement chart.

$$\frac{\theta((m_1,m_2,\cdots,m_n),(\sigma_1,\sigma_2,\cdots,\sigma_{n-1}))}{m_1\xrightarrow{\sigma_1}m_2\succ\cdots\succ m_{n-1}\xrightarrow{\sigma_{n-1}}m_n}\succ_{out}$$

Rule seven defines the nested operational semantics of an input transition relation, where nested input transitions are presented by input arrows. $\delta((m_1, m_2, \cdots, m_n), (\sigma_1, \sigma_2, \cdots, \sigma_{n-1}))$ presents an input transition relation, where $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1})$ expresses a set of transition events, and $(m_1, m_2, \cdots, m_n)$ presents a set of modes of the refinement chart.

$$\frac{\delta((m_1,m_2,\cdots,m_n),(\sigma_1,\sigma_2,\cdots,\sigma_{n-1}))}{m_1\xrightarrow{\sigma_1}m_2\sqsubseteq\cdots\sqsubseteq m_{n-1}\xrightarrow{\sigma_{n-1}}m_n}\sqsubseteq_{in}$$

Rule eight shows the nested operational semantics of an output transition relation, where nested output transitions are presented by output arrows. $\theta((m_1, m_2, \cdots, m_n), (\sigma_1, \sigma_2, \cdots, \sigma_{n-1}))$ presents an output transition relation, where $(\sigma_1, \sigma_2, \cdots, \sigma_{n-1})$ expresses a set of transition events, and $(m_1, m_2, \cdots, m_n)$ presents a set of modes in the refinement chart.

$$\frac{\theta((m_1,m_2,\cdots,m_n),(\sigma_1,\sigma_2,\cdots,\sigma_{n-1}))}{m_1\xrightarrow{\sigma_1}m_2\sqsubseteq\cdots\sqsubseteq m_{n-1}\xrightarrow{\sigma_{n-1}}m_n}\sqsubseteq_{out}$$

Kleene Closure RC. This operator is borrowed from regular expressions. It allows an arbitrary number of iterations (including zero) on RC. An iteration is completed when the component RC has reached at the concrete level or final mode. Formally, let $Closure \triangleq \langle\bigstar, m\rangle$ is a set of Kleene closure RC, where $m \in M$ is a mode of the closure. It is essentially used to determine if a closure can immediately exit without any iteration.

There are two inference rules related to the input and output arrows. $\bigstar_{in}$ allows for zero or infinite iterations using input arrows from an initial mode of the RC until to reach at the concrete mode or final mode. Similarly, $\bigstar_{out}$ allows zero or infinite iterations using output arrows from an initial mode to the concrete mode or final mode. However, Kleene closure RC allows any operation from the parallel, sequential and nested in any order.

$$\frac{\delta((m_1,m_2),\sigma)}{(\langle||,\succ,\sqsubseteq\rangle^{\bigstar},m_1)\xrightarrow{\sigma}(\langle||,\succ,\sqsubseteq\rangle^{\bigstar},m_2)}\bigstar_{in}$$

$$\frac{\theta((m_1,m_2),\sigma)}{(\langle||,\succ,\sqsubseteq\rangle^{\bigstar},m_1)\xrightarrow{\sigma}(\langle||,\succ,\sqsubseteq\rangle^{\bigstar},m_2)}\bigstar_{out}$$

4 Case Study : Cardiac Pacemaker

4.1 Informal Description of Cardiac Pacemaker

A pacemaker is an electronic device implanted in the body to regulate the abnormal heart rhythm. The primary functions of pacemaker are *pacing* and *sensing*. The pacemaker actuator is used to pace by delivering a short, intense electrical pulse into the heart. The pacemaker sensor senses an intrinsic activity of the heart. The pacing and sensing activities are synchronized with natural rhythm in both chambers right atria and ventricle. The basic elements of pacemaker are as follows:

1. **Leads:** One or more flexible metal wires, that transmit electrical signals between the heart and a pacemaker, and the same wires are also used to detect the intrinsic heart activities.
2. **The Pacemaker Generator:** It contains an implanted battery for power source and a controller as a brain of pacemaker for pacing and sensing activities.
3. **Device Controller-Monitor (DCM):** An external device that controls the functionalities of pacemaker remotely through wireless connection.
4. **Accelerometer (Rate Modulation Sensor):** An electromechanical device inside a pacemaker that measures an acceleration of a body in order to allow modulated pacing during various physical activities like running, sleeping, walking etc.

4.2 Bradycardia Operating Modes

Table 1 presents the generic code of a cardiac pacemaker. The codes are in a sequential order of letters that provides a description of pacemaker pacing and sensing functions. The first letter of the code indicates which chambers are being paced; the second letter indicates which chambers are being sensed; the third letter of the code indicates the response to sensing and the final letter, which is optional indicates the presence of rate modulation in response to the physical activity measured by the accelerometer.

Table 1. Generic code of cardiac pacemaker

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

4.3 Development of Cardiac Pacemaker Using Refinement Chart

This section presents the development of cardiac pacemaker using refinement chart. Each subsystem of the pacemaker is modelled incrementally to capture all the operating modes. Fig. 3 and Fig. 4 present the abstract models for one and two-electrode pacemaker (A), and the resulting models of three successive refinement steps (B to D). The diagrams use a visual notation to indicate the bradycardia operating modes of pacemaker under functional and parametric requirements. One or multiple operating modes are presented by a box; an operating mode transition is an arrow connecting two operating modes. The direction of an arrow indicates the previous and next operating modes in a transition. Refinement is expressed by nesting boxes to show an integration of new behaviour of the system.

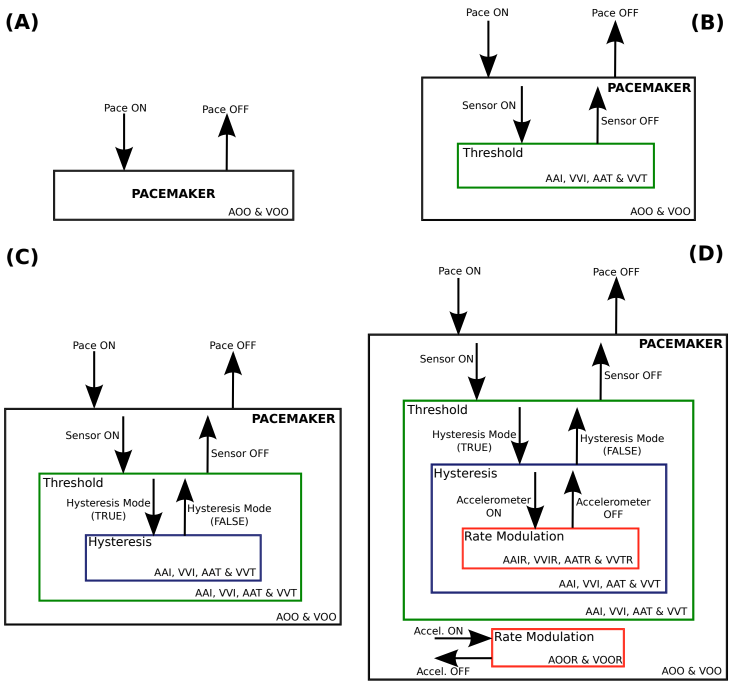


Fig. 3. Refinements of one-electrode pacemaker using the refinement chart

A refined diagram of an abstract mode is equivalent to a concrete mode. At the most abstract level, we introduce *pacing* activity into single and both heart chambers. In Fig. 3(A) and Fig. 4(A), *pacing* is presented by transitions *Pace ON* and *Pace OFF* for single chamber or both chambers. It is the basic transitions for all bradycardia operating modes.

In the next refinement (Fig. 3(B), Fig. 4(B)) step *pacing* is refined by *sensing*, corresponding to the activity of the heart, when sensing period is not under the refractory period (RF^1). In the first refinement of two-electrode pacemaker, sensors are introduced in both chambers. In Fig. 3(B) of one-electrode, sensing is represented by transitions *Sensor ON* and *Sensor OFF*, while in Fig. 4(B) of two-electrode, sensing is represented

[1] RF : Atria Refractory Period (ARP) or Ventricular Refractory Period (VRP).

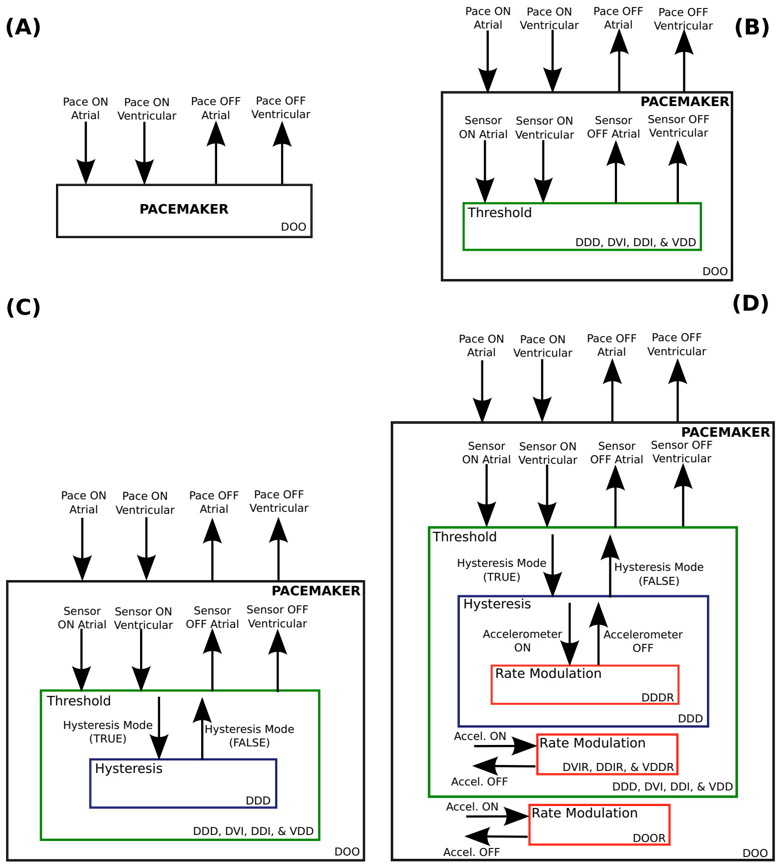


Fig. 4. Refinements of two-electrode pacemaker using the refinement chart

by transitions *Sensor ON Atria*, *Sensor ON Ventricle*, *Sensor OFF Atria* and *Sensor OFF ventricle*. The pacemaker's actuator and sensor are synchronizing to each other under the real-time constraints. The block diagrams (Fig. 3(B), Fig. 4(B)) represent the *threshold* refinement, that is a measuring unit which measures a stimulation threshold voltage value of the heart and a pulse generator for delivering stimulation pulses to the heart. The pacemaker's sensor starts sensing after the refractory period (RF) but pacemaker's actuator delivers a pacing stimulus according to the response of the sensor. Sensor-related transitions are available in all operating modes except AOO, VOO and DOO modes.

Third refinement step (Fig. 3(C), Fig. 4(C)) introduces different operating strategies under *hysteresis* interval: if the *hysteresis* mode is TRUE, then the pacemaker paces at a faster rate than the sensing rate to provide consistent pacing in one chamber (atrial or ventricle) or both chambers (atrial and ventricle), or prevents constant pacing in one chamber (atrial or ventricle) or both chambers (atrial and ventricle). In case of FALSE state of *hysteresis* mode, the pacemaker's sensor and actuator works in normal state and does not try to maintain the consistent pacing. Hysteresis mode is represented by transitions *Hysteresis Mode TRUE* and *Hysteresis Mode FALSE*. The main objective of hysteresis is to allow the patient to have his or her own underlying rhythm as much as possible.

According to the last refinement step (Fig. 3(D), Fig. 4(D)), it introduces the rate adapting pacing technique in the bradycardia operating modes of pacemaker. The rate modulation mode is represented by transitions *Accel. ON* and *Accel. OFF*. The rate modulation operating modes are available in all pacemaker operating modes which are given under multiple refinements. The pacemaker uses the accelerometer sensors to sense the physiologic need of the heart and increase or decrease the pacing rate. The amount of rate increases is determined by the pacemaker based on maximum exertion is performed by the patient. This increased pacing rate is sometimes referred to as the "sensor indicated rate". When exertion has stopped the pacemaker will progressively decrease the pacing rate down to the lower rate [3].

Refinement chart helps to model the system integration, which also complies with refinement based formal development. The block diagrams of the refinement chart help to build the complete system and used to handle the complexity of the whole system through decomposing in multiple independent parts. Here, refinement chart models different kinds of operating modes, and decompose the whole system based on operational modes. Decomposing using the refinement chart helps to analyse individual component and interaction or switching from one operating mode to other operating modes.

5 Conclusion

Today, in order to respect the certifiable assurance and safety, time to market and strict cost constraints, critical system designers need some new modelling and simulation solutions. The solutions must also permit software component modelling, component integration in a distributed environment, easier debugging of complex specifications, and mitigated connection with other, existing or new systems.

In this paper, we would like to stress the original contribution of our work through providing the automata and semantical operations of the refinement chart, where this technique can be used to model the desired system using layering approach in graphical block diagrams. The functional blocks are divided into multiple simpler blocks in a new refinement level, without changing the original behaviour of the system. Moreover, this technique offers decomposition, integration, and synchronization of the system components using incremental refinements. This approach helps in code integration and to test the different subsystems independently.

However, the refinement chart presents a block diagram for each subsystems and provides a structure in various refinements to build a complete system. The concrete refinement charts provide system integration information in the form of compose and decompose of software codes according to the blocks diagrams. Composition and decomposition help to improve the code structure and code optimization. To find a minimum set of events for each independent subsystem is known as code optimization. The refinement chart specially covers component-based design frameworks and decomposition, integration of critical infrastructure and device integration. The complexity of design is reduced by structuring systems using modes and by detailing this design using refinement.

System integration methodology using refinement charts are also used for system development, which helps a code designer to improve the code structure, code optimization, code generation for synthesizing, and synchronizing the software codes of a critical medical system like pacemaker. In the pacemaker case study, system has different kinds of functional requirements, and all the possible operating modes are decomposed in the refinement chart using multiple refinements related to the state flow models. Therefore use of the refinement chart, and formal specifications state the correctness of the system design and implementation. As a result, we have used manual development of the refinement chart in our pacemaker case study. In the future, we will develop an integrated development environment (IDE) for designing the medical critical systems using refinement chart and then automatic formalization from the developed models.

References

1. Miller, S.P.: Specifying the mode logic of a flight guidance system in CoRE and SCR. In: Proceedings of the Second Workshop on Formal Methods in Software Practice, FMSP 1998, pp. 44–53. ACM, New York (1998)
2. Abrial, J.-R., Börger, E., Langmaack, H. (eds.): Formal Methods for Industrial Applications, Specifying and Programming the Steam Boiler Control. LNCS, vol. 1165. Springer, Heidelberg (1996)
3. Singh, N.K.: Using Event-B for Critical Device Software Systems. Springer, Heidelberg (2013)
4. Méry, D., Singh, N.K.: Formal specification of medical systems by proof-based refinement. ACM Trans. Embed. Comput. Syst. 12(1), 15:1–15:25 (2013)
5. Butler, R.W.: An Introduction to Requirements Capture Using PVS: Specification of a Simple Autopilot. NASA Technical Memorandum 110255, NASA Langley Research Center, Hampton, VA (May 1996)
6. Jahanian, F., Mok, A.K.: Modechart: A specification language for real-time systems. IEEE Trans. Softw. Eng. 20(12), 933–947 (1994)
7. Real, J., Crespo, A.: Mode change protocols for real-time systems: A survey and a new proposal. Real-Time Syst. 26(2), 161–197 (2004)
8. Fohler, G.: Realizing changes of operational modes with a pre run-time scheduled hard real-time system. In: Proceedings of the Second International Workshop on Responsive Computer Systems, pp. 287–300. Springer, Heidelberg (1992)
9. Dotti, F.L., Iliasov, A., Ribeiro, L., Romanovsky, A.: Modal systems: Specification, refinement and realisation. In: Breitman, K., Cavalcanti, A. (eds.) ICFEM 2009. LNCS, vol. 5885, pp. 601–619. Springer, Heidelberg (2009)
10. Smith, D.R.: Generating programs plus proofs by refinement. In: Meyer, B., Woodcock, J. (eds.) Verified Software 2005. LNCS, vol. 4171, pp. 182–188. Springer, Heidelberg (2008)
11. Walters, H.: Hybrid implementations of algebraic specifications. In: Kirchner, H., Wechler, W. (eds.) ALP 1990. LNCS, vol. 463, pp. 40–54. Springer, Heidelberg (1990)
12. Harel, D.: Statecharts: A visual formalism for complex systems. Sci. Comput. Program. 8(3), 231–274 (1987)