Real-Time Animation for Formal Specification

Dominique Méry, Neeraj Kumar Singh

Real-Time Animation for Formal Specification

Dominique Méry and Neeraj Kumar Singh

Abstract A formal specification is a mathematical description of a given system. Writing a formal specification for real-life, industrial problems is a difficult and error prone task, even for experts in formal methods. It is crucial to get the approval and feedback when domain experts have a lack of knowledge of any specification language, to avoid the cost of changing a specification at later stage of development. This paper introduces a new functional architecture, together with a direct and efficient method of using real-time data set, in a formal model without generating the legacy source code in any target language. The implemented architecture consists of six main units. These units are: Data acquisition and preprocessing unit; Feature extraction unit; Database; Graphical animations dedicated tool: Macromedia Flash; Formal model animation tool Brama plug-in to interface between Flash animation and Event-B model; and formal specification system Event-B. These units are invoked independently and allow for simple algorithms to be executed concurrently. All the units of this proposed architecture help to animate the formal model with real-time data set and offer an easy way for specifiers to build a domain specific visualization that can be used by domain experts to check whether a formal specification corresponds to their expectations.

Dominique Méry
LORIA
Université Henri Poincaré Nancy 1
BP 239
54506 Vandœuvre-lès-Nancy
e-mail: mery@loria.fr

Neeraj Kumar Singh
LORIA
Université Henri Poincaré Nancy 1
BP 239
54506 Vandœuvre-lès-Nancy
e-mail: neerajkumar.singh@loria.fr

1 Introduction

Formal methods aim to improve software quality and produce zero-defect software, by controlling the whole software development process, from specifications to implementations. Formal methods are used by industries in a range of critical domains, involving higher safety integrity level certification and IEC 61508 [7] safety standard. IEC 61508 is intended to be a basic functional safety standard applicable to all kinds of industry.

In formal model development, they use top-down approaches and start from high-level and abstract specifications, by describing the fundamental properties of the final system. A detailed information about a given system is introduced in an incremental way [2]. The correctness between two levels is ensured by refinement proofs. The final refinement leads to the expected behaviour of the system implementation model.

The role of verification and validation is very important in the development of safety critical systems. Verification starts from the requirements analysis stage where design reviews and checklists are used for validation where functional testing and environmental modelling is done. The results of the verification and validation process are an important component in the safety case, which is used to support the certification process.

Event-B is a formal method for system-level modelling and analysis. There are two main proof activities in Event-B: *consistency checking*, which is used to show that the events of a machine preserve the invariant, and *refinement checking*, which is used to show that one machine is a valid simulation of another. *There are several ways to validate a specification: prototyping, structured walkthrough, transformation into a graphical language, animation, and others.* Each technique has a common goal, to validate a system according to the operational requirements. *Animation focuses on the observable behaviour of the system [19, 21]. The principle is to simulate an executable version of the requirements model and to visualize exact behaviours of the actual system. Animators use finite state machines to generate a simulation process which can be then observed with the help of UML diagrams, textual interfaces, or graphical animations [12, 21]. Animation can be used in the early stage of development during the elaboration of the specification: there is no need to wait until it is finished and get the generated code. As a relatively low cost activity, animation can be frequently used during the process to validate important refinement steps. It then provides us with a validation tool consistent with the refinement structure of the specification process* [21].

The final code generation process consists of two stages: final level formal specifications are translated into programs in a given programming language, and then these programs are compiled. Nevertheless all approaches which support formal development from specification to code must manage several constraining requirements, particularly in the domain of embedded software where specific properties on the code are expected [23]. Finally, it is not possible to use the real-time data in the early stage of formal development without compiling the source code in any target language. Based on our various research experience using formal tools in an

industrial requirements (*verification* and *validation*) and our desire to disseminate formal methods, we have imagined a new approach to present an animated model of specification using real-time data set, in the early stage of formal development.

It can help a specifier gain confidence that the model that is being specified, refined and implemented, does meet the domain requirements. This is achieved by the animation component of Brama [16] with Macromedia Flash tool, that allows to check the presence of desired functionality and to inspect the behaviour of a specification.

In this paper, we describe an approach to extend an animator tool which will be useful to use the real-time data set. Now, present time all the animation tools use a *toy data* set to test the model while we are proposing a *key idea* to use the real-time data set with the model without generating the source code in any target language (C, C++, VHDL etc.). In this work, we present an architecture which allows to easily develop visualizations for a given specification. Our architecture supports state-based animations, using simple pictures to represent a specific state of a Event-B specification, and transition-based animations consisting of picture sequences by using real-time data set. The animated model consists in Macromedia Flash components in picture sequences that is controled by the real-time data set and it presents an actual view of the system. Before moving on we should also mention that there are scientific and legal applications as well, where the formal model based animation can be used to simulate (or emulate) certain scenarios to glean more information or better understanding of the system requirements.

This paper is organized as follows. Section 2 briefly introduces an animator tool, Brama. Section 3 presents the functional architecture which enables the animation of a proved specification with real-time data set. The functional architecture is then illustrated in section 4 on a real case study, the cardiac pacemaker system. Section 5 concludes the paper with some lessons learned from this experience and some perspectives along with future works.

2 Overview of Brama

Brama [16] is an animator for Event-B specifications which is designed by Clearsy. Brama is an Eclipse plug-in suit and Macromedia Flash extension that can be used with Windows, Linux and MacOS for RODIN platform [13]. Brama can be used to create animations at different stages of development of a simulated system. To do so, a modeler may need to create an animation using the Macromedia Flash plug-in for Brama. The use of this plug-in is established through a communication between the animation and the simulation.

A modeler can represent the system manually within RODIN [13] or represent the system with the Macromedia Flash tool that allows for communication with the Brama animation engine through a communication server. Brama communicates with Macromedia Flash through a network connection. Brama acts as a server to which the animation will connect. In order to connect to a Brama server, a Flash

animation has to use the Brama component. This component handles the connection and the communication with the Brama server. This server has been created to communicate with animations which would stimulate the simulation and display an image of the system. The communication server exchanges the information packets between a model and a tool. Macromedia Flash controls the functional behaviour of all graphical components (button, check box, movie clip etc.). When the modeler and domain experts are satisfied with output, Brama can export the finished animation.

Brama contains the following main modules: B2Rodin: an animation engine (predicate solver), event and B variable visualization tools, an automatic event linkage management module, a variable management module, observed predicates and expressions, and a Macromedia Flash communication module. The Brama model animation tool provides some feedbacks that can be used by the modeler throughout the modeling process. The animation functions allow it to "create" various model events, filters and properties during testing process [16].

3 Description of the Architecture

Figure 1 depicts the overall functional architecture that can use the real-time data set to animate the Event-B model without generating source code of the model in any target language (C, C++, VHDL etc.). This architecture has six components: Data acquisition and preprocessing unit; Feature extraction unit; Database; Graphical animations dedicated tool: Macromedia Flash; Formal model animation tool Brama plug-in to interface between Flash animation and Event-B model; and formal specification system Event-B.

Data acquisition and preprocessing begin with the physical phenomenon or physical property to be measured. Examples of this include temperature, light intensity, heart activities and blood pressure [14] and so on. Data acquisition is the process of sampling of real world physical conditions and conversion of the resulting samples into digital numeric values. The data acquisition hardware can vary from environment to environment (i.e camera, sensor etc.). The components of data acquisition systems include sensors that convert physical properties. A sensor, which is a type of transducer, that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument.

Data preprocessing is a next step to perform on raw data to prepare it for another processing procedure. Data preprocessing transforms the data into a format that will be more easily and effectively processed for the purpose of the user. There are a number of different tools and methods used for preprocessing on different types of raw data, including: sampling, which selects a representative subset from a large population of data; transformation, which manipulates raw data to produce a single input; denoising, which removes noise from data; normalization, which organizes data for more efficient access.

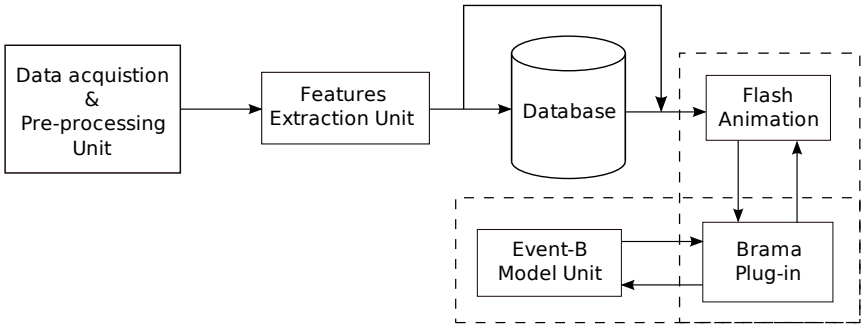

Fig. 1 A functional architecture to animate a formal specification using real time data set without generating source code

The features extraction unit is a set of algorithms that is used to extract the parameters or features from the collected data set. Theses parameters or features are numerical values that are used by animated model at the time of animation. The feature extraction relies on a thorough understanding of the entire system mechanics, the failure mechanisms, and their manifestation in the signatures. The accuracy of the system is fully dependent on the feature or parameter values being used. Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power or a classification algorithm which overfits the training sample and generalizes poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. Collecting measured data and processing these data to accurately determine model parameter values is an essential task for the complete characterization of a formal model.

The database unit is optional. It stores the feature or parameter values in the database file in any specific format. This database file of parameters or features can be used in future to execute the model. Sometimes, feature extraction algorithms take more time to calculate the parameters or the features. In such a situation, modeler can store the parameters or the features in database file to test the model in future. A modeler can also use the extracted parameters or features directly in the model, without using the database.

The animated graphics are designed in the Macromedia Flash tool [15]. Macromedia Flash, a popular authoring software developed by Macromedia, is used to create vector graphics-based animation programs with high graphic illustrations and simple interactivity. Here we use this tool to create the animated model of the physical environment and use the Brama plug-in to connect the Flash animation and the Event-B model. This tool also helps to connect the real-time data set to a formal model specification using some intermediate steps and finally makes the animated model closer to the domain expert expectations.

Brama is a tool allowing to animate Event-B models on the RODIN platform. It allows animating and inspecting a model using Flash animations. Brama has two objectives: to allow the formal models designer to ensure that his model is executed in accordance with the system it is supposed to represent; to provide this model with a graphic representation and animate this representation in accordance with the state of the formal model. The graphic representation must be in Macromedia Flash format and requires the use of a separate tool for its elaboration (Flash MX, for example). Once the Event-B model is satisfactory (it has been fully proven and its animation has demonstrated that the model behaves like its related system), you can create a graphic representation of this system and animate it synchronously with the underlying Event-B Rodin model. Brama does not create this animation. It is up to the modeler to create the representation of the model depending on the part of the model he wants to display. However, Brama provides the elements required to connect your Flash animation and Event-B model (in more detail see sec. 2) [16].

Event-B is a *proof-based* formal method [5, 2] for system-level modeling and analysis of large reactive and distributed systems. In order to model a system, Event-B represents in terms of *contexts* and *machines*. The set theory and first order logic are used to define contexts and machines of a given system. Contexts [5, 2] contain the static parts of a model. Each context may consist of carrier sets and constants as well as axioms which are used to describe the properties of those sets and constants. Machines [5, 2] contain the dynamic parts of an Event-B model. This part is used to provide behavioral properties of the model. A machine is made of a state, which is defined by means of variables, invariants, events and theorems. The use of refinement represents systems at different levels of abstraction and the use of mathematical proof verifies consistency between refinement levels. Event-B is provided with tool support in the form of an open and extensible Eclipse-based IDE called RODIN [13] which is a platform for Event-B specification and verification.

4 Applications and Case Studies

We have tested our proposed architecture on a case study performed on bradycardia operating modes of an artificial single electrode cardiac pacemaker [9]. A pacemaker is a high confidence medical device [1, 6, 20] that is implemented to provide proper heart rhythm when the body's natural pacemaker does not function properly. In the single electrode pacemaker, the electrode is attached to the right atrium or the right ventricle. It has several operational modes that regulate the heart functioning. The specification document [4] describes all possible operating modes that are controlled by the different programmable parameters of the pacemaker. All the programmable parameters are related to real-time and action-reaction constraints, that are used to regulate the heart rate.

In order to understand the "language" of pacing, it is necessary to comprehend the coding system that is produced by a combined working party of the North American Society of Pacing and Electrophysiology (NASPE) and the British Pacing and

Electrophysiology Group (BPEG) known as NASPE/BPEG generic (NBG) pacemaker code [8, 24, 18, 25]. This is a code of five letters of which the first three are most often used. The code provides a description of the pacemaker pacing and sensing functions. The sequence is referred to as "bradycardia operating modes"(see Table-1). In practice, only the first three or four-letter positions are commonly used to describe bradycardia pacing 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. Accelerometer is an additional sensor in the pacemaker system that detects a physiological result of exercise or emotion and increases the pacemaker rate on the basis of a programmable algorithms. "X" is a wildcard used to denote any letter (i.e. "O", "A", "V" or "D"). *Triggered* (T) refers to deliver a pacing stimulus and *Inhibited* (I) refers to an inhibition from further pacing after sensing of an intrinsic activity from the heart chamber.

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

Table-1 Bradycardia operating modes of a pacemaker system

An Event-B specification of the model has been written [4, 9] as an effort to make it amenable to the formal techniques required by high confidence medical device certification [1, 20, 6]. The formal specification of a cardiac pacemaker system consists of five machines; one abstract and four refinements.

This study represents a formal specification and a systematic block diagram (see Fig. 2) of hierarchical tree structure of the bradycardia operating modes of the single electrode cardiac pacemaker. The hierarchical tree structure shows the stepwise refinement from abstract to concrete model. Each level of refinement introduces the new features of pacemaker as functional and parametric requirements. The root of this tree indicates the single electrode cardiac pacemaker. The next two branches of tree show the two chambers; atrium and ventricular. These atrium and ventricular are the right atrium and ventricular. The atrium chamber uses the three operating modes; AOO, AAI and AAT (see Table-1). Similarly, the ventricular chamber uses the three operating modes; VOO, VVI and VVT (see Table-1). It is an abstract level of the model. The abstract model presents all the operating modes abstractly with required properties of the pacemaker. From the first refinement to the last refinement, there is only one branch in every operating modes of the atrium and ventricular chambers. The subsequent refinement models introduce all detailed informations for the resulting system. Every refinement level shows an extension of previous operating

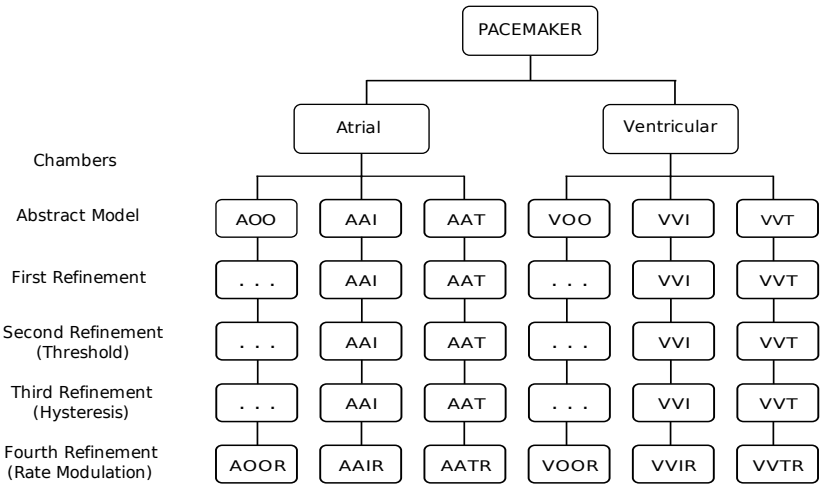

Fig. 2 Refinement structure of bradycardia operating modes of the single electrode cardiac pacemaker

modes as an introduction of a new feature or functional requirement. The triple dots (...) represents that there is no refinement at that level in particular operating modes (AOO and VOO). From abstract level to third refinement level, there are similar operating modes. But the fourth refinement level has represented by additional rate adaptive operating modes(i.e AOOR, AAIR, VVTR etc). These operating modes are different from the previous levels operating modes. These refinement structure is very helpful to model the functional requirements of the single electrode cardiac pacemaker. The following outline is given about every refinement level to understand the basic notion of a formal model of the cardiac pacemaker system:-

- **Abstract Model :**In the context of an abstract model of the single electrode pacemaker contains the definitions and properties of different time interval parameters (upper rate limit (URL), lower rate limit (LRL), refractory period (RF)...etc.), and the pacemaker's actuator and sensor status (ON and OFF). The first abstract model has been specified by the pacing, sensing and timing components with the help of action-reaction and real-time pattern using some initial events ($Pace\_ON$, $Pace\_OFF$, and tic). In an abstract of AAI and VVI modes, two new extra events ($Pace\_OFF\_with\_Sensor$ and $Sense\_ON$) has been introduced. Similarly in AAT and VVT modes, two new extra events ($Pace\_ON\_with\_Sensor$ and $Sense\_ON$) has been introduced. Remaining other rate adaptive bradycardia operating modes (AOOR, VOOR, AAIR, AATR,VVIR and VVTR) of the single electrode pacemaker are refinement of basic bradycardia operating modes, which are described in stepwise refinements. The basic abstract model of a pacemaker represents the action-reaction and real-time pattern for describing pacing and sensing modes of the single electrode cardiac pacemaker.

- **Refinement 1 :** In this refinement of the single electrode cardiac pacemaker model, we have introduced more invariants to satisfy the pacing and sensing requirements of the system under real time constraints.
- **Refinement 2 :** This refinement is relatively more complex then the last refinement. In this refinement, we have introduced the *threshold* parameter is used to filter the exact sensing value within a sensing period to control the sensing and pacing events. A pacemaker has a stimulation threshold measuring unit which measures a stimulation threshold voltage value of a heart and a pulse generator for delivering stimulation pulses to the heart. The pulse generator is controlled by a control unit to deliver the stimulation pulses with respective amplitudes related to the measured threshold value and a safety margin.
- **Refinement 3 :** In this refinement, we have introduced the application of *hysteresis* interval to provide consistent pacing or to prevent constant pacing in the heart chambers (atrial or ventricle).
- **Refinement 4 :** In the last and final refinement, we have introduced the *accelerometer sensor* component and *rate modulation* function to obtain a new rate adaptive operating modes of the pacemaker. Rate adaptive term is used to describe the capacity of a pacing system to respond to physiologic need by increasing and decreasing pacing rate. The rate adaptive mode of the pacemaker can progressively pace faster than the lower rate, but no more than the upper sensor rate limit, when it determines that heart rate needs to increase. This typically occurs with exercise in patients that cannot increase their own heart rate. The amount of rate increase is determined by the pacemaker on the basis of 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 paced rate down to the lower rate. The final refinement represents the rate modulation function with new operating modes (AOOR, VOOR, AAIR, VVIR, AATR and VVTR) of the pacemaker system.

To find the complete formal development of the single electrode cardiac pacemaker see the research report [9].

We have mainly used this case study to experiment on our proposed architecture which enables the animation of a proved specification with real-time data set without generating the legacy source code in any target language. According to the proposed architecture (see Figure 1) for this experiment, we have not used any data acquisition device to collect the ECG (electrocardiogram) signal [26, 17, 18]. We have done this experiment in off-line mode, meant we have used our architecture to test the real-time data set of ECG signal that is already collected. ECG signal collection and features extraction in on-line mode is too expensive due to complex data acquisition process and limitation of feature extracting algorithms. So, we have used the ECG signal and feature extraction algorithms for our experiment from the MIT-BIH Database Distribution [11].

We have downloaded the ECG signal from ECG data bank [11]. ECG signals are freely available for academic experiments. We have applied some algorithms to extract the features (P, QRS, PR, etc.) from ECG signal and stored it into a database.

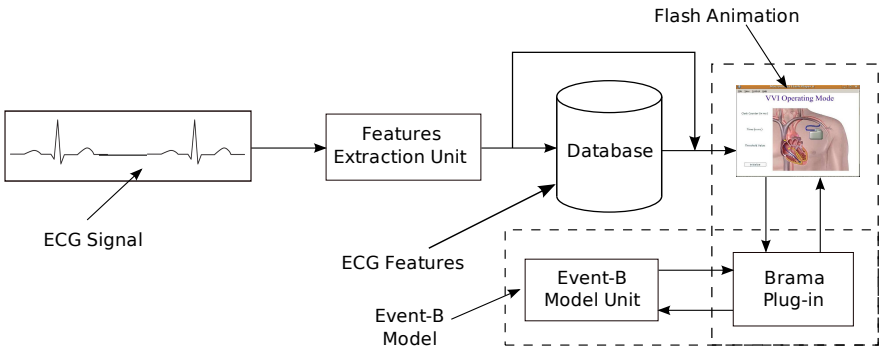

Fig. 3 Implementation of proposed functional architecture on the single electrode cardiac pacemaker case study

We have written down some Macromedia Flash scripts to interface between Flash tool and Brama component, to pass the real data set as a parameters from database to the Event-B model. No any tool is available to interface between database and the Event-B model. Extra Macromedia Flash script coding and the Brama animation tool help to test the Event-B formal model of cardiac pacemaker on real-time data set. We have designed an animated graphics of heart and pacemaker in Macromedia Flash, where this animated model represents the pacing activity in right ventricular chamber. This animated model simulates the behaviour of heart according the bradycardia operating modes (VVT, VVI) and animates the graphic model. The animation of the model is fully based on the Event-B model. Event-B model is executing all events according to the parametric value. These parametric values are the extracted features from the ECG signal, which are passing into the Event-B model.

Figure 3 represents an implementation of proposed architecture on the formal model of a cardiac pacemaker case study. According to the architecture, data acquisition unit collect the ECG signal and features extraction are done by the feature extraction or parameter estimation unit. The extracting features are stored in database as the XML file format. Macromedia Flash tool helps to design the animated graphics of the heart and pacemaker. In next unit, Brama plug-in helps to communicate between animated graphics and Event-B formal model of the single electrode cardiac pacemaker. Finally, we have tested a real-time data set in the formal models without generating the source code with the help of Brama existing animation tool.

5 Conclusion and Future work

The objective of this proposed architecture is to validate the formal model with real-time data set in the early stage of development without generating the legacy source code in any target language. In this paper, we focused the attention on the techniques introduced in the architecture for using the real-time data set to achieve the adapt-

ability and confidence on formal model. Moreover, this architecture should guarantee that the formal model is correct with respect to the high level specifications and it is runtime error free. At last, this proposed architecture should be adaptable to various target platforms and formal models techniques (Event-B, Z, Alloy, TLA$^+$ etc.).

With respect to adaptability of new architecture, two techniques were considered useful, implemented and tested on the case study. The proposed architecture results are satisfactory and demonstrate the ability to validate the formal model of a single electrode cardiac pacemaker system with real-time data set. The certification of a software item is concerned both by verification and validation activities. Ideally, the former should be fully formal, relying on proofs and formal analysis. The latter deals with an inherently informal element: the requirements [21]. The gains rely then on the guarantees provided by the use of a formal method and on the certification level which can be obtained by this way. The technique discussed here aims at improving the confidence in the software at earlier stages of development cycle. As far as we know, no any animation tool support to validate the formal model on real-time data set, which can be closed to the source code. The adaptation of this architecture needs more complete experiments, specially for their impact on the data acquisition and features extraction time for some specific domains.

A alternative approach is developed: rather than generating a source code of formal model in advance that the proposed architecture always produces a desired result which is similar to the correctly implemented source code. A key feature of this validation as it is full automation and animation of specification in the early stage of formal development. The case study has shown that requirement specifications could be used directly in real-time environment without modifications for automatic test result evaluation using our approach. Moreover, there are scientific and legal applications as well, where the formal model based animation can be used to simulate (or emulate) certain scenarios to glean more information or better understanding of the system and assist to improve the final given system.

While arguing about the relationship between refinement based modeling and its stepwise validation, we discovered that not every refinement step is animatable. This is consistent with using animation as a kind of quality-assurance activity during development. We believe that one animation per abstraction level is sufficient. In fact, the first refinement of a level may often have a non-determinism too wide to allow for meaningful animation (concept introduction), but subsequent refinements get the definitions of the new concept precise enough to allow animation [22].

The proposed architecture is not complete yet due to certain limitations; acquisition devices, features extraction algorithms and so on. In future this is expected to use same architecture in multi domain, the proposed architecture to use the real time data set to validate the any formal model specification in the early stage of development. Manual application of this architecture to apply real-time data set in the formal model is tedious, cumbersome and may be error prone if not applied carefully. Therefore we are planning to write an application programming interface (API) which can interface automatically from any acquisition device or database to formal model using Flash animation and Brama component.

Acknowledgements Work of Dominique Méry and Neeraj Kumar Singh is supported by grant No. ANR-06-SETI-015-03 awarded by the Agence Nationale de la Recherche. Neeraj Kumar Singh is supported by grant awarded by the Ministry of University and Research.

We are deeply grateful to our colleague Dominique Cansell, Jean-Pierre Jacquot, Atif Mashkoor, Joris Rehm and Nazim Benaissa, who provided us expertise and information for shaping our ideas.

References

1. A Reseach and Development Needs Report by NITRD. High-Confidence Medical Devices : Cyber-Physical Systems for 21st Century Health Care. http://www.nitrd.gov/About/MedDevice-FINAL1-web.pdf.
2. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. 2010. Forthcoming book.
3. Dines Bjørner and Martin C. Henson, editors. EATCS Textbook in Computer Science. Springer, 2007.
4. Boston Scientific Boston Scientific: Pacemaker system specification, Technical report. 2007.
5. Dominique Cansell and Dominique Méry. *Logics of Specification Languages*, pages 33–140. Springer, 2007. See [3].
6. C.A.R. Hoare, Jayadev Misra, Gary T. Leavens, and Natarajan Shankar. The verified software initiative: A manifesto. *ACM Comput. Surv.*, 41(4):1–8, 2009.
7. IEC, IEC functional safety and IEC 61508: . Working draft on functional safety of electrical/electronic/programmable electronic safety-related systems (2005).
8. Writing Committee Members, Andrew E. Epstein, John P. DiMarco, Kenneth A. Ellenbogen, III Estes, N.A. Mark, Roger A. Freedman, Leonard S. Gettes, A. Marc Gillinov, Gabriel Gregoratos, Stephen C. Hammill, David L. Hayes, Mark A. Hlatky, L. Kristin Newby, Richard L. Page, Mark H. Schoenfeld, Michael J. Silka, Lynne Warner Stevenson, and Michael O. Sweeney. ACC/AHA/HRS 2008 Guidelines for Device-Based Therapy of Cardiac Rhythm Abnormalities: Executive Summary: A Report of the American College of Cardiology/American Heart Association Task Force on Practice Guidelines (Writing Committee to Revise the ACC/AHA/NASPE 2002 Guideline Update for Implantation of Cardiac Pacemakers and Antiarrhythmia Devices): Developed in Collaboration With the American Association for Thoracic Surgery and Society of Thoracic Surgeons. *Circulation*, 117(21):2820–2840, 2008.
9. Dominique Méry and Neeraj Kumar Singh. Pacemaker's Functional Behaviors in Event-B. Research Report (http://hal.inria.fr/inria-00419973/en/), 2009.
10. Dominique Méry and Neeraj Kumar Singh. Technical Report on Formal Development of Two-Electrode Cardiac Pacing System. Research Report (http://hal.inria.fr/inria-00465061/PDF/Report_2electrode.pdf),2010.
11. MIT-BIH Database Distribution and Software : . http://ecg.mit.edu/index.html.
12. C. Ponsard, P. Massonet, A. Rifaut, J.F. Molderez, A. van Lamsweerde, and H. Tran Van. Early verification and validation of mission critical systems. *Electronic Notes in Theoretical Computer Science*, 133:237 – 254, 2005. Proceedings of the Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2004).
13. Project RODIN. Rigorous open development environment for complex systems. http://rodin-b-sharp.sourceforge.net/, 2004. 2004–2007.
14. M. A. Quiones, F. Tornes, Y. Fayad, R. Zayas, J. Castro, A. Barbetta, and F. Di Gregorio. *Rate-Responsive Pacing Controlled by the TVI Sensor in the Treatment of Sick Sinus Syndrome*. Springer, 2006.
15. Robert Reinhardt and Snow Dowd. *Adobe Flash CS3 professional bible*, page 1232. Wiley, 2007. ISBN-9780470119372.
16. Thierry Servat. *BRAMA: A New Graphic Animation Tool for B Models*, pages 274–276. LNCS. Springer, 2006.

17. Kenneth A. Ellenbogen and Mark A. Wood. Cardiac Pacing and ICDs. *4th Edition, Blackwell*, 2005, ISBN-10 1-4051-0447-3.
18. Aaron Hesselson. Simplified Interpretations of Pacemaker ECGs. *Blackwell Publishers*, 2003, ISBN 978-1-4051-0372-5.
19. Hung Tran Van, Axel van Lamsweerde, Philippe Massonet, and Christophe Ponsard. Goal-oriented requirements animation. *Requirements Engineering, IEEE International Conference on*, 0:218–228, 2004.
20. Jim Woodcock and Richard Banach. The verification grand challenge. *J. UCS*, 13(5):661–668, 2007.
21. Atif Mashkoor, Jean-Pierre Jacquot and Jeanine Souquires. Transformation Heuristics for Formal Requirements Validation by Animation. *2nd International Workshop on the Certification of Safety-Critical Software Controlled System*, 2009.
22. Atif Mashkoor Formal Domain Modeling: From Specification to Validation. *16th International Symposium on Formal Methods - FM 2009 (Doctoral Symposium)*, 2009.
23. Didier Bert and Sylvain Boulmé and Marie-Laure Potet and Antoine Requet and Laurent Voisin Adaptable Translator of B Specifications to Embedded C Programs. *FME 2003, Springer*, 94–113, 2003.
24. S. Serge Barold and Roland X. Stroobandt and Alfons F. Sinnaeve. Cardiac Pacemakers Step by Step. *Futura Publishing*, 2004, ISBN 1-4051-1647-1.
25. Charles J. Love. Cardiac Pacemakers and Defibrillators. *Landes Bioscience Publishers*, 2006, ISBN 1-57059-691-3.
26. Jaakko Malmivuo. Bioelectromagnetism. *Oxford University Press*, 1995, ISBN 0-19-505823-2.