Behavior Verification of Hybrid Real-time Requirements by Qualitative Formalism

Jang-Soo Lee[†], Sung-Deok Cha[‡]
[†]MMIS Lab., Korea Atomic Energy Research Institute
P.O.Box 105, Yusong-ku,Taejeon, 305-600, Korea
jslee@nanum.kaeri.re.kr
[‡]Computer Science Dept., Korea Advanced Institute of Science and Technology
373-1, Kusong-dong, Yusong-ku,Taejeon, 305-701, Korea
cha@salmosa.kaist.ac.kr

Abstract

Although modern control theories have been successfully applied to solve a variety of problems, they are often mathematically and physically too specific to describe and analyze the qualitative properties of hybrid real-time systems. In this paper, we propose the use of qualitative formal methods, Compositional Modeling Language (CML) and Causal Functional Representation Language (CFRL) in particular, to specify continuous plant dynamics and the required behavior respectively. The system behavior has been simulated by a qualitative simulator known as the Device Modeling Environment (DME), and verified against the required behavior. Using the Electrical Power System (EPS) as an example, we demonstrate the effectiveness of our approach by illustrating how a simple SCR-style specification can be transformed and analyzed.

1. Introduction

It is well-known in the software engineering community that a significant portion of software failures found in operational software can be traced to errors made during the requirements engineering phase and that it is expensive to correct such errors. Hybrid real-time and embedded software development is an area where software requirements must be subject to highly rigorous and systematic analysis. Examples include software used to control nuclear power plants, commercial and military jets, satellites, or manufacturing plants. A typical closed-loop process-control system consist of the following components (Figure 1): plant, controller, actuators, and sensors.

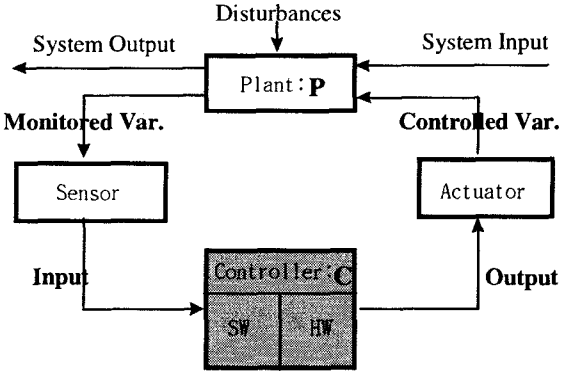

Figure 1. The control loop

When developing embedded software, one must have insight into the properties of the controlled physical processes and ensure that the behavior of the controller(C) satisfies the required system behavior(S_p) when viewed from the perspective of the plant(P) [10]. That is, the truth of the following proposition must be demonstrated:

$$P \cap C \Rightarrow S_p \qquad (1)$$

The plant P is usually a non-linear continuous system with disturbances and modeled by differential equations. The software-based controller(C) is discrete and prescriptive, in that it represents a plan for a course of action to be taken, and so usually modeled by a state machine. Though modern control theories have been

0-8186-8073-3/97 $10.00 © 1997 IEEE

useful in tackling a variety of real-world problems, they are often mathematically and physically too specific to model the qualitative system properties, such as the discrete description of the digital controller(C) and the abstract description of the required system behavior(S_D) in the requirements phase, since the focus of control theory has been on continuous variable systems modeled by differential and difference equations.

Requirements engineering is not a process of simple derivation of a controller specification(C) from a given set of requirements(S_D) and plant(P) properties. Rather, the three models, P, C, and S_P, are developed until an acceptable combination is reached, with the formal justification between all three being shown to hold. A consequence of this exploratory, iterative aspect of the requirements engineering process is that the three models may be progressively developed and modified, leading to a need for the formal methods to be not only rigorous, but also amenable to proof and re-proof with a minimum of effort. In response to the need for developing the requirements specification and verification methods for the hybrid real-time process-control systems, there have been a number of different formal methods, such as hybrid automata [4], temporal logic [13], process algebra [5], and so on. We believe most of these formalisms are too rigid to justify the proposition (1) in the early phase of the iterative requirements engineering process.

This paper concentrates on the qualitative formal method to specify the continuous plant and the required system behavior in terms of languages, based on the qualitative physics, CML and CFRL. The behavior of the hybrid real-time system has been simulated by a qualitative simulator, DME, and then verified as to whether it satisfies the required system behavior. The qualitative reasoning approach is selected to help the designers specify the desired system properties concisely without having to describe unnecessary details. Furthermore, qualitative simulation techniques allow for behavioral analysis of such abstract specification. In order to demonstrate the effectiveness of using a qualitative formalism in specifying and analyzing requirements of embedded real-time software, we use an electrical power system (EPS) to illustrate how SCR-style specification can be converted transformed and analyzed. We have chosen the SCR-style specification as an example because the SCR-style specifications and the four-variable method have been used on several industrial projects including the Darlington nuclear power plant shutdown system (SDS) , a similar SDS system in the Wolsong plant, and others[11].

The rest of our paper is organized as follows: Section 2 briefly describes the Four-variable approach and explains the limitations encountered when attempting to analyze software requirements specification. Section 3 provides a brief introduction to the qualitative formalism we adapted, explains how to transform a Four-variable model into a qualitative model, and how to conduct qualitative behavioral simulation of the system. Section 4 concludes the paper.

2. Four-variable model

The Four-variable model, illustrated in Figure 2, represents requirements of the system in terms of the mathematical relations of the four sets of variables called monitored(M), controlled(C), input(I), and output(O). While a monitored variable represents an environmental quantity that influences system behavior, a controlled variable describes an environmental quantity the system controls.

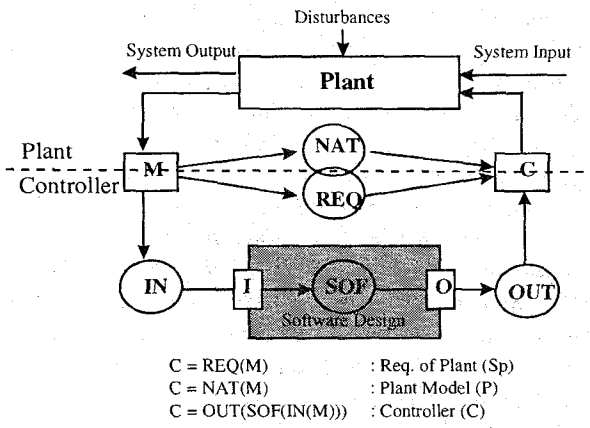

Figure 2. Four-variable model

A black box specification of required behavior is defined as two relations, REQ and NAT, from the monitored quantities to the controlled quantities. While NAT describes any constraints on behavior, such as those imposed by physical laws, REQ defines the additional constraints imposed by the system to be built. NAT relation is important because it explicitly captures the limits of the required behavior. REQ describes the required system behavior by defining the relation that the system must maintain between the monitored and the controlled quantities. There are also three relations, IN, OUT, and SOF, which are related to the controller [12]. To meet proposition (1), five relations, NAT, REQ, IN,

OUT, SOF, are needed to satisfy the following proposition:

$$NAT(M) \cap OUT(SOF(IN(M))) \Rightarrow REQ(M) \quad (2)$$

This proposition is compatible with the software acceptability condition in [12]. When the Four-variable model was used to specify and verify the software requirements specification (SRS) and the software design description (SDD) of the shutdown system of Wolsong nuclear power plant, the condition, OUT(SOF(IN(M))) => REQ(M), was demonstrated to verify the correctness of SDD against SRS. The correctness of the SRS itself against system specification, the satisfaction of proposition (2), however, was informally reviewed by the system and software engineers.

3. Qualitative formal models

In requirements engineering, many problems remain unsolved, such as the completeness of software requirements, the communication problems between the software engineer and the system engineer, the complexity problem, the model mismatch, and so on. We draw on work in the area of qualitative physics-qualitative formal models, and its associated reasoning methods-in order to alleviate the above problems and support the justification of proposition (2) in the requirements engineering process. We have used CML[3] to model the plant instead of the NAT relation, and CFRL[9] to model the requirements instead of the REQ relation.

Qualitative physics is an area of artificial intelligence concerned with reasoning qualitatively about the behavior of physical systems. An engineer who is designing an embedded computer system must have insight in understanding completely the properties of the controlled physical process to transform the system requirements into software requirements. By using the qualitative formal models to transform the requirements between them, the ability of engineers to specify and verify the requirements can be enhanced. One of the most important aspects of the requirements engineer's thinking method is the causal reasoning about the given plant. By reasoning from the knowledge of the structure and physical principles underlying the functions of the plant, the qualitative causal model written in CFRL can provide an intuitive and causal explanation on how the behavior is achieved. All of these properties of the qualitative formal models come from the ability to predict the behavioral aspect of the real-time system at an abstract level of qualitative characteristics of its behavior without unnecessary numerical details. We have conducted a qualitative simulation on the models represented by CML and CFRL, using the qualitative simulator DME[6], in order to verify the behavior of the hybrid system and to support the justification of the proposition (2). Qualitative simulation is different from numerical simulation of the physical system behavior in that behavior reasoning is carried out, not by collecting the numerical values of variables at different time points, but by employing a more abstract method involving the qualitative characteristics of its behavior [1].

3.1 An example: Electrical Power System (EPS)

EPS, an example adapted from Iwasaki's work [9], is a hybrid real-time system embedded in an earth orbiting satellite. A simplified schematic diagram of the EPS is shown in Figure 3. The main purpose of the EPS is to supply a constant source of electricity to the other subsystems of the satellite. The charge of the battery is maintained by a solar array. When the charge level of the battery exceeds a threshold, the charge-current controller opens the relay, allowing the battery to provide the power to the load. When the charge level drops below another threshold, the charge-current controller closes the relay, allowing the solar array to recharge the battery. Many of these kinds of the electro-mechanical devices exhibit both continuous and discrete behavior. Modeling such hybrid systems presents special challenges for specification and verification [4].

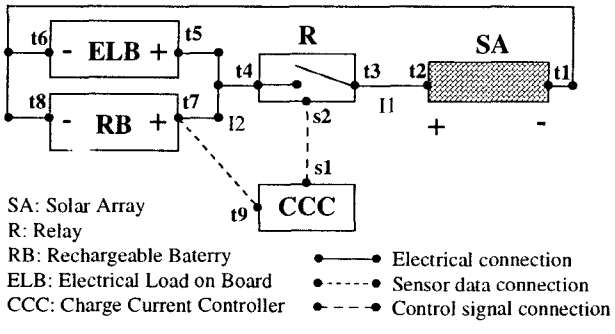

Figure 3. An Electrical Power System

3.2 Transformation of the plant model

When the NAT relation is used to model the physical behaviors of the plant, only certain quantities in the

environment are relevant. The external behavior of the system is specified by describing the relationships which must be maintained between the quantities it monitors and those it controls. For example, the NAT relation for the EPS can be described as follows using only one monitored variable (Rbvoltage).

1. Monitored Variables

Name	Type/Values	Physical Interpretation
RBvoltage	volts 31.0...33.8 volts	Since the battery can be damaged when it is charged beyond its capacity, the CCC opens the relay when the voltage exceeds a threshold to prevent the battery from being over-charged

2. NAT Relation

RBvoltage 6 ampere-hours $\leq$ d(RBvoltage(t))/dt $\leq$ 30 ampere-hours

31.0 volts $\leq$ RBvoltage $\leq$ 33.8 volts

Figure 4. Monitored variable and NAT relation

The NAT relation makes explicit the environmental constraints that are implicit in most specifications. However, when expressing the NAT relation using differential equations and time functions for the physical situation of the plant, the expressed model will bring about a complexity problem in the requirements engineering process, which can make it hard to justify the proposition (2). In addition, it is hard to simulate and verify formally whether the requirement model(S_D) follows from the conjunctive behavior of the controller model(C) with the plant model(P), because the NAT relation does not describe the plant model(P) but rather the external constraints by the plant.

On the other hand, a qualitative modeling approach can capture the behavioral aspects of the plant itself. The external plant behavior modeled as C=NAT(M) in Figure 2 can be expressed using the qualitative physics model based on the structure and the behavior of the plant because both models are governed by the same physical laws. There have been a variety of productive approaches to qualitative reasoning about physical systems used to analyze their behavior over time [2]. Although detailed analysis procedures are different, the basic idea behind all of them can be clarified by decomposing qualitative reasoning into two tasks:

- **Model-building task:** creates a qualitative differential equation as a model of a physical situation.
- **Qualitative simulation task:** starts with a qualitative differential equation, and predicts the possible behaviors following the model.

The CML is a declarative model-building language for logically specifying the symbolic and mathematical properties of the structure and behavior of physical systems qualitatively. CML can specify an abstract behavior of the plant by using qualitative physics and causality, instead of differential equations and time function. As illustrated in Figure 5, the physical situation is modeled as a collection of model fragments, each of which represents a physical object or a conceptually distinct physical phenomenon, such as a particular aspect of component behavior or a physical process. A model fragment representing a phenomenon specifies a set of conditions under which the phenomenon occurs and a set of consequences of the phenomenon. The conditions specify a set of instances of object classes that must exist (called Participants) and a set of relations (called Conditions) that must hold among those objects and their attributes for the phenomenon to occur. The consequences hold when an instance of the model fragment is active. Model fragment consequences are typically qualitative differential equations that describe the behavior of the entities that satisfy the conditions.

The physical situation and structure of the plant are captured in a general purpose domain theory that describes a class of related phenomena and systems. A domain theory consists of a set of quantified definitions, called model fragments, each of which describes some partial piece of the domain's physics, such as the processes (e.g., liquid flows), devices (e.g., transistors), and objects (e.g., containers). Each definition applies whenever there exists a set of participants for whom the stated conditions are satisfied. A specific system or situation being modeled is called a scenario. A model of the scenario consists of fragments that logically follow from the domain theory and the scenario definition.

A typical implementation supporting CML might be used as follows to predict the behavior of a nuclear power plant. A domain theory is constructed to describe the objects, systems, and phenomena present in a typical nuclear power plant. The theory would include physical phenomena such as mass and heat flows, boiling, evaporation, and condensation; it would also include chemical reactions, the effects of catalysts, and models of components such as reaction vessels, pumps, controllers, and filters. A major goal of CML is to support the interchange and reuse of such theories.

Once the domain theory has been constructed, it can be used to model many different processing plants under a variety of different conditions. The user specifies a

scenario that defines an initial configuration of the plant, the initial values of some of the parameters that are relevant to modeling it, and perhaps conditions that further characterize the system. The CML implementation automatically identifies model fragments that are applicable in the scenario. These model fragments are made into a single model that consists of both a symbolic description as well as a set of governing equations. The equations may be solved or simulated to produce a behavioral description. Because the conditions under which the model fragments hold are explicit in the domain theory, the system automatically constructs additional models that describe the plant as it moves into new operating regions.

The part of the plant model specified by NAT in Figure 4 can be transformed to a model fragment, specified by CML in Figure 5, which represent the behavior of a rechargeable battery when it is charged beyond its normal operating range. The model fragment definition states that it requires a rechargeable battery whose charge level is over 30 ampere-hours. When these conditions are satisfied, the consequence is that the electro-motive force produced by the battery can be expressed as a monotonically increasing function, denoted as M+ in the example, of the charge level.

Battery-over-charged
Participants: (rechargeable-battery ?b)
Conditions: (Charge ?b) > 30.0 amp-hours
Consequences: (Electro-Motive-Force ?b) = M+((Charge-level ?b))

Figure 5. Model fragment example

While the EPS is modeled by a differential equation and a time function in Figure 4, the same physical situation can be declaratively modeled by a qualitative differential equation, denoted M+, and a causal relation as in Figure 5. Given a CML representation of the physical situation, DME generates an equation model, analyzes its behavior, and gives a causal explanation for the behavior. The role of a qualitative simulator, such as DME, is to identify the set of model fragment instances that are active in a scenario and compose a mathematical model from the consequences of the activated instances. The behavior of the plant model which is specified by CML and transformed from the original model, C=NAT(M), can then be verified against the required behavior model, which is described by CFRL and transformed from C=REQ(M), by a qualitative simulation under DME.

3.3 Transformation of the required behavior model

The REQ relation in the Four variable model specifies the required external behavior of the system. Required behavior relates specific, observable changes in the environment to observable system actions. REQ is typically a relation instead of a function because there is tolerance in the required behavior of time, value, or both. The controlled variables specify what observable behavior the software must produce in response to environmental changes. For example, Figure 6 shows the controlled variable definition and the REQ relation of a required behavior of EPS.

1. Controlled Variables

Name	Type/Values	Physical Interpretation
RState	:ENUMERATED:	
	ON	It opens the electrical connection.
	OFF	It closes the electrical connection.

2. REQ Relation (Required Behavior)

Mode	Condition	Behavior of RState
RState = OFF	RBvoltage > 33.8 volts	RState = ON
RState = ON	RBvoltage < 31.0 volts	RState = OFF

Figure 6. Controlled variable and REQ relation

In practice, although REQ is a relation, it is specified by giving an ideal behavior as a function and defining the allowed tolerances in value or time separately. It means the required behavior of physical systems should be specified qualitatively. It is hard to verify formally that behavior of the plant satisfies the required behavior modeled by C=REQ(M) because NAT is just the external constraints imposed by the plant. When the required behavior is described by CFRL, it has been possible to verify formally the required behavior against the plant because CFRL has the same behavior-based semantics as the behavior of the plant.

CFRL is a language for specifying an expected function of a device and defines its semantics in terms of the type of behavior representation used in model-based, qualitative simulation. In order to use both functional and behavioral models to verify the behavior of the real-time system, it is crucial that the functional model is represented in such a way that it has a clear

interpretation in terms of observed behavior. For example, in EPS the function of a charge current controller is to prevent damage to a battery by cutting off the charge current when the battery is fully charged. To be able to determine whether this function is actually accomplished by an observed behavior of the device, the representation of the function must specify conditions that can be evaluated against the behavior. Such conditions might include the occurrence of a temporal sequence of expected events and causal relations among the events and the components. Without clear semantics given to a representation of functions in terms of behavior, it would be impossible to evaluate a design based on its predicted behavior and intended functions. An essential element in a requirement model is causality. In order to say that a device has achieved a function, which may be expressed as a condition on the state of the world, one must show not only that the condition is satisfied, but also that the device has participated in the causal process bringing about the condition

A behavior is a linear sequence of states. The output of a qualitative simulation by DME is a graph of states. Each path through the graph represents a possible behavior over time. We call such a path, i.e., a linear sequence of states, a trajectory. Because the semantics of CFRL is defined in terms of the matching between a trajectory of the plant's states and a functional specification of the required behavior, the language is immediately useful for the purpose of behavior verification. The language allows engineers to explicitly describe the physical context in which the function is to be achieved, the structural characteristics of the device that are assumed in the function description, and the causal sequence of events that must occur for the function to be achieved [9]. In CFRL, the function F is a four-tuple Ef, Df, Cf, Gf, where:

- Ef is the name of a function which F elaborates.
- Df is a description of the device of which F is a function.
- Cf is a description of the context in which the device is to function.
- Gf is a description of the functional goal to be achieved.

Figure 7 shows an example of the top-level representation of the required behavior of the EPS. The device description, Df, consists of three parts, Device, Components, and Conditions. The Conditions typically specify aspects of the device structure that are assumed in the description of the function. The specification assumes that there exists a generic description of the Electrical-power-system with a Plus-terminal and Minus-terminal as its components. The notion of a device function assumes some physical context in which the device is placed, and Cf is a specification of such a context. The description of the functional goal, Gf, is represented as an expression consisting of a Causal Process Description(CPD), quantifiers, and Boolean connectives. There are two quantifiers, ALWAYS and SOMETIMES. Connectives are AND, OR, IMPLIES, and NOT. A CPD is a directed graph, in which each node describes a state and each arc describes a temporal relation between states. CPDs are abstract descriptions of expected behavior in terms of a conditionalized sequence of events. A CPD can be considered as an abstract specification of a trajectory. Unlike a trajectory, it does not specify every state or everything known about each state. It only specifies some of the facts that must be true during the course of the trajectory and partial temporal/causal orderings among those facts. In other words, ordering is total for states in a trajectory because a trajectory is a linear sequence of states, while the order is partial for states in a CPD. The intuitive meaning of a CPD is that:

- For each node in the CPD, there must be a state in the trajectory in which the condition specified by the node is satisfied, and
- For each pair of nodes directly connected by an arc, the causal and temporal relationships specified by the arc must hold in the trajectory.

Ef: nil
Df: **Device**: (?eps Electrical-power-system)
Cf: **Objects**:
(?sun Sun)
(?load Electrical-load)
Conditions:
(Electrically-connected (Plus-terminal ?eps)
(Plus-terminal ?load))
(Electrically-connected (Minus-terminal ?eps)
(Minus-terminal ?load))
Gf: (ALWAYS (Powered-p ?load))

Figure 7. Function F1 of EPS

3.4 Verification of the behavior

When the shutdown system of the Wolsong nuclear power plant in Korea was developed by a Rational Design Process (RDP) [11] and the Four variable model,

there was no method to verify the required behavior modeled as C=REQ(M) against the plant model. While the software design description of the system specified by OUT(SOF(IN(M))) in Figure 2, was formally verified with respect to the software requirements specification, modeled as C=REQ(M), the software requirements itself was informally reviewed against the system requirements. That is, it was not possible to verify proposition (2), NAT(M) and OUT(SOF(IN(M))) => REQ(M).

In our approach, after the required behavior C=REQ(M) is transformed into the required behavior model, specified CFRL, and the plant model C=NAT(M) is transformed into a model fragment by the CML, it is possible to verify the plant against the required behavior because the semantic of CFRL is defined in terms of matching between a trajectory of the plant's states and a functional specification of the required behavior. The DME is used to do a qualitative simulation of the behavior of EPS, and produce the behavior trajectory from the plant model represented by CML.

The notion of a causal dependency relation between states in a behavior trajectory and the causal constraints of a CPD arc is used to define the requirements for a behavior trajectory needed to achieve a function represented by CFRL. The causal dependency among facts in a trajectory is based on the theory of causal ordering [7]. Though mathematical equations are inherently acausal and symmetric, causal ordering theory can be used to reveal causal dependencies among variables in a set of equations and produce a directed graph structure that reflects an intuitive notion of causal relations among variables. Given a self-contained set of equations, the theory essentially orders variables in the order they can be solved, starting from the given independent variables. The algorithm that verifies trajectories produced by the DME system with respect to a function specified in CFRL was developed by Iwasaki's work [9]. Given a function F and trajectory T, the function verifier performs the following tests:

- Determines whether trajectory T achieves function F;
- If trajectory T achieves function F and function F elaborates function F', determines whether trajectory T also achieves function F'.

These tests either confirm or refute the claim that the device denoted by Df of the model fragment in Figure 7 achieves function F. Defining verification with respect to a specific trajectory, avoids the generally undecidable problem of proving that a given design always implies the achievement of F. If a case is discovered where the design fails to achieve F, it is proof that either the design or the functional specification must be revised.

Though the algorithm is still exponential in the number of states of the trajectory, the behavior verification capability, the prediction capability, and the causal relationship can be useful designing of the safety-critical hybrid real-time system. Since the simulation system, such as DME, knows the exact condition necessary at each branching point to follow a desired path, there is also a possibility of using the conditions as a guide for safety analysis, like fault tree analysis in the requirement phase.

4. Conclusion

In this paper, we demonstrated that a qualitative formal method is effective in specifying and verifying the behavior of a real-time process control system. The qualitative reasoning approach is selected to help the designer specify his insight on the properties of the physical plant, to eliminate unnecessary numerical details, and to predict the behavior of the embedded, real-time system in the requirements phase. We have illustrated an application of our approach by converting a SCR-style specification, used in several safety-critical industrial applications, into qualitative models. While existing analysis methods on SCR-style specification did not provide adequate means of verifying proposition (2), we were able to overcome such a limitation using a qualitative formal method.

Furthermore, our approach enables causal reasoning when an explanation on how observed behavior is actually achieved. Reasoning from knowledge of the structure and the physical principles underlying functions is similar to that of humans. We believe that such causal analysis capability can be effectively used to guide safety analysis on the system. The causal ordering which results from qualitative reasoning on the dynamic behavior of physical systems could assume the role of guide in determining the root cause a fault.

References

[1] A. Barr, P. R. Cohen, and E. A. Feigenbaum, The Handbook of Artificial Intelligence, Vol. IV, Addison-Wesley Pub. 1989.

[2] D. Bobrow, Qualitative Reasoning about Physical Systems, MIT Press, 1985.

[3] B. Falkenhainer, A. Farquhar, D. Bobrow, R. Fikes,

K. Forbus, T. Gruber, Y. Iwasaki, and B. Kuipers, "CML: A Compositional Modeling Language," KSL in SRI Technical Report KSL-94-16, 1994.

[4] A. R. Courcoubetis, C. Henzinger, and P. H. Ho, "Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems," Proceedings of the Workshop on the Theory of Hybrid Systems in Lyngby, Denmark, 1992.

[5] C. A. R. Hoare, Communicating Sequential Processes, Prentice-Hall International, 1985.

[6] Y. Iwasaki and C. M. Low, "Model generation and simulation of device behavior with continuous and discrete change," Intelligent Systems Engineering, 1(2), 1993.

[7] Y. Iwasaki and H. A. Simon. "Causality and Model Abstraction," Artificial Intelligence, Vol. 67(1), May 1994.

[8] Y. Iwasaki, A. Farquhar, V. Saraswat, D. Bobrow, and V. Gupta. "Modeling Time in Hybrid Systems: How Fast is 'Instantaneous'?," Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, 1995.

[9] Y. Iwasaki, M. Vescovi, R. Fikes and B. Chandrasekaran. "A Causal Functional Representation Language with Behavior-Based Semantics," Applied Artificial Intelligence, Vol. 9(1), pp. 5-31, Jan. 1995.

[10] J. S. Ostroff, Temporal Logic for Real-Time Systems, Research Studies Press Ltd. 1989.

[11] D.L. Parnas and P.C. Clements. "A Rational Design Process: How and Why to Fake It," IEEE TSE, SE-12(2), Feb. 1986.

[12] D.L. Parnas, J. Madey, "Functional Documentation for Computer Systems Engineering," Technical Report 90-287, Queen's University, TRIO, Sep. 1990.

[13] A. P. Ravn, H. Rischel, and K. M. Hansen, "Specifying and Verifying Requirements of Real-Time Systems," IEEE TSE, vol. 19, no. 1, Jan. 1993.

[14] Wolsong NPP 2/3/4, "Software Requirements Specification for Shutdown System 2 PDC," 86-68350-SRS-001, Rev. 0, Jun. 1993.