Analysis Capabilities for Requirements Specified in Statecharts [1]

Bonnie E. Melhart
Nancy G. Leveson
Dept. of Information and Computer Science
University of California, Irvine

Matthew S. Jaffe
Hughes Aircraft Company
Ground Systems Group
Fullerton, California

Abstract

This paper considers various types of analysis that are possible for formal requirements specifications in the Statecharts language. The application of recently developed criteria for completeness analysis of embedded systems requirements to specifications in Statecharts is discussed, in particular. Additions for the language that will enable such analysis are indicated.

Introduction

As today's computer systems are asked to perform more and more complex tasks, the individual components of the task or system and their interface become more complex, and the difficulty increases of assuring such properties as consistency, completeness, and unambiguity in the specifications. Embedded systems, in particular, have become increasingly complex as the software is tasked to interface with many machines and humans at once, often with stringent fault tolerance, safety, and reliability requirements. Evaluation for correctness and other properties is essential early on, i.e. at the requirements level, before commitment is made to a physical solution that may be costly or even impossible to change. Postponing such analysis is inadvisable: time and effort spent on verification of the design or the implementation to the requirements specification are wasted if the requirements are incorrect.

Work is ongoing at the University of California, Irvine to develop a formal model that can be used to determine and describe the features required for a real-time requirements specification language suitable for embedded systems and to develop completeness analysis methods based on this model. The focus has been on those aspects of the requirements specification that are often handled poorly or inadequately such as safety, robustness, timing, and the human-machine interface.

[1]This work has been partially supported by NSF Grant CCR-8718001, NSF VPW Grant 8800505-RII, NSF CER Grant DCR-8521398, and NASA Grant NAG-1-668.

The language used to specify software requirements is important not only for its ability to express and to communicate the requirements, but also for its ability to enable analyses of the requirements specification. It is therefore interesting and appropriate to consider such analysis capabilities in Statecharts [Har87], a state-of-the-art language for the specification of complex embedded systems.

In [MLJ88], the authors present a critical evaluation of Statecharts as a language that enables such analysis and suggest some additional features that are needed to completely specify real-time embedded systems. The application of the completeness analysis procedures that we have developed to an example specification in Statecharts is examined there as well. A brief description of a few of the results of this work is given here.

Statecharts

Statecharts is a language under development at Weizmann Institute in Israel by David Harel and his colleagues. It is intended for use in defining real-time reactive systems and is appropriate for the embedded applications on which this research is focused.

There are several characteristics of a language used for specifying software requirements that enable the application of various analyses of the resulting system description. In general, to preclude the need for some sort of natural language processor to manipulate the description, a formal, nonprose language is desired for requirements specification. On the other hand, formal languages can be hard to read and understand; tradeoffs are necessary when selecting a language that must be appropriate for use on many levels (i.e., design, validation, specification, analysis). As Statecharts employs a graph structure with multiple abstraction levels to represent an underlying formalism based on finite state machines, formal analysis can be focused on a certain level of detail in the specification or on parts of the representation that are relevant without looking at the entire specification.

Timing is an essential consideration for real-time embedded software. A means for representing minimum and/or maximum times for certain activities is present in the language. Syntax is provided to specify desired transition in the event of a time-out; that is, the passing of a given time since the last occurrence of a specified event. This capability is impor-

©1989 ACM 0-89791-305-1/89/0500/0100$00.75

tant to most analysis techniques. Another important feature for requirements analysis is the capability to include an environmental model as an orthogonal (independent, parallel) state to the software decription. Thus analyses of the requirements specification can include analysis of the software model interacting with its environment.

The next two sections describe two specific analyses for requirements level descriptions that have been considered for application to a Statecharts model.

Safety analysis

Leveson and Stolzy [LS87] have developed analysis techniques for timed Petri net specifications to analyze a model for safety and fault tolerance properties and to derive associated timing and functional requirements from a global system model. These techniques basically require a finite state machine formalism, the ability to derive at least a partial reachability graph, and the specification of maximum and minimum timing for state transitions. These capabilities are all a part of the Statecharts formalism, though not entirely backed by the current formal semantics. The procedures and algorithms can be transferred to Statechart specifications, with certain additions to the basic Statecharts model for modeling failures. There may even be advantages in the use of Statecharts for the specification since they allow the possibility of concentrating on detailed specification of hazardous states while performing less detailed analysis of non-critical functionality and performance.

Some additional features, however, may be necessary to completely specify and analyze timing requirements. In Statecharts, a state configuration of the software system is actually a tuple of substates of orthogonal components. If this "system state" then has a timing requirement, it should go into effect when the last orthogonal substate is entered. Sometimes it is desirable to give certain transitions priority over others for safety considerations; it is not clear how to give priority to a transition with timing on such a system state, since the substates may have starting times at quite a distance (time and space) from each other. Much work remains to be done in the area of time modeling.

There are five basic elements of a failure model considered in the Leveson and Stolzy work:

- a required event that does not occur,
- an undesired event,
- an incorrect sequence of required events,
- two incompatible events that occur simultaneously, and
- timing failures in event sequences.

These can be modeled in Statecharts by making use of the maximum and minimum timing features. Additional features are recommended to express, in particular, the case of incompatible simultaneous events. If the events are at the top level of the software description, a global transition is appropriate for the simultaneous occurrence. In fact, global transitions are quite useful for handling undesired events that require the same transition for all states. However, the general case demands a feature that transitions from perhaps widely separated internal states. To further facilitate analysis, there should also be some distinction between the normal, expected (desired) transitions and the error transition, which is usually, but not always, a time-out transition. The Leveson and Stolzy notion of a failure transition can be added to the Statecharts model in a straightforward way.

Completeness analysis

There is little consensus on what constitutes completeness in a requirements specification. Jaffe and Leveson [JL88] have attempted to provide a formal definition of completeness in real-time requirements specification and to identify criteria that can be used to detect and generate missing requirements for a particular specification. The analysis is applied to a black-box behavioral description that is composed of a set of assertions of the form: *trigger* $\Leftrightarrow$ *output*. The triggers and outputs are described using first-order predicate calculus and a simple state machine model. The general format of a requirements assertion in the model is:

$$\exists E_1, \ldots, E_i \ni P_E \Leftrightarrow \exists! O \ni P_O$$

where:

- E_i: observables in environment (including passage of time), inputs and/or outputs,
- P_E: predicate on trigger events, and
- P_O: predicate on outputs (the trigger must existentially quantify any events used in the definition of P_O).

The E_i are the necessary state history or conditions for an output O.

Criteria have been developed to formally define completeness of both trigger conditions (i.e., robustness criteria) and output predicates using logical completeness to "close" the set of requirements. Closure with respect to trigger conditions ensures that undesired or unexpected events are considered and the behavior desired should they occur is specified, while closure with respect to outputs ensures that all relevant output behavior is specified. In addition, application-dependent rules and heuristics can be developed to close the output specification with respect to various criteria important in the controlled system. Criteria have been developed for safety, reachability, and path robustness (i.e., fault tolerance).

These criteria require a language for specification capable of expressing time and value for outputs, assumptions of timing and value ranges for environmental triggers and internal state information. Statecharts environmental model and state representation provide an appropriate format and necessary information to allow most analysis for the completeness of environmental assumptions and requirements that concern

states. It may be desirable, however, to express time and value range assumptions elsewhere with reference to these specifications on the corresponding transitions so that arcs in the representation do not become cluttered. To specify exceptions to these assumptions of range and timing of input triggers, an additional state is generally required for each assumption. Addition to the Statecharts formalism of a specific mechanism for modeling exceptions would promote readibility and allow analysis.

It is important to have an appropriate syntax and semantics for expressing timing requirements, as certain analyses rely heavily on a formal semantic representation of timing and time-outs. Most critical states will require double timing specifications for minimum and maximum timing of certain activities that result in output. Notation for these should be made less confusing. In particular, expression of periodic functions is ambiguous in most representations as one description can allow or disallow phase shift depending on the interpretation. (This problem is described in detail in [JL88].) Constructs to make the desired timing explicit are necessary before these requirements can be completely specified.

Summary

No current real-time requirements specification language contains all the features we feel are necessary to completely specify a real-time, embedded system. Many features of Statecharts facilitate the analysis of the requirements. Orthogonality and hierarchy structuring, which augment the underlying finite state machine model, provide the modular breakdown needed for dealing with large, complex systems and allow specification of prerequisite inputs to assure their existence before use. The state representation allows evaluation for dynamic properties through reachability analysis. Some analysis is simplified as the specification of global requirements is possible with superstates in the hierarchy. The inclusion of the environmental model and means for expressing timing constraints are important for embedded systems; in particular, time-outs allow expression of some exceptions to timing assumptions for the environment.

Some additions have been suggested for the language that would enable analysis for safety and completeness of a Statecharts model. The Statecharts language is still under development, and it is hoped that the formal semantics will be completed in the near future. Semantics for maximum and minimum timing and time-out transitions are especially important. Features to describe value ranges and exceptions concisely and to indicate error and undesired event transitions are recommended. More details about how analysis might be done and suggestions for adding missing capabilities to the Statecharts language may be found in [MLJ88].

In general, there is a need for more scientific approaches to analyzing requirements specification documents for correctness. Informal techniques do not provide the reliability or confidence demanded for today's complex safety-critical systems. Techniques for analysis of the external completeness of the model and methods to analyze the specification of undesired event handling for completeness and consistency are a critical need for real-time embedded systems specification. Special modeling abstractions may be required to make complete specification of these models possible. While developing formal analysis methods is a complex endeavor, there are real benefits for doing so, as the alternative may mean incomplete and incorrect specifications that result in systems expensive to correct and that may even lead to disaster.

References

[Har87] David Harel. Statecharts: A Visual formalism for complex systems. *Science of Computer Programming*, 8:231–274, 1987.

[JL88] Matthew S. Jaffe and Nancy G. Leveson. Completeness, robustness, and safety in real-time software requirements specification. Technical report, Dept. of Information and Computer Science, University of California, Irvine, September 1988.

[LS87] Nancy G. Leveson and Janice L. Stolzy. Safety analysis using Petri nets. *IEEE Transactions on Software Engineering*, SE-13(3):386–397, March 1987.

[MLJ88] Bonnie E. Melhart, Nancy G. Leveson, and Matthew S. Jaffe. Analysis capabilities for requirements specified in Statecharts. Technical report, Dept. of Information and Computer Science, University of California, Irvine, September 1988.