13

Modelling the Requirements of Process Controlled Systems

T.K. Sateesh and P.A.V. Hall

Computing Department, Faculty of Mathematics and Computing
The Open University, Milton Keynes MK7 6AA, United Kingdom

Abstract

Computer controlled systems with strict timing requirements are ubiquitous. Temporal requirements are often the timing restrictions imposed by the application environment. Real-time properties concerns quantitative timing constraints. We introduce the concepts of Timed Requirements Language (TRL) a language invented primarily for the exposition of temporal properties of process controlled systems. TRL is designed to project operational behaviour through time. TRL is based on event model and realises the behaviour of system with strict timing constraints. TRL treats functionality and timing in the same framework.

Keyword Code: D.2.0; D.2.1; D.2.2
Keywords: Software Engineering, General; Requirements; Tools and Techniques

1. INTRODUCTION

Requirements of a system evolve as a set of simple needs. These needs are comprehended with many essential properties of that class of system. The technique employed for the evolution of requirements are pertinent only to a particular class of system as the properties differ from one class to another. In defining requirements the goal is to maximise the functionality of system while minimising the failure. Requirements are derived from a wide range of concepts. The focus here is to capture real time requirements of systems. A major application of computers has been in the control of physical processes like industrial plant control, nuclear reactors, flight control systems and patient monitoring systems. These systems are characterised by hard response time and dynamic environment. Process controlled systems normally interact with external world entities. A system can be described from external user's point of view. External description establishes associations among activities which occur within different subsystems of a system. Sequence of interaction among components provides a description of the system operation and furnishes the operational behaviour of the system. Requirements specification based on real world models are transparent and easy to understand. System interactions are related to the communication among components and environment. In order to avoid any misunderstanding requirements must be expressed in a formal representation. A formalism specifies a class of objects under discussion in an unambiguous and general manner [9]. For easy analysis requirements are required to be stated in a language understandable by computers.

Process controlled systems have to satisfy stringent timing characteristics. Timing constraints arise while maintaining the stability of plant. These timing constraints depend on the physical characteristics of plant. System needs are required to be represented in a language that incorporates timing feature. As Leveson [6] observes 'most real-time specification languages have only simple primitives for timing, such as a time-out, that do not adequately capture the complexities of time and therefore are inadequate for fully specifying and modelling requirements'. A requirements language must facilitate the description of temporal properties that are meaningful for these systems. Substantial work has been carried out by a number of researchers in the requirements specification of real-time systems like [1] [8]. The works on temporal requirements [3] [8] [5] has influenced us. Our language philosophy builds on these works and simplifies system evolution to handle possibly unforeseen requirements. In this article we introduce the concepts of timed requirements language (TRL) to specify real time aspects of the systems. TRL is designed to specify systems where temporal aspects are of considerable interest. TRL promotes a descriptive method. TRL emphasises understanding what takes place and when it takes place in the system, which are system behaviours.

M. Lee et al. (eds.), *Software Quality and Productivity*

2. MODEL OF SYSTEM

System modelling starts with informal requirements and a set of assumptions on the system and environment. A model is a simplified representation of something to be developed. In general model is an abstraction of the system. Model gives description of the function of the real system. Model incorporates the essential feature of the system. A model of a computational paradigm is a set of direct or indirect observations that can be made of a computational process [4]. A system is composed of interacting components. Behaviour arises out of time ordered interactions between subsystems. These interactions provide a qualitative description of how the system works. A system description constructs a model of a system. This model identifies the approximate structure of system by recognising the important incidents that occur. The word structure refers to a partial description of system showing it as a collection of parts and showing relations between the parts. Balzer [2] argues that system requirements must be a cognitive model. It must describe a system as perceived by its user community. The object it manipulates must correspond to the real objects of that domain.

A simple model of a real time system can be described by the time ordered interactions between subsystems. System is connected by physical devices which operate in their own time scales. Computer is said to operate in real-time if actions carried out by the computer relate to the time scales of the external tasks. Typically the requirement of a computer is to complete a set of operations within a prespecified time. Majority of tasks fall into this category. Events occur in real time without any reference to the operations inside computer. The sequence of actions is determined by the events occurring in the environment. Controller has to synchronise its operations with that of environment. Environment is an active part of the system. It is the responsibility of the controller to overcome any unsatisfactory behaviour of environment. While system modelling, environment has to be modelled first. This becomes essential to capture the lag time and response time requirements.

2.1. An Approach to Modelling

Functional requirements originates from a sense of causation. A letter can not be read unless it is received, and a customer gets what he orders. The requirement is a chain that mirrors this causal relationship among several events occurring in a system. Real-time systems interact with physical devices which are monitored and controlled. A complex system is a combination of interacting components. In all these systems one device energises the other. The behaviour of a system is the causal relationship among real world events. Requirements evolve from this simple set of behaviour. Requirement of a system involves the order of occurrence of events and also constraints on the time of occurrence. We make use of event based model to capture the behaviour of system. The behaviour of a software system may be characterised by the sequence of events which occur. Distinct events carry distinct labels. Each event is assumed to occur instantaneously. Event occurrence marks a point in time, which is significant for the analysis of dynamic behaviour of the system. Thus event serves as a temporal marker. Continuous events which consume a definite time interval and is vital for reasoning are represented by two atomic events like start of the event and end of that event. By atomicity we mean that the events are indivisible.

In event model, one is interested in the ongoing process involving real world entities (i.e., how an event is caused, how an event affects other events, and which event is dependent on other events). Description of behaviour produces a chronological relationships between corresponding transactions. With real-time systems we are interested in the precise sequencing of operations and the detailed timing and control characteristics of external devices. Behaviour is a sequence of exchange of information between the co-operating units. Behavioural descriptions has to be operational to aid the easy analysis of system behaviour. Data flow diagram shows only the flow of data and is not interested in the sequence of activities occurring. Event sequence represents a finer detail of activities involved in processing the action. Behavioural specification describes constraints on observable behaviour. A constraint imposes a restriction on the behaviour of a system. A commonsense description of the system's behaviour envisions the relationship between the components of system and their potential behaviour. Functional description of system is described in terms of the exact role of its components. The component model is responsibility driven rather than data driven. We can model the functions as agents cooperating with each other to achieve the goal. Agent characterises the resources and the task assigned to it. Agents can be

either concrete or abstract. A concrete agent may have a representation in system like a button, a printer and so on. An abstract agent may have no direct representation in system, instead it models a behaviour which is a set of operations that it can be requested to carry out. Abstract agent models a certain responsibility of system. Responsibility of a system as conceived by a user emphasises the utility point of view.

3. MODELLING LANGUAGE

Functional behaviour is a collection of operations which selectively manipulates the agents to provide an intended function. TRL describes behaviour using only the interactions with which the system and its environment communicate. Behaviour of a system is defined by behaviour expression. Behaviour expression is a sequence of interaction of events.

behaviour : behaviour expression --> setof (sequenceof (TimedEvent))

In other words behaviour expression consists of a set of events that can take part.

A behaviour definition is of the form

(behaviour (e_1, e_2, .. e_n))

where behaviour is the language construct that binds events (e_1 , e_2 , . . e_n). System activity is asserted by defining bodies of processes. Processes are used to describe the dynamics of the system. Process consists of a sequence of behaviour expressions, where each expression is justified by expression previous in the sequence. With TRL computations are described in terms of processes. We treat all communication mechanisms by manipulation of events. TRL treats the system as a set of processes. Process reflects a synoptic view of system consisting of a sequence of events that the real system may engage.

A **process P** is a set of behaviour expressions of the form

(behaviour (e_1, e_2, .. e_n)) . . . .

(behaviour (e_1, e_2, .. e_n))

In TRL every event is a timed event. A timed event associates a time parameter with event. It is expressed as (button_pressed, t1) where t1 is timing parameter associated with event 'button pressed'.

3.1. Language Constructs

Requirements can be elicited by stepping through the scenario in which events motivate the system to behave in a particular pattern. A triggering mechanism provides the basis for describing these scenarios. Events are defined as what happens to the environment. This provides a qualitative reasoning about observed or postulated behaviour. Such a reasoning is quite useful for generating explanations on what can happen and how the things that happen can interact. System description is derived from the behavioural description of components and their interconnections. Behavioural description of each component reveals its objective.

Causal ordering manifests relation between <triggering event> and <action sequence>. Action sequence describes the complete pattern of behaviour. Action sequence may consist of one or more events. In other words each event may *energise* finetly one or more events. *Energise* is a strict partial ordering on events with utmost one immediate predecessor event. Action sequence may be represented by a single event or as a composition of events. It is possible to specify several action events which all meet the same selection criteria. Action sequence is to be performed one at a time and in the order they are written. A system can react to several types of events. For example an automated library assistant (ALA) can react differently depending on the type of order received. ALA depending on the order received generates a collection of events or a single event.

IF (command = reserve, t1) THEN (proceed to reserve a book, t2)
ELSIF (command = overdue_notice, t3) THEN (proceed to issue notice, t4)
ELSIF (command = check_status, t5) THEN (proceed to check the status, t6)
ELSIF (command = help, t7) THEN (provide help, t8) FI

A boolean condition may also be associated with the triggering event. Triggering operation takes place only if the condition is satisfied. An absence of a condition is evaluated as condition = true. Following Martin-Löf's constructive type theory [7] we define types as predicates that state the properties of system or its components. For example a book can only be reserved if it is available in library. This is expressed as

IF (command = reserve, t1) ASWELL (book is available) THEN
(proceed to reserve a book, t2) FI

Defunct behaviour is also built into TRL. Behaviour function on defunct behaviour (NIL) is an empty set. Other constructs like 'do' and 'with' are used to capture the other scenarios of system activity.

3.2. Representation of timing constraints

Timing constraints are an essential part of real-time requirements. Dasarathy[3] provides a classification of timing constraints. A slightly different approach to temporal requirements is employed in [5]. Timing constraints of both types can easily be expressed with TRL. More complicated timing constraints can be expressed involving event associated timers. The timing constraint is expressed as a relation between the timing parameters, like

IF (command = reserve , t1) THEN (proceed to reserve a book , t2)
WHERE (t2 <= t1+5) ^ (t2 > t1 +1) FI

WHERE specifies a condition on the time period between events. Temporal requirements are the qualitative and quantitative statements about the required temporal order of activities. Timing constraints are explicit specifications of the behaviour of system components. Timing constraints in TRL are specified by timing relation(s) between events. Any timing constraint user defined or system derived can easily be expressed. Timing constraints like the latest allowable start time, the earliest allowable start time and the interval during which a task is allowed to start are easily specified. Timing constraint can involve the constraint over many events. An operation like measuring the level of water in a tank or measuring the temperature of furnace is to be accomplished periodically. Periodic behaviour represents such repeated actions. Repeated behaviour is associated with a predicate. As long as this predicate evaluates true, specified action is observed. Periodicity of repetition is specified. Periodic behaviour is represented with 'while' construct. TRL treats functionality and timing in the same framework. Naturally it is possible to build several behaviours for the same system. These can be developed independently to simplify the task of requirements evolution. Here we are not describing the language in detail. It is not essential to this paper, as here we are only considering the concepts of language.

4. CONCLUSION

TRL has been designed to specify systems where temporal properties are of interest. The essential feature of the language is being descriptive and this is appropriate for modelling real time aspects of activity. Description requires the identification of events contained in system. As this method is parametrized with respect to events in a system, it allows to treat different systems in a uniform way. In essence, TRL provides a notation for the description of system models.
The importance of obtaining conceptual model requirements has been underlined by various researchers. TRL has been used with some real-time systems. PREET (Process-oriented REquirements Elucidating Tool) checks the validity and consistency of requirements expressed in TRL. Also PREET provides description of the temporal and functional behaviour through timed transition diagrams. It helps to experiment with the description of systems, as the descriptions can easily be modified and then analysed for their suitability.

REFERENCES

1. Alford M.W, 'A Requirements Engineering Methodology for Real-time Processing Requirements', IEEE Transactions On Software Engineering, January 1977, pp. 60 - 69
2. Balzer R.M. and Goldman N.M, 'Principles of Good Software Specification and their Implication for Specification Languages', in Proceedings of the Specifications of Reliable Software Conference, April 1979, pp. 58 - 67
3. Dasarathy B, 'Timing Constraints of Real-time Systems: Constructs for Expressing Them, Methods for Validating Them', IEEE Transactions On Software Engineering, SE- 11 (1) Jan 1985, pp. 80 - 86
4. Hoare C.A.R, 'Let's Make Models', in LNCS - 458, Ed J.C.M. Baeten and J.W. Klop, Theories of Concurrency: Unification and Extension, Springer Verlag, 90 pp. 33
5. Jahanian F and Mok A.K, 'Safety Analysis of Timing Properties in Real-Time Systems', IEEE Transactions on Software Engineering, Se-12(9), Sept. 1986, pp. 890 - 904
6. Leveson N.G, 'The Challenge of Building Process-control Software', IEEE Software, November 1990, pp. 55 - 62
7. Nordström B and Smith J, 'Propositions And Specifications of Programs in Martin-Löf's Type Theory', BIT 24(1984) pp. 288 - 301
8. Zave P, 'An Operational Approach to Requirements Specification for Embedded Systems', IEEE Transactions on Software Engineering, vol. SE-8, May 1982, pp. 250 - 269
9. Zeigler B.P, 'Multifaceted Modelling and Discrete Event Simulation', Academic Press, Orlando 1984