Abstract

Event-B is a language for the formal development of reactive systems. At present the RODIN toolkit [15] for Event-B is used for modeling requirements, specifying refinements and doing verification. In order to extend graphical requirements modeling capability into the real-time domain, where timing constraints are essential, we propose a Timing diagram (TD) [13] notation for Event-B. The UML 2.0 based notation provides an intuitive graphical specification capability for timing constraints and causal dependencies between system events. A translation scheme to Event-B is proposed and presented. Support for model refinement is provided. A partial case study is used to demonstrate the translation in practice.

1. Introduction

There are many ways to describe the system requirements such as using goal orientation e.g. Knowledge Acquisition in Automated Specification (KAOS), Formal methods (FM) such as Event-B, and graphical notations such as UML diagrams.

KAOS [3] is a Goal-modeling requirement specifications technique. KAOS uses Goal and Operational models to declare system requirements in a form of Linear Temporal Logic (LTL), and has a concept of goal refinement to refine a goal into subgoals. Event-B [8, 17] is a formal method (FM) that describes system requirements in a form of a set-theoretic notation. Event-B is used to improve formal requirements analysis, verifying each next level of detail (refinement) and helping eliminate error early in the design process. The timing diagram (TD) is an UML2.0 notation, used to show the behaviors of objects over time.

As [12, 14, 16], FM can be difficult to construct and demands trained professionals. Demonstration of requirements in graphical forms helps software developers to define specification more easily than by using the FM mathematical notations. Thus, there are many researchers trying to bridge the gap between Event-B and UML diagrams. For example [1, 7, 9] and UML-B [5, 6]. UML-B is a front end to the Event-B and is a toolkit developed in RODIN [15]. The tool provides graphical modeling capability in term of UML-like Class diagrams and Statecharts. Even though those researchers combining Event-B with UML diagrams, do not focus on the combination of timing constraints and casual dependencies among different objects’ states, other essential part of requirements analysis, to the Event-B model as our research does.

The aim of our research is to develop techniques to specify requirements by using graphical notations, i.e. TD, and then generating KAOS, Event-B and UML-B models. Our work comprised two main steps. First, we amend UML2.0 TD notation and define its Backus Naur Form (BNF) [4]. Second, we generate three kinds of pattern to create KAOS, Event-B and UML-B models from TD. A lift system based on Jackson’s work [10] is used as a primitive case study. The case study has been modified by adding timing constraints to dependency requirements on events to demonstrate the issues. In this paper, we present a subset of our complete set of translation rules for generating Event-B from TD. The translation rules from TD to KAOS can be found in [18].

In section 2, we describe the TD; section 3 illustrates case study specifications; section 4 describes the Event-B modeling; section 5 explains how patterns
are used for generating Event-B; section 6, we make conclusions.

2. Timing Diagrams

Examples of TD notations are used in the research is shown below

An arrowed line indicates cause and effect between objects. The beginning of line represents the cause while the end of the line (with arrow) represents the effect. One can identify additional conditions which make state changes by plain text above the arrowed line. A duration constraint is used to describe how long a state or value must be in effect and identified by a symbol \([t1,t2]\) where \(t1\) and \(t2\) indicate the timing constraint starts from \(t1\) to \(t2\).

AND OR “AND” and “OR” notations are used for specifying relationships within between CauseEffect arrows (note, they are not used to contribute a cause to many effect segments). A node can has many “AND” or “OR” causes which are represented by dash-line here.

3. Case study

Figure 1 shows part of the lift specification used in this paper. The requirements maybe described “…A part of the lift system contains two objects: the floor sensor and the lift. The lift movement states are separated into three steps: moving up, stop at floor and moving down. The floor sensor has two states: on and off. The relation between the lift movement and the floor sensors means whenever a user presses a button to request the lift, the lift starts moving up/down (1 and 2) from the current floor; within between 2 – 5 seconds after the lift starts moving, the floor sensor of the current floor will turn off…”

In Figure 1, in term of TD notations, we can describe that there are two Timelines which represent the state change in time for that particular object: floorsensor and lift. The lines 1 and 2 show the combination of the CauseEffect arrow by using “OR” notation; it means the floorsensor is set to Off according to whether the lift is in the state of MvgUp or MvgDwn. Predicates such as \(f = currentF1 \land dir = Up\) are additional conditions on the CauseEffect arrow where \(f\) represents a floor and is a dynamic state parameter that can change in time. Object states and their indices such as On1 and Off2 represent segments of the floorsensor Timeline. They are not TD notations but used in translation rules as described in section 5.

![Figure 1. Timing diagram](image)

4. The Event-B Modeling

The dynamic part of an Event-B model is called the MACHINE. The MACHINE includes state VARIABLES definitions, an INVARIANT predicate on the state, INITIALISATION and EVENTS. In Event-B, the units of behavior are called EVENTS. Each event \(E\) is composed of an enabling guard \(G(l,v)\) and an action \(S(l,v)\) where \(v\) are state variables constrained by invariants \(I(v)\); \(l\) are local variables that the event may contain. The general form of an event is, \(E = \text{ANY } l \text{ WHERE } G(l,v) \text{ THEN } S(l,v) \text{ END}\). The structure of a Event-B model is shown below

\[
\text{MACHINE name = VARIABLES } v, \ldots \text{ INVARIANT } I(v), \ldots \\
\text{INITIALISATION \ldots} \\
\text{EVENTS} \\
\text{eventname = ANY \ldots WHERE \ldots THEN \ldots END} \\
\text{eventname = \ldots} \\
\text{END}
\]

5. Pattern Transforming Timing Diagrams into Event-B

5.1 BNF Timing Diagrams definitions

We identify TD BNF definitions and use them to create the translation rules to transform TD to Event-B. Examples of BNF definitions for Timeline are shown in the following.

A Timeline comprises a chain of segments which individual segment represents the object state (Objst) and its position (Index) in the Timeline.

\[
\text{Timeline ::= Segment'} \\
\text{Segment ::= Objst Index; Index ::= INT}
\]
5.2 Translation rules

Normally, one rule comprises many basic sub-rules. Examples of those basic sub-rules are illustrated below.

**Rule 1:** \[ \text{TObj}(\text{Segment}) \rightarrow \text{Obj}; \]
This rule gives the object for an input segment.

**Rule 2:** \[ \text{TState}(\text{Segment}) \rightarrow \text{Objst}; \]
This rule gives the object state for an input segment.

In this paper we demonstrate the `TimeCtnt` rule as an example for generating a duration constraint as guards for `floorsensorOff` event (figure 2). This rule is defined as recursion from the main rule (not shown in this paper) and uses a segment (\text{segm}) as an input parameter. The detail of the rule is shown in the following.

\[ \text{TimeCtnt}(\text{segm}) \rightarrow \]
\[ (\text{gclock} - \text{TObj}(\text{segm})\text{TState}(\text{segm})\text{Time}) \geq \]
\[ \text{LOWER\_LIMIT\_TObj}(\text{segm}) \land \]
\[ (\text{gclock} - \text{TObj}(\text{segm})\text{TState}(\text{segm})\text{Time}) \leq \]
\[ \text{UPPER\_LIMIT\_TObj}(\text{segm}) \]

When `MvgUp2` and `MvgDwn3` are used as parameters, parts of event’s guards are generated as shown in figure 2 (in rectangle); where `LOWER\_LIMIT\_floorsensor` = 2 and `UPPER\_LIMIT\_floorsensor` = 5 are created as constants by another rule (not show in this paper).

```
floorsensorOff =
ANY ....
WHERE ....

\{( (\text{gclock} - \text{liftMvgUpTime}) \geq \}
\{ \text{LOWER\_LIMIT\_lift} \land \}
\{ (\text{gclock} - \text{liftMvgUpTime}) \leq \}
\{ \text{UPPER\_LIMIT\_lift} \} \}
\}
\}
\}
\}
\}
\}

\{ \}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
\}
7. References


Expectation and benefits statements

1. How can Timing diagram be used to advance the requirement engineering and Formal method?

2. How is the beneficial of using Timing diagrams to model critical systems’ requirements?

   It would be helpful for us to justify whether the Timing diagram is useful for use in the critical systems. Moreover, we would like to have opinions/suggestions for what kind of state-based system that cannot be explained by Timing diagrams? So, we can identify the limitations of our Timing diagram.

3. How far the Timing diagram can be used to demonstrate human actions? How can we model environments of the system if that is human?

   Since there are many requirements concern with human activities, for example in the lift system that needs human to request the lift by pressing buttons. In this case, we can demonstrate the pressing activity by represent it as an event in Event-B.

   However, there is a case study such as Ambulance Service system. The timing constraints are concerning with responding to emergency calls requiring the rapid intervention of an ambulance. How can we identify these requirements in the Timing diagrams and how we control the timing constraints?

4. Is there any issues/problems with our currently translation rules?

   We really appreciate if anyone can point out any problems in our model. So, we can correct the translation rule in order to obtain complete models.

5. What are the model integration issues we should be aware of?