Real-Time Systems Modeling Based on Aspect-Oriented Timed Statecharts

Xinxiu Wen 1, Huiqun Yu 1

1 College of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
{wenxinxiu, yhq}@ecust.edu.cn

Abstract. Real-time systems modeling is a challenging task in the area of software engineering. To improve the modularization of real-time systems effectively and facilitate the verification of systems model formally, this paper proposes a formal modeling method with aspect-oriented timed statecharts. An elevator example illustrates our modeling method.

Keywords: Real-time systems; Aspect-orientation; Timed statecharts; Weaving.

1 Introduction

It is challenging to model real-time systems formally because of their complexities. Aspect-oriented software modeling (AOSM) has attracted much attention in recent years due to separating core concerns from crosscutting concerns successfully and improving software modularity effectively [1]. Real-time systems are in essence composed of core and crosscutting concerns, thus applying aspect-orientation concepts to real-time systems modeling is very helpful. In this paper, requirements of real-time systems are divided into base functional requirements, including functional requirements, extended functional requirements and non-functional properties. Base functional requirements are modeled as base timed statecharts (BTSCs), while the others are modeled as aspect timed statecharts (ATSCs). Furthermore, BTSCs and ATSCs are woven into aspect-oriented timed statecharts (AOTSCs) for model checking. To illustrate our approach vividly, we go through with an elevator example.

The remainder of this paper is organized as follows. Section 2 gives a brief introduction to base timed statechart. Section 3 describes our aspect-oriented modeling method. Section 4 is the conclusion.
2 Background

Statecharts were first introduced by David Harel to describe the complex behavior of reactive systems in 1987 [2], while UML statecharts are object-oriented variations of Harel’s statecharts with properties as orthogonality, refinement, etc.

**Definition 1** A base timed statechart (BTSC) is a 7-tuple \((S, E, C, T, r, \rho, \text{type})\), where: \(S\) is a finite set of states; \(E\) is a finite set of events; \(C\) is a finite set of clocks; \(T = T_1 \cup T_2\) is a finite set of transitions, \(T_1\) represents immediate event transitions that are triggered by inputs, while \(T_2\) represents timed transitions that depend on time rather than inputs; \(r \in S\) is root; \(\rho(s)\) specifies the direct offspring of state \(s\); \(\text{type}(s) \in \{\text{BASIC}, \text{AND}, \text{OR}\}\) means type of state \(s\).

3 Aspect-oriented Modeling with Timed Statecharts

Aspect timed statecharts extend base timed statecharts with pointcut and advices for modeling crosscutting concerns of real-time systems.

**Definition 2** An aspect timed statechart (ATSC) is a 3-tuple \((A, P, \text{tp})\), where: \(A\) is a set of advices, each advice is a base timed statechart; \(P\) means pointcut, it is a finite set of join points (join points are certain transitions of BTSC); \(\text{tp} \in \{\text{before}, \text{after}, \text{around}\}\) is the type of advices.

Elevator control system is a typical case of real-time systems [3]. Moving up, moving down, opening the doors and closing the doors are base functions of an elevator, they should be modeled into the BTSC (Fig. 1). Timeout handling is non-functional property, passengers’ requests scheduling and passengers weighing are including requirements. Each of them should be modeled as an ATSC separately (Fig. 2). From Fig.2.a we know that the crosscutting position of aspect TIMEOUT should be the join point transition \(t_8/t_9\) of the BTSC, which means when the clock exceeds timing constraint (1 second for \(p_{\text{open}}\) event or 8 seconds for \(p_{\text{close}}\) event), transition \(t_a\) will substitute for the base join point transition \(t_8/t_9\). Fig.2.b illustrates the PLAN aspect and from the graph we know that transition \(t_{a1}(t_{a1}\text{ is immediate})\) would be triggered after base join points transition \(t_5/t_6\). Fig.2.c illustrates the CHECKING aspect and transition \(t_{a1}(t_{a1}\text{ is immediate})\) would be triggered before base join points transition \(t_9\).

The introduction of ATSC realizes the modularization and formalization of crosscutting concerns at the design phase, the next work is to weave BTSC with ATSC into aspect-oriented timed statechart for model checking.

**Definition 3** An aspect-oriented timed statechart (AOTSC) is a pair \((\text{BTSC}, \text{ATSC}_i, 1 \leq i \leq n)\), where: \(\text{BTSC}\) represents a base timed statechart; \(\text{ATSC}_i\) represents an aspect timed statechart, \(n\) is the number of aspect timed statecharts.

This paper adopts weaving mechanism like AspectJ, which join points are well-defined points in the execution of BTSC [4]:

1. Initially, \(\text{AOTSC} = \text{BTSC}\);
2. For each aspect timed statechart \(\text{ATSC}_i\), the weaving process is as follows according to the type of advices: If the type of advice is before or after, then substitute advice of \(\text{ATSC}_i\) for transitions at join points of BTSC. If the type of advice is
around, then interrupt the transitions at join points of BTSC and execute the advice of ATSC$_i$;

(3) If there are several advices which apply to the same join point, it is necessary to plan their weaving sequences. If the advices are in the same ATSC$_i$, then their positions decide their precedence; if they are in different ATSCs, then declaring precedence specifies their weaving sequences with BTSC.

aspect TIMEOUT
transition pointcut handle(Bi, ei, ci, Bj):
    (ARRIVED, p_open, , OPENED)||(OPENED, p_close, , CLOSED)
advice replace
type around

aspect PLAN
transition pointcut schedule(Bi, ei, ci, Bj):
    (READYUP, up, , MOVINGUP)||(READYDOWN, down, , MOVINGDOWN)
advice schedule
type after

aspect CHECKING
transition pointcut weigh(Bi, ei, ci, Bj):
    (OPENED, p_close, , CLOSED)
advice weigh
type before

Fig.1 The BTSC of elevator

Fig.2 The ATSCs of elevator

4 Conclusions

This paper introduces a formal method for modeling real-time systems with AOTSCs. The method not only separates real-time systems requirements successfully, but also formalizes them rigorously. In future work, we are planning to verify the model by NuSMV.

References