Problems of the semantic-based weaving of scenarios

Jacques Klein, Jean-Marc Jézéquel

HAL Id: hal-00795065

https://inria.hal.science/hal-00795065

Submitted on 27 Feb 2013

Problems of the Semantic-based Weaving of Scenarios[1]

Jacques Klein, Jean-Marc Jézéquel

IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
{jklein, jezequel}@irisa.fr

Abstract

The notion of aspect looks promising for handling cross-cutting concerns earlier in the software lifecycle, up from programming to design, analysis and even requirements. Support for aspects is thus now raising interest also at the modelling level, including in behavioural modelling languages such as scenarios. However with these kinds of behavioural modelling languages, aspect weaving cannot always be performed at the abstract syntax level. In this paper we present the problems relating to the design of a semantic based aspect weaver for Hierarchical Message Sequence Charts (HMSCs).

1. Introduction

The idea of encapsulating cross-cutting concerns into the notion of aspects looks very promising for complementing the usual notion of modules available in most languages. By localizing these cross-cutting concerns, the software engineer can get a greater degree control over variations, either in the spatial dimension (product line context) or in the temporal dimension (software evolutions). The need to isolate these cross-cutting concerns has been popularized by the AspectJ programming language, but there is a growing interest in also handling them earlier in the software lifecycle, for instance at design time [2],[4] or during requirements analysis [1]. Beyond being able to represent aspects at requirement or design time, an automatic aspect weaver at these stages can be very useful for validation purposes (simulation or test case generation) and also for targeting non-aspect-oriented platforms (e.g. vanilla Java).

There are indeed many works addressing model level aspect weaving, including in behavioural modelling languages such as scenarios, as in Hierarchical Message Sequence Charts (HMSCs). However with these kinds of behavioural modelling languages, aspect weaving cannot always be performed at the abstract syntax level. For instance, we might consider the simple HMSC of Figure 1a, which consists of a loop over a basic scenario where we have a message 'a' and then a message 'b'. Imagine now that we want to weave some extra-behaviour into our system each time a message 'a' directly follows a message 'b' (called a pointcut in the AOSD community). A simple specification of this pointcut is given in Figure 1b as another scenario. While at the semantic level our aspect should be woven in the base scenario of Figure 1, there is no way we can do it without the knowledge of the semantics of the loop construct. In other words, we cannot do it at the abstract

syntax level. Clearly the problem is the same for other HMSC composition operators such as alt, seq etc.

As we will discuss later in the paper, the problem is even worse than that, and in some cases may raise some interesting computability issues.
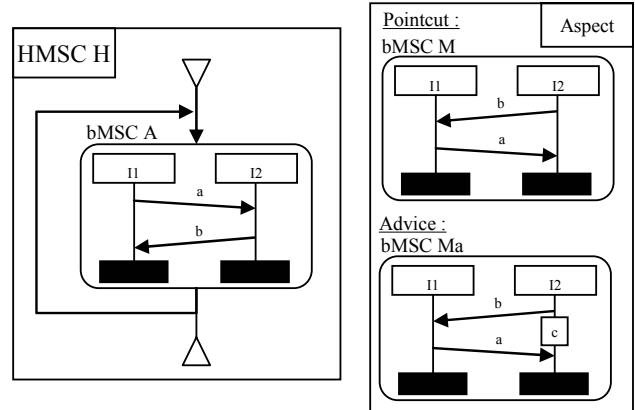

Figure 1 :   a) a HMSC   b) the two parts of an aspect

In this paper we present the problems relating to the design of a semantic-based aspect weaver for HMSC. Two major problems appear: one relating to the detection of a pointcut within a loop, and one relating to the detection of a pointcut across a loop. The rest of the paper is organized as follows. Section 2 presents the used language to describe the scenarios and the aspects. Section 3 and 4 describe the two problems cited above.

2. Scenarios and Behavioural Aspects Weaver

Scenario languages have been the subject of significant interest during the last decade. They are mainly used to describe behaviours of distributed systems at an abstract level or to capture requirements in early development stages with a clear, graphical, and intuitive representation. More specifically, we use scenarios expressed as Message Sequence charts (MSC) [3], which propose two levels of specifications. At the lowest level, basic MSCs (bMSCs) describe simple communication patterns between entities of the system called instances. In the Figure 1b, the bMSC Advice represents a behaviour where the messages b and a are exchanged between the instances I1 and I2, and where a local action c is performed on I2.

However, bMSCs alone do not have a sufficient expressive power: they can only define finite behaviours, without real alternatives. For this reason, MSCs have been extended with High-level MSCs, a higher level of specification. HMSCs allow the composition of bMSCs with composition operators such as sequence, alternative and loop. Figure 2 shows a HMSC where the bMSCs A and A1 are composed alternatively, A1 and A2 are composed sequentially, A2 is in a loop, etc... So, a HMSC is a bMSC automaton and it defines a set of bMSCs (a bMSC for each path that defines a HMSC). By example, in the Figure 1, the HMSC defines the set of bMSCs S= {A, AA, AAA, …}. We can also say that a HMSC defines a language where the words are the bMSCs of S.

Notably, UML 2.0 Sequence Diagrams [6] are largely inspired by MSCs (a comparison between UML2.0 and MSC-2000 is given in [5]), and so this paper concerns also sequence diagrams.

Consider the HMSC H in the Figure 1a and the aspect in the Figure 1b. The goal of the semantic-based weaver is to produce a new HMSC H' (Figure 2) where the aspect is woven each time that the pointcut is detected within a bMSC of the set of bMSCs defined by the HMSC H. More specifically, we use aspects, which we call behavioural aspects because they can be specified by two bMSCs as depicted on Figure 1b: one (the pointcut) allows the specification of the pattern to detect, and one (the advice) describes the expected behaviour.

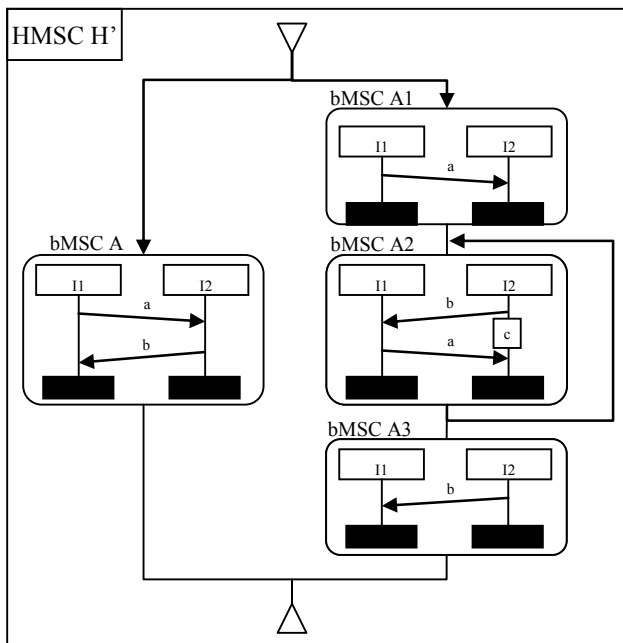The two following sections show that it is not always possible to produce a woven HMSC.

Figure 2 : Result of the weaving

3. First Problem: an infinite number of potential matches

Consider a HMSC H and a bMSC pointcut P. By respecting the semantic of HMSCs, H defines a set S of bMSCs: a bMSC for each path of H.

Firstly, if S is a finite set, as the pointcut P is a bMSC (and thus finite by definition), it is clear that the search for patterns will be done in a finite time.

Secondly, if S is an infinite set (as for the HMSC of the Figure 1a), i.e. as soon as the HMSC H contains at least one loop, the problem of the search for patterns is not trivial. For this problem, we introduce a new notion which we call *potential match*.

Roughly speaking, for a bMSC D and a pointcut P, the potential match P' is a part of D which represents what could be matched by P if we add messages after D. For example, for the bMSC A and the pointcut M of Figure 1, there exists a potential match M' which is a bMSC with only the message b. If there were no potential match, there would be no chance of finding the pointcut in a longer bMSC. However, in this case, as the potential match exists, if we want to detect the pointcut M, we have to take a longer bMSC. So, we take the bMSC AA, where M matches one time and where there exists the same potential match. As the potential match is the same, we can transform the HMSC H into the HMSC H' (Figure2) where the aspect is woven.

However, there exist cases where the potential matches are always different as shown in Figure 3. Indeed, for a bMSC B^n (B…B n times), the potential match contains n local actions 'a', for each n>0. So, the expected behaviour will be a succession of n bMSCs representing the behaviour of the advice followed by n bMSCs with only a local action 'a', for each n>0. The expected behaviour can be seen as a kind of irregular language of type X^nY^n (where X is a bMSC specifying the behaviour of the advice, and Y is a bMSC with only the local action a). As a HMSC is a bMSC automaton, it defines only regular languages and so it is impossible to represent the expected behaviour by a HMSC.

4. Second Problem: matching across a loop

The previous problem was relating to the detection of the pointcut within a loop, and we have shown that if the number of potential matches is infinite, it is not always possible to specify the expected behaviour with a new HMSC. A second problem can appear when the detection of the pointcut is done across a loop. Figure 4 shows an example where the pointcut M is detected within the bMSCs ABC, ABBC, …, AB…BC, but where B doesn't participate in the match. Here too, the expected behaviour cannot be specified by a HMSC because the behaviour can be seen as a kind of irregular language of type X^nMaY^n (where X is a bMSC with only a local action y, and Y is a bMSC with only a local action x).

5. Conclusion

Beyond being able to represent aspects at requirement or design time, an automatic aspect weaver at these stages can be very useful for validation purposes (simulation or test case generation) and also for targeting non-aspect-oriented platforms (e.g. vanilla Java). As the behavioural modelling language, we have chosen HMSCs, but if we respect the semantic of HMSCs two major problems appear when we want to weave an aspect described by bMSCs into a HMSC to produce a new HMSC. These problems cause behaviours which cannot always be specified by a HMSC.
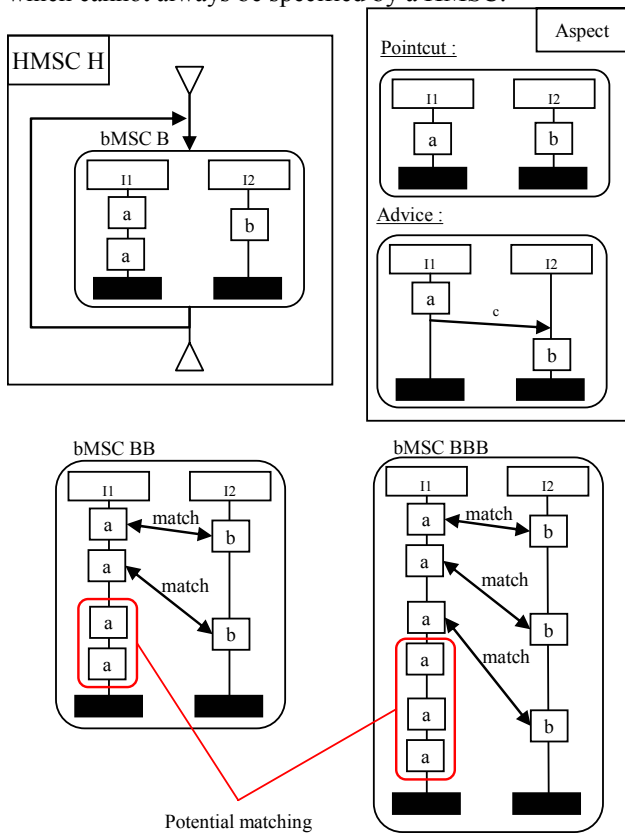

Figure 3 : problem relating to the number of potential match

References

[1] J. Araujo, J Whittle, and D.-K. Kim. Modeling and composing scenario-based requirements with aspects. *In Proc. of the 12th IEEE International Requirements Engineering Conference*. Japan. September 2004.

[2] S. Clarke and R. J. Walker. Composition patterns: An approach to designing reusable aspects. *In International Conference on Software Engineering*, 2001.

[3] ITU-TS. ITU-TS Recommendation Z.120: Message Sequence Chart (MSC). ITU-TS, September 1999.

[4] Robert France, Dae-Kyoo Kim, Sudipto Ghosh, and Eunjee Song, "A UML-Based Pattern Specification Technique," *IEEE Transactions on Software Engineering*, 2004

[5] O. Haugen. Comparing UML 2.0 interactions and MSC-2000. *In Proceedings of SAM 2004: SDL and MSC fourth International Workshop*. LNCS 3319, 2004.

[6] Object Management Group. Unified modeling language specification version 2.0: Superstructure. Technical Report pct/03-08-02, OMG, 2003.
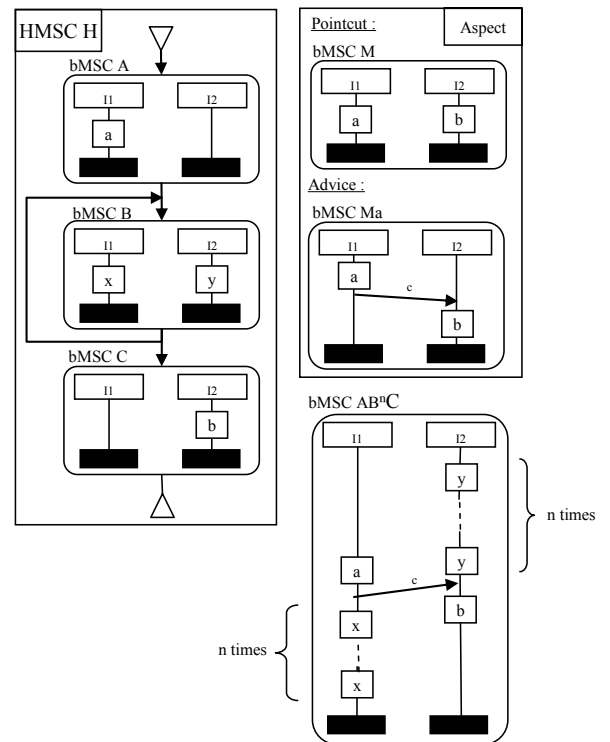
Figure 4 : problem relating to a matching through a loop