Supporting the reconciliation of models of object behaviour

George Spanoudakis, Hyoseob Kim*

City University, Department of Computing, Northampton Square, London EC1V 0HB, UK;
E-mail: gespan@soi.city.ac.uk

Received: 23 September 2002 / Accepted: 18 August 2003
Published online: 1 April 2004 – © Springer-Verlag 2004

Abstract. This paper presents Reconciliation+, a method which identifies overlaps between models of software systems behaviour expressed as UML object interaction diagrams (i.e., sequence and/or collaboration diagrams), checks whether the overlapping elements of these models satisfy specific consistency rules and, in cases where they violate these rules, guides software designers in handling the detected inconsistencies. The method detects overlaps between object interaction diagrams by using a probabilistic message matching algorithm that has been developed for this purpose. The guidance to software designers on when to check for inconsistencies and how to deal with them is delivered by enacting a built-in process model that specifies the consistency rules that can be checked against overlapping models and different ways of handling violations of these rules. Reconciliation+ is supported by a toolkit. It has also been evaluated in a case study. This case study has produced positive results which are discussed in the paper.

Keywords: Consistency management – Software design models – Object interaction diagrams

1 Introduction

The specification of software system behaviour using multiple object interaction diagrams (i.e., sequence and/or collaboration diagrams) creates the potential of conflicting specifications of messages, objects and operations in these models. This is because different object interaction diagrams may, by virtue of the exchanges of messages that they specify and other elements in the specifications of these messages, imply different behaviours for the same objects and operations.

Consider, for example, an object model for a library system that includes the object interaction diagrams $I_1$ and $I_2$ of Fig. 1 and the class diagram of Fig. 2. The diagrams $I_1$ and $I_2$ specify interactions, which occur when the library system is used to search for items in the library either by keywords which refer to the author of an item ($I_1$) or by keywords which refer to the title of an item ($I_2$). The class diagram of Fig. 2 specifies the classes of the objects that participate in the interactions of $I_1$ and $I_2$. According to $I_1$ and $I_2$, the library system: (i) gets search keywords from a UI component (see messages $11$:getText() in $I_1$ and $8$:getText() in $I_2$); (ii) formulates a database query (see message $9$:formulateQuery() in $I_2$); and (iii) executes the query (see messages $12$:executeQuery(SQLStatement) in $I_1$ and $10$:executeQuery(SQLStatement) in $I_2$).

In this example, it is plausible to assume that the messages $10$:actionPerformed(ActionEvent) in $I_1$ and $7$:actionPerformed(ActionEvent) in $I_2$ overlap since (in the current state of the models) they both appear to invoke the operation $\text{actionPerformed}(e:\text{ActionEvent})$ of the class DatabaseActionListener in Fig. 2. If, however, this assumption is correct then the specifications of the behaviour of the operation $\text{actionPerformed}(e:\text{ActionEvent})$ which are implied by $I_1$ and $I_2$ are conflicting. This is because according to $I_2$ (but not $I_1$) the execution of the operation $\text{actionPerformed}(e:\text{ActionEvent})$ leads to the dispatch of the message $9$:formulateQuery() and therefore the execution of the operation $\text{formulateQuery}()$ of the class Manager.

Conflicts of this form need to be detected and reconciled in the design phase of a system development project.
in order to eliminate ambiguities that could be more expensive to resolve at the implementation phase.

In this paper, we describe a tool-supported method, called Reconciliation+, that we have developed to support software designers to detect and handle conflicts in behavioural specifications in models of object interactions which are expressed as sequence (or collaboration) diagrams in UML [9]. This method is a newly developed extension of a method for managing modelling discrepancies in structural object models that is discussed in [15].

Conflicts in Reconciliation+ are detected as violations of consistency rules. A consistency rule sets the conditions that should be satisfied by overlapping messages in different object interaction diagrams. In the above example of the library system, for instance, the conflict in the specifications of the behaviour of the operation actionPerformed(e:ActionEvent) implied by I₁ and I₂