On the Semantics of Operation Contracts in Conceptual Modeling[1]

Anna Queralt and Ernest Teniente

Universitat Politècnica de Catalunya
Dept. de Llenguatges i Sistemes Informàtics
{aqueralt, teniente}@lsi.upc.edu

Abstract. This paper describes two different ways of understanding operation contracts in conceptual modeling: the *strict* and the *extended* interpretations. The main difference between them lies in the way operation postconditions and integrity constraints are guaranteed. Understanding an operation contract under each of these interpretations can result in different semantics for the same specification. Thus, assuming one or another affects the way operation contracts are specified.

1. Introduction

An information system maintains a representation of the state of a domain in its information base. The state of the information base is the set of instances of the entity and relationship types defined in the conceptual schema. Additionally, a conceptual schema contains a set of integrity constraints that define conditions that each state of the information base should satisfy.

The content of the information base changes due to the execution of operations. The effect of an operation on the information base is usually defined by means of an operation contract, composed of pre and postconditions.

Several books on conceptual modeling give precise definitions for integrity constraints and also for pre and postconditions [3-5, 7, 9-12]. However, they usually pay little attention to the precise semantics of operation contracts since, in general, they do not establish an explicit relation between operation postconditions and the integrity constraints defined in the conceptual schema.

Depending on the way postconditions and integrity constraints are guaranteed, operation contracts can be understood in two different ways, which we call *strict* and *extended* interpretations. Under a strict interpretation, an operation has a passive behaviour, since it cannot be applied if some integrity constraint is violated (although its precondition is satisfied). Instead, an extended interpretation entails a reactive behaviour of operations, since it must take care of maintaining integrity constraints satisfac-

[1] This work has been partially supported by the Ministerio de Ciencia y Tecnología and the FEDER funds, under the project TIC2002-00744.

Proceedings of the CAiSE'05 Forum - O. Belo, J. Eder, J. Falcão e Cunha, O. Pastor (Eds.)
 - ISBN 972-752-078-2

tion, so that the operation will always be applied if its precondition is satisfied (and it is possible to repair all the violated integrity constraints).

Our work is independent of any specific conceptual modeling language and, thus, our results can be applied to any of them provided that it allows the definition of operations and explicit integrity constraints. In addition, they can be applied to almost all current conceptual modeling approaches like the ones in [3-5, 7-10].

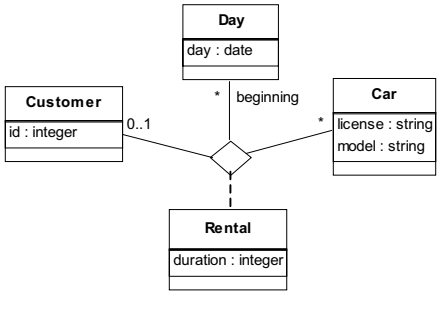
2. Basic Concepts

The content of the information base changes due to the execution of operations. A behavioural schema contains a set of operations and the definition of their effect on the information base. This knowledge is usually defined by the preconditions and postconditions of the operations. A precondition expresses a condition that must be satisfied when the call to the operation is done. A postcondition expresses a condition that the new state of the information base must satisfy. The execution of an operation results in a set of one or more structural events to be applied. The application of a set of structural events to a state *IB* of the information base results in a new state of the same information base.

A structural event (SE) is an elementary change (creation or deletion of instances) in the population of an entity or relationship type. We will only use here one kind of SE, the relationship insertion: *newLink(RelType(part$_1$,...,part$_n$, [attr$_1$,...,attr$_n$]))*.

Given a state of the information base *IB*, there are several sets of SEs that lead to new states satisfying an operation postcondition. From those, we are only interested in the minimum ones. A set *S* of SEs is minimum if no proper subset of *S* is also a set of SEs that satisfies the postcondition. This is the way in which we deal with the frame problem [1]. Note that, given a postcondition, several minimum sets of SEs may be acceptable, but only exceptionally when some kind of non-deterministic behaviour is desired.

We present a simple conceptual schema that models car rentals, which will serve as an example throughout the paper. The conceptual schema is specified by means of a UML class diagram, and the additional constraints needed are defined textually.



Integrity constraints

IC1: Customers are identified by id

IC2: Cars are identified by license

IC3: Days are identified by day

IC4: There cannot exist overlapping rentals of the same car

Fig. 1. Class diagram and integrity constraints for car rentals

3. Strict and Extended Interpretations of a Postcondition

An operation precondition expresses requirements that any call must satisfy in order to be correct, while its postcondition expresses properties that are ensured in return by the execution of the call [6].

In addition to pre and postconditions, integrity constraints play an important role in the definition of the semantics of operation contracts, since they must be preserved by all the operations. The semantics of operation contracts in conceptual modeling we identify in this paper differ mainly on the way integrity constraints are guaranteed.

Given a state of an information base *IB* and an operation *Op*, the semantics of *Op* defines the conditions under which *Op* can be applied, and the new state of the information base we obtain as a result of applying *Op* to *IB*.

Let *Pre* and *Post* be the precondition and the postcondition of *Op*. Let *S* be the minimum set of SEs that satisfies *Post* when applied to *IB*. Let *IC* be the set of integrity constraints defined in the conceptual schema.

Definition 1: Strict Interpretation of an Operation Postcondition

Under a strict interpretation, an operation *Op* is applied to *IB* iff the following two conditions hold:

a) *Pre* and *IC* hold before applying *S* to *IB*
b) *Post* and *IC* hold after applying *S* to *IB*

Intuitively, the first condition states that there is a transition from an information base *IB* to a new state as a result of applying *Op* only if *IB* satisfies *Pre* and it is consistent (i.e. it satisfies all integrity constraints). Moreover, according to the second condition, the new state of the information base is obtained exactly as a result of applying to *IB* the minimum set *S* of SEs that satisfies *Post*, as long as it is consistent. If any of the conditions does not hold, then *Op* is not applied to *IB*.

Definition 2: *Extended Interpretation of an Operation Postcondition*

Under an extended interpretation, an operation *Op* is applied to *IB* iff the following two conditions hold:

a) *Pre* and *IC* hold before applying *S* to *IB*
b) *Post* and *IC* hold after applying either *S* or a minimal superset of *S* to *IB*

The first condition states, as before, that the transition is only possible if *IB* satisfies *Pre* and it is consistent. The second condition asserts now that to obtain the new state of the information base at least the SEs in *S* must be applied. However, it does not discard the application of additional SEs in order to restore consistency when needed.

From previous proposals, [3, 7, 8] follow an extended interpretation to define the semantics of operation contracts.

As we said before, the main difference between both interpretations relies on how integrity constraints are handled. As stated in Definition 1, a strict interpretation defines a set *S* of SEs, determined only by the postcondition, to perform the transition from *IB* to the new state according to *Op*. *S* may only be applied to *IB* if it does not lead to a violation of any integrity constraint. Otherwise, *Op* must be rejected.

On the contrary, an extended interpretation allows for several different sets of SEs to be applied to *IB*, provided that all of them include at least the SEs in *S*. The additional SEs must be such that they guarantee that no constraint is violated in the resulting state, even though some of them were violated by the events in *S*. Clearly, if *S* itself does not violate any constraint there is no need for considering additional SEs.

To illustrate the previous definitions, assume the following contract for an operation *rent*, aimed at registering the rental of a car for a period of time starting in the current date:

```
Operation:  rent(cust:Customer, car: Car, num-days:Integer)
Pre:
            - The car is not assigned to an overlapping rental
Post:       - A new rental of the car car, for a duration of
            num-days, made by the customer cust and starting at
            the current date is created
```

Fig. 2. Contract for the operation *rent*

Note that, in this example, we have that *S={newLink(Rental(cust, car, currentDate, num-days))}* suffices to satisfy the postcondition of *rent(cust, car, num-days)*.

Let us start by assuming a strict interpretation of the contract in Figure 2. We are going to show the conditions under which *rent* can be applied to *IB* and the state of the information base resulting of applying *rent* according to this semantics. We distinguish two different situations depending on the contents of *IB*:

1. *IB* already contains an overlapping rental of the same car. In this case, the operation may not be applied since condition a) of Definition 1 is not satisfied.
2. *IB* does not contain an overlapping rental of the same car. Then, condition a) is guaranteed, since *Pre* is satisfied and *IB* is consistent. This time the operation can be applied, and the new state of the information base is obtained by applying *S* to *IB,* that is by creating a new link between the customer cust, the car car and the current date. This new state satisfies b) since *S* necessarily satisfies the postcondition and applying *S* to *IB* never violates any integrity constraint. Note that the only integrity constraint that *rent* may violate is IC4, but this will never happen when *IB* does not contain any overlapping rental of the same car.

Summarizing, we have that according to a strict interpretation the semantics of the operation *rent* is the following. If *IB* does not contain any overlapping rental of the same car then the operation results on the application of the SE *newLink(Rental(cust, car, currentDate, num-days))*. Otherwise, the operation may not be applied.

If we assume now an extended interpretation of the operation in Figure 2, we have that when *IB* contains a user identified by *email* the operation will not be executed. If this does not happen, the application of *S={insert(Registered(email,creditCard))}* suffices also to satisfy the four conditions of Definition 2, following the same reasoning as for the strict interpretation. Hence, we have in this particular example that the semantics of strict and extended interpretations coincide. The main reason for that coincidence is that no integrity constraint is violated by *S* and, thus, the application of *S* is enough to obtain *IB'* in both cases.

However, according to [2], the previous contract presents an important drawback regarding desirable properties of software specifications. The problem lies in the fact

that the operation precondition is redundant, since the same aspect of the specified system (there cannot exist overlapping rentals of the same car) is already guaranteed by an integrity constraint.

To avoid redundancy in the specification of the operation *rent* we should define an operation contract like the one in Figure 2 but with an empty precondition. The set *S* will be the same (*S={newLink(Rental(cust, car, currentDate, num-days))}*). However, we have now that the semantics of both interpretations does not coincide anymore.

In a strict interpretation, the semantics of *rent* is the same as the one of the redundant contract. We show it by distinguishing again the same situations:

1. *IB* contains an overlapping rental of the same car. Then, condition a) is guaranteed, since *Pre* is empty. However, if we apply *S* to *IB,* a new link between the customer, the car and the current date will be created, and this new state will always violate IC4. We have then that *rent* cannot be applied to *IB,* since it is impossible to satisfy all conditions required by a strict interpretation.
2. *IB* does not contain any overlapping rental of the same car. In this case, condition a) is guaranteed and the new state of *IB* satisfies b), since *S* necessarily satisfies the postcondition and applying *S* to *IB* never violates IC4. Therefore, the operation is applied in this situation.

In an extended interpretation, we must distinguish also the same two situations:

1. *IB* contains an overlapping rental *r* of the same car. Then a) is guaranteed. Condition b) states that the new state of *IB* must satisfy both *Post* (this is always true since *S* necessarily satisfies it) and *IC*. Since *S* itself violates the first integrity constraint, the set of SEs to be applied must be a superset of *S*. In particular, it must contain additional SEs to repair this violation. This can be done adding SEs that either delete the overlapping rental *r*, change its beginning, change its duration or change its car.
2. *IB* does not contain any overlapping rental of the same car. The behaviour in this case is the same as the one of the strict interpretation.

Summarizing, the semantics of the non-redundant operation contract for *rent* according to an extended interpretation is the following. If *IB* does not contain an overlapping rental of the same car, the operation results in the application of the SE *newLink(Rental(cust, car, currentDate, num-days))*. Otherwise, i.e. *IB* contains another rental of the same car that overlaps with the new one, it results on the application of *S* plus additional SEs that delete or modify the existing rental in such a way that IC4 is not violated.

As we have seen, the main differences between strict and extended interpretations lie in the way they enforce the integrity constraints. When there is no violation of integrity constraints, the semantics of a given contract is equivalent in both interpretations. In this case, the operation always results in the state described in the postcondition. On the contrary, when some integrity constraint is violated, in a strict interpretation the operation is not applied, whereas in an extended one the operation is executed and the constraint violation is repaired.

4. Conclusions and Further Work

The main goal of this paper has been to analyze the semantics of operation contracts in conceptual modeling. In this sense, we have discussed two different interpretations of operation contracts, a *strict* and an *extended* one, which differ on the way to understand the relationship between operation postconditions and integrity constraints.

Roughly speaking, a strict interpretation prevents an operation from being applied if some integrity constraint is violated. On the contrary, an extended interpretation assumes that the operation must take care of maintaining integrity constraints when they are violated as a consequence of applying the events that satisfy the postcondition.

We have provided definitions for the strict and the extended interpretations, independently of any particular modeling language. Thus, our results can be applied to any of them provided that it allows the definition of operations and explicit integrity constraints in the conceptual schema.

Further work may be devoted to formalize the proposed approaches and also to study their application to different kinds of operations, in order to evaluate which one does better in most cases. Also, it would be interesting to apply them to case studies, to see which one is more advantageous in practice.

References

[1] Borgida, A., Mylopoulos, J., Reiter, R.: On the frame problem in procedure specifications. IEEE Transactions on Software Engineering 21(10) (1995) 785-798

[2] Costal, D., Sancho, M. R., Teniente, E.: Understanding Redundancy in UML Models for Object-Oriented Analysis. In: Advanced Information Systems Engineering: 14th International Conference, CAiSE 2002 Proceedings. LNCS 2348 (2002) 659-674

[3] D'Souza, D. F., Wills, A. C.: Objects, Components and Frameworks. The Catalysis Approach. Addison-Wesley (1998)

[4] Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process. 2nd edn. Prentice Hall PTR, Upper Saddle River, NJ (2002)

[5] Martin, J., Odell, J. J.: Object-Oriented Methods. A Foundation. P T R Prentice Hall, Englewood Cliffs, New Jersey (1999)

[6] Meyer, B.: Applying 'Design by Contract'. Computer 25(10) (1992) 40-51

[7] Meyer, B.: Object-Oriented Software Construction. 2nd edn. Prentice Hall, New York (1997)

[8] Olivé, A.: Definition of Events and their Effects in Object-Oriented Conceptual Modeling Languages. In: Conceptual Modeling - ER 2004. LNCS 3288 (2004)

[9] Pressman, R. S.: Software Engineering: A Practitioner's Approach. 5th edn. McGraw Hill (2001)

[10] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modeling and Design. Prentice Hall, Englewood Cliffs, New Jersey (1991)

[11] Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison Wesley Longman, Reading, Massachusetts (1999)

[12] Wieringa, R.: A Survey of Structured and Object-Oriented Software Specification Methods and Techniques. ACM Comput. Surv. 30(4) (1998) 459-527