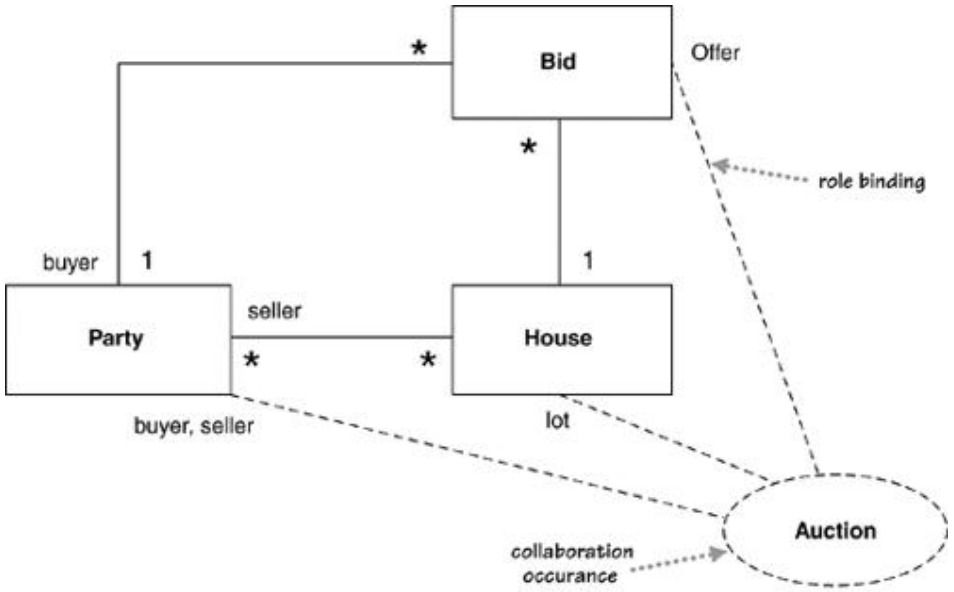
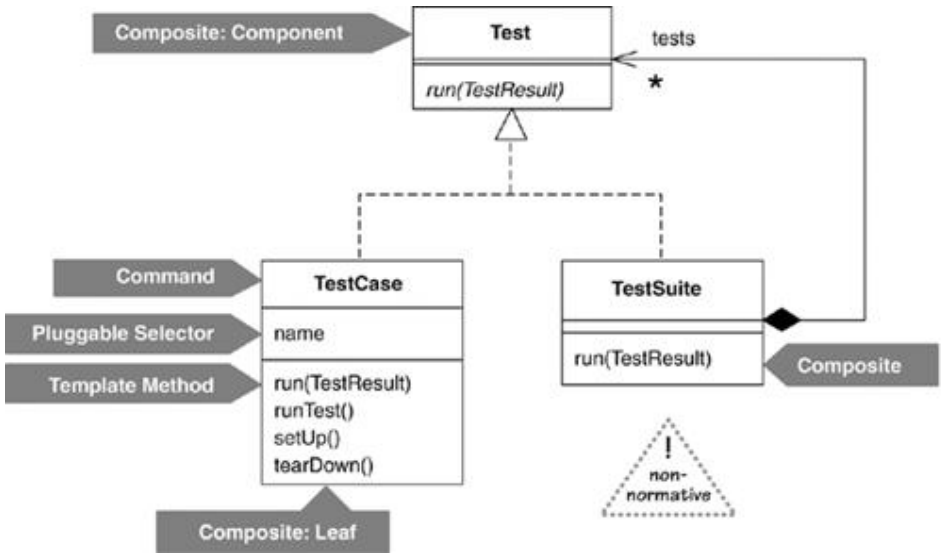


The UML suggests that you can use the collaboration occurrence notation to show the use of patterns, but hardly any patterns author has done this. Erich Gamma developed a nice alternative notation (Figure 15.4). Elements of the diagram are labeled with either the pattern name or a combination of `pattern:role`.

Figure 15.4. A nonstandard way of showing pattern use in JUnit (junit.org)



When to Use Collaborations

Collaborations have been around since UML 1, but I admit I've hardly used them, even in my patterns writing. Collaborations do provide a way to group chunks of interaction behavior when roles are played by different classes. In practice, however, I've not found that they've been a compelling diagram type.

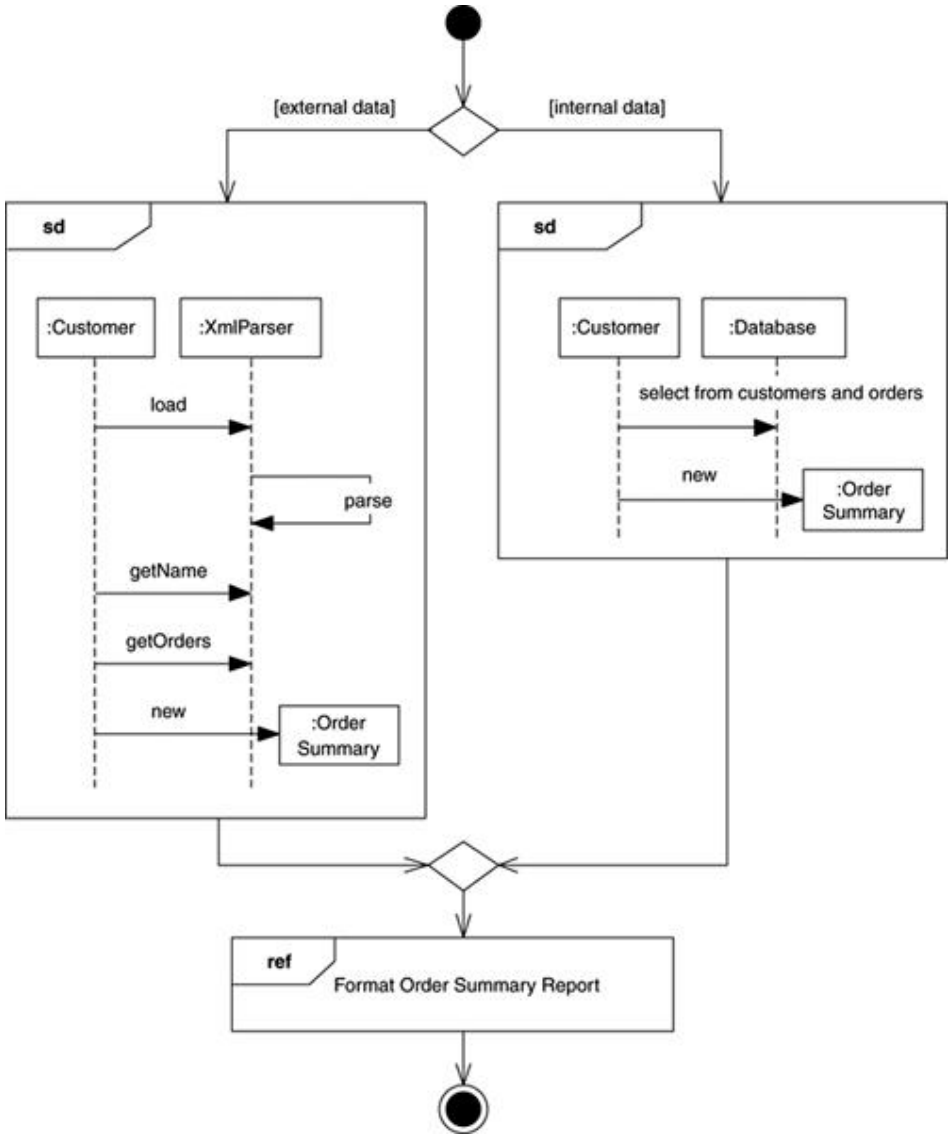
Chapter 16. Interaction Overview Diagrams

Interaction overview diagrams are a grafting together of activity diagrams and sequence diagrams. You can think of interaction overview diagrams either as activity diagrams in which the activities are

replaced by little sequence diagrams, or as a sequence diagram broken up with activity diagram notation used to show control flow. Either way, they make a bit of an odd mixture.

Figure 16.1 shows a simple example of one; the notation is familiar from what you've already seen in the activity diagram and sequence diagram chapters. In this diagram, we want to produce and format an order summary report. If the customer is external, we get the information from XML; if internal, we get it from a database. Small sequence diagrams show the two alternatives. Once we get the data, we format the report; in this case, we don't show the sequence diagram but simply reference it with a reference interaction frame.

Figure 16.1. Interaction summary diagram



When to Use Interaction Overview Diagrams

These are new for UML 2, and it's too early to get much sense of how well they will work out in practice. I'm not keen on them, as I think that they mix two styles that don't really mix that well. Either draw an activity diagram or use a sequence diagram, depending on what better serves your purpose.

Chapter 17. Timing Diagrams

After leaving secondary school, I started out in electronic engineering before I switched into computing. So I feel a certain anguished familiarity when I see the UML define timing diagrams as one of its standard diagrams. Timing diagrams have been around in electronic engineering for a long time and never seemed to need the UML's help to define their meaning. But since they are in the UML, they deserve a brief mention.

Timing diagrams are another form of interaction diagram, where the focus is on timing constraints: either for a single object or, more usefully, for a bunch of objects. Let's take a simple scenario based on the pump and hotplate for a coffee pot. Let's imagine a rule that says that at least 10 seconds must pass between the pump coming on and the hotplate coming on. When the water reservoir becomes empty, the pump switches off, and the hotplate cannot stay on for more than 15 minutes more.

Figures 17.1 and 17.2 are alternative ways of showing these timing constraints. Both diagrams show the same basic information. The main difference is that Figure 17.1 shows the state changes by moving from one horizontal line to another, while Figure 17.2 retains the same horizontal position but shows state changes with a cross. The style of Figure 17.1 works better when there are just a few states, as in this case, and Figure 17.2 is better when there are many states to deal with.

Figure 17.1. Timing diagram showing states as lines

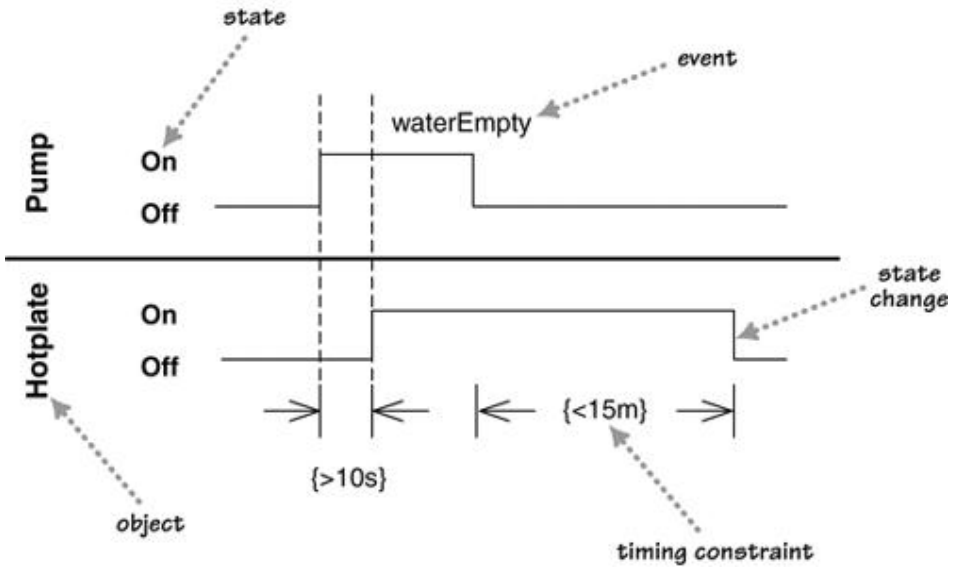
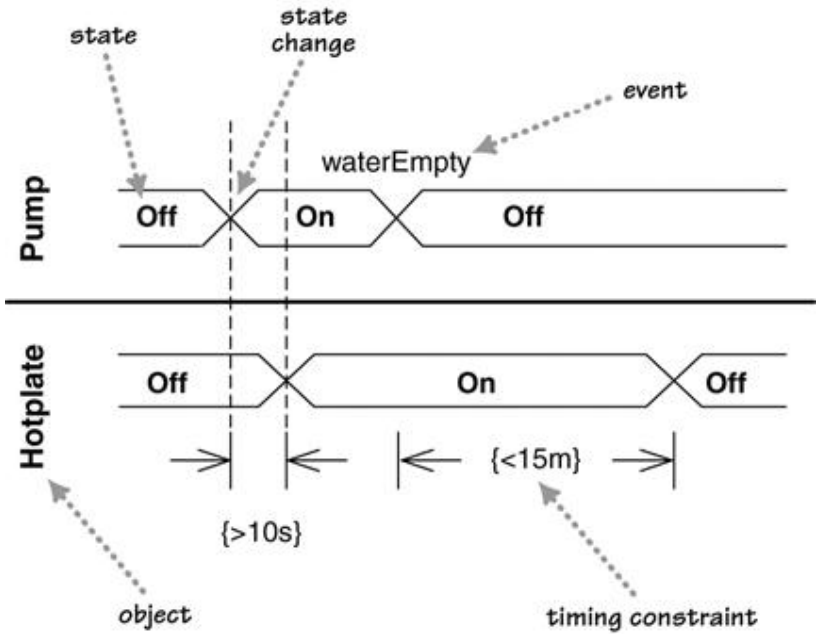


Figure 17.2. Timing diagram showing states as areas



The dashed lines that I've used on the {>10s} constraints are optional. Use them if you think they help clarify exactly what events the timing constrains.

When to Use Timing Diagrams

Timing diagrams are useful for showing timing constraints between state changes on different objects. The diagrams are particularly familiar to hardware engineers.

Appendix Changes between UML Versions

When the first edition of this book appeared on the shelves, the UML was in version 1.0. Much of it appeared to have stabilized and was in the process of OMG recognition. Since then, there have been a number of revisions. In this appendix, I describe the significant changes that have occurred since 1.0 and how those changes affect the material in this book.

This appendix summarizes the changes so you can keep up to date if you have an earlier printing of the book. I have made changes to the book to keep up with the UML, so if you have a later printing, it describes the situation as it was as of the print date.

Revisions to the UML

The earliest public release of what came to be the UML was version 0.8 of the Unified Method, which was released for OOPSLA in October 1995. This was the work of Booch and Rumbaugh, as Jacobson did not join Rational until around that time. In 1996, Rational released versions 0.9 and 0.91, which included Jacobson's work. After the latter version, they changed the name to the UML.

Rational and a group of partners submitted version 1.0 of the UML to the OMG Analysis and Design Task Force in January 1997. Subsequently, the Rational partnership and the other submitters combined their work and submitted a single proposal for the OMG standard in September 1997, for version 1.1 of the UML. This was adopted by the OMG toward the end of 1997. However, in a fit of darkest obfuscation, the OMG called this standard version 1.0. So, now the UML was both OMG