Definitions

Throughout the description of the techniques, the following terms are constantly used:

1. Functionality: *Functionality* describes the behavior of the system. Typically, functionality is described from the user's point of view. That is, a description of system functionality should answer the question: What can a user use the system to do? In the case of a word processor, an example of system functionality is formatting text.

2. Service: Like "functionality", a *service* of the system is an action performed by the system. However, services are much more low-level; they are the "atomic units" out of which system functionalities are composed. Users do not typically consider services an end in themselves; rather, services are the steps by which some larger goal or functionality is achieved. In the case of a word processor, typical services include selecting text, using pull-down menus, and changing the font of a selection.

3. Message: *Messages* are the very lowest-level behaviors out of which system services and, in turn, functionalities are composed. They represent the communication between objects that work together to implement system behavior. Messages may be shown on sequence diagrams and must have associated class behaviors.

For example, consider the example diagrams provided in the appendix D. In example 2, the sequence diagram describes how classes collaborate to provide some *functionality*: the ability to lease a parking spot. This functionality is meant to describe a use of the system from the user's point of view; although the user may have to perform several steps in his interaction with the system, we expect that his or her final goal is the lease of a spot to park his car.

Two *services* are marked on the diagram, represented by the heavy dashed and solid lines, which group together a collection of *messages*. These services represent particular steps that must be accomplished for the user to achieve the task of purchasing the parking spot. The dashed grouping may be thought of as the service of "getting an open spot" while the grouping circled by the solid line accomplishes the step of "paying for the spot at the time of leasing." To the user, neither step makes sense as a goal in and of itself; e.g. it is of little use to the customer to find an open spot but not pay for it.

It should be noted that there may be multiple ways to group messages together into services. The messages lease_parking_spot, add_to_bill, and new_purchase may be grouped to compose a service that can be thought of as "paying for a spot via monthly bill." Each of these services represents a different execution path the system will follow under different conditions, and thus all are necessary to describe the full range of system functionality. In some cases, the designer may choose to use a number of similar sequence diagrams, with each diagram showing one such execution path, in order to avoid the complexity of many services being represented on the same diagram, as is the case in Example 2.