Collaboration Diagram

Collaboration diagrams show objects, their links, and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model. These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object

Sequence diagrams are closely related to collaboration diagrams and both are alternate representations of an interaction. There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other.

To demonstrate this close relationship, we will translates a sequence diagram into a collaboration diagram. This sequence diagram is one of the design documents created for a research and development library. This particular sequence, shown in Figure 1, documents the interaction that occurs between business objects when determining how many items a borrower can check out of the library.

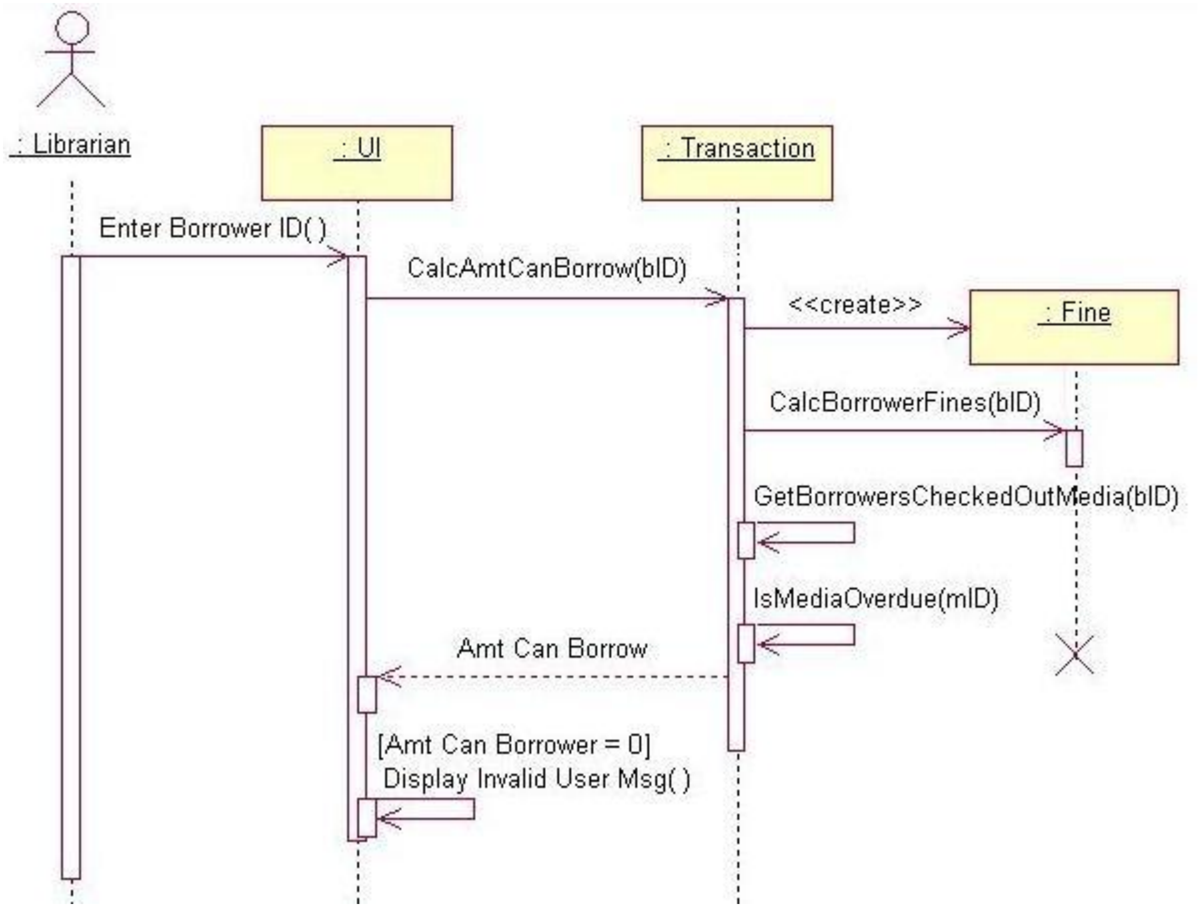

Figure 1: A sequence diagram is ideal for showing the time-ordering of messages during an object interaction.

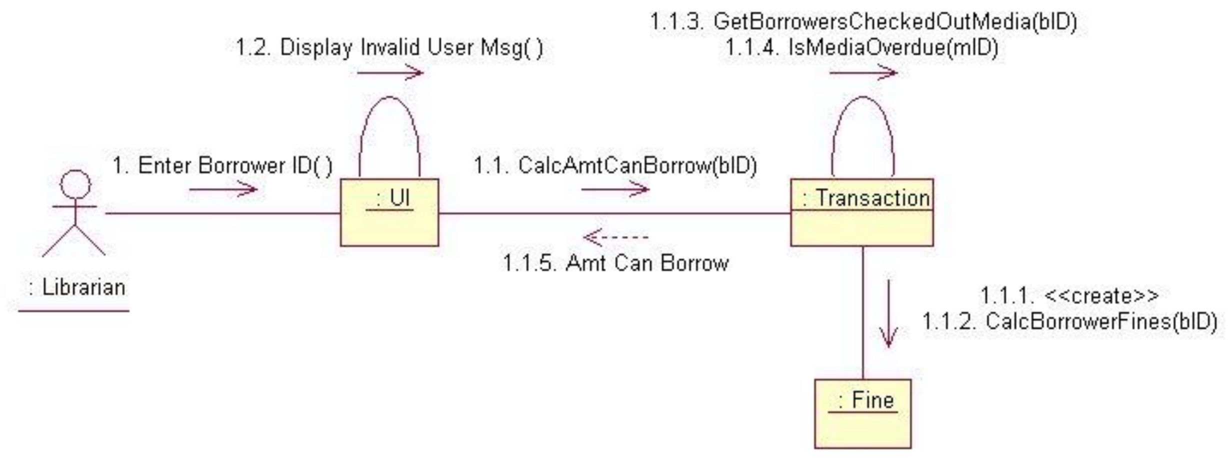
1.2. Display Invalid User Msg( )
1.1.3. GetBorrowersCheckedOutMedia(bID)
1.1.4. IsMediaOverdue(mID)
1. Enter Borrower ID( )
: UI
1.1. CalcAmtCanBorrow(bID)
: Transaction
1.1.5. Amt Can Borrow
: Librarian
1.1.1. <<create>>
1.1.2. CalcBorrowerFines(bID)
: Fine

Figure 2: A collaboration diagram is best suited for showing the relationships between objects involved in a collaboration.

Collaboration Diagram Elements

There are three primary elements of a collaboration diagram:

- Objects
- Links
- Messages

Objects

Objects participating in a collaboration come in two flavors?supplier and client. Supplier objects are the objects that supply the method that is being called, and therefore receive the message. Client objects call methods on supplier objects, and therefore send messages.

In **Figure 2**, the Transaction object acts as a Supplier to the UI (User Interface) Client object. In turn, the Fine object is a Supplier to the Transaction Client object.

Links

The connecting lines drawn between objects in a collaboration diagram are *links*. These links are what set collaboration diagrams apart from sequence diagrams. They enable you to see the relationships between objects. Each link represents a relationship between objects and symbolizes the ability of objects to send messages to each other. A single link can support one or more messages sent between objects. This is different from sequence diagrams, where the lines drawn between objects represent messages sent from one object to another.

As you can see by looking at **Figure 2**, the visual representation of a link is a straight line between two objects. If an object sends messages to itself, the link carrying these messages is represented as a loop icon. This loop can be seen on both the UI object and the Transaction object.

Messages

Messages in collaboration diagrams are shown as arrows pointing from the Client object to the Supplier object. Typically, messages represent a client invoking an operation on a supplier object.

Message icons have one or more messages associated with them. Messages are composed of message text prefixed by a sequence number. This sequence number indicates the time-ordering of the message. For example, in the collaboration diagram in **Figure 2**, you can follow the sequence numbers to determine the order of messages between objects:

1. Enter Borrower ID

1.1 CalcAmtCanBorrow

1.1.1 <<create>>

1.1.2 CalcBorrowerFines

1.1.3 GetBorrowersCheckedOutMedia

1.1.4 IsMediaOverdue

1.1.5 Amt Can Borrow

1.2 Display Invalid User Msg

The first message in a collaboration diagram is always numbered 1, the second is 2, and so on. You can indicate that a message is nested under a parent message by adding a decimal point and incremental digits to the parent's sequence number. For example, the "CalcAmtCanBorrow" message is the first nested message under "Enter Borrower ID" and is given the sequence number 1.1. The second nested message under "Enter Borrower ID" is "Display Invalid User Msg," so it's given a sequence number of 1.2. As you can see, there are several messages nested under "CalcAmtCanBorrow" and these are numbered 1.1.1 through 1.1.5.

Iterating and Conditional Messages

Collaboration diagrams use syntax similar to sequence diagrams to indicate that either a message iterates (is run multiple times) or is run conditionally.

You can indicate that a particular message iterates by prefixing a message sequence number with an iteration expression. You can simply use an asterisk (*) to indicate that a message runs more than once, or you can get more specific and show the number of times a message is repeated (for example, 1..5).

To indicate that a message is run conditionally, you can prefix the message sequence number with a conditional clause such as [x = true]. This indicates that the message is sent only if the condition is met. The UML leaves the syntax of conditional clauses wide open, so you can create expressions that make sense in the context of your application.

Creation and Deletion of Objects

Unlike sequence diagrams, you don't show an object's lifeline in a collaboration diagram. If you want to indicate the lifespan of an object in a collaboration diagram, you can use create and destroy messages to show when an object is instantiated and destroyed.

For example, in **Figure 2**, there is a 1.1.1 <<create>> message before the 1.1.2 message call to the Fine object. This indicates that the Transaction object instantiates the Fine object before calling its CalcBorrowerFines() method.

Collaboration vs. Sequence Diagrams

As you follow the sequence of messages in **Figure 2**, you can definitely see why the time-ordering of messages is *not* the strong suit of collaboration diagrams! In fact, messages on sequence diagrams do not even need sequence numbers, because the order in which messages occur is made obvious by the physical layout of messages from top to bottom in the diagram.
That said, collaboration diagrams have a distinct advantage over sequence diagrams in that they allow you to show more complex branching as well as multiple concurrent flows of control. In contrast, the format and nature of sequence diagrams really only allow you to show simple branching.

Summary

Although collaboration diagrams are not used as often as sequence diagrams, they are a very useful part of the UML. If you find yourself flipping back and forth between sequence diagrams (dynamic view) and associated class diagrams (static view) to try to get a handle on the associations between business objects, then you may want to try a collaboration diagram instead. It allows you to see both the dynamic aspects of a collaboration as well as the relationships between objects, in a single diagram.