Validation (2): Collaboration Diagrams

CS3: SEOC1

Note 7

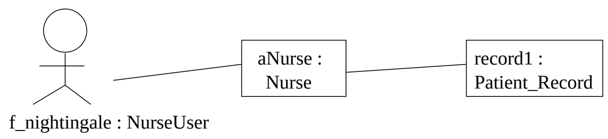
Interaction Diagrams

UML interaction diagrams refine the kind of activity undertaken in checking with CRC cards. They document in detail how the classes realize the use cases. There are two different kinds of interaction diagrams: collaboration diagrams *and* sequence diagrams. *They document the same information in different ways.*

Collaboration Diagrams

- We consider the instance forms of the diagrams
- There are generic forms of diagram and extensions to deal with concurrency
- There is redundancy between collaboration and sequence diagrams
- A collaboration is a collection of named objects and actors with links connecting them. They collaborate in performing some task.

Collaborations



Actors: Each is named and has a role. One actor will be the *initiator* of the use case.

Objects: Each is named and has its class specified. Not all classes need appear. There may be more than one object of a class.

Links: Links connect objects and actors and are instances of associations. We cannot insert a link if there is no association in the class diagram.

Interactions

- Here we add the message sent along the link. This is driven from the use case and some record of the class diagram.
- The message is directed from sender to receiver.
- The receiver must understand the message.
- The association must be *navigable* in that direction

Flow of Control

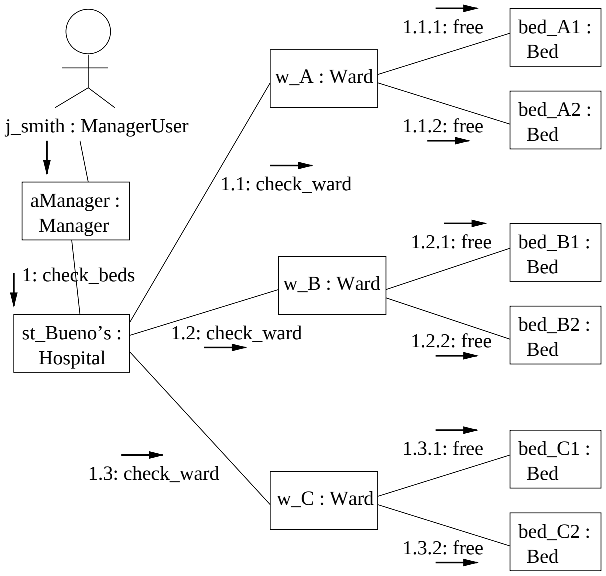
Procedural interactions: At most one object is computing at any time.

Activation: An object has a live *activation* from when it receives a message until it responds to the message.

Waiting for response: *Synchronous* messages: on sending a message to another object, an object will wait until it receives a response.

Activation stack: Activations are stacked and the top activation has control. When the top action responds the next to top regains control and so on ....

Collaboration Diagram for HIS



Law of Demeter:

Where should O be able send messages in dealing with message m?

1. to itself.
2. objects sent as argument in message m
3. objects O creates in responding to m
4. objects that are directly accessible from O, using attribute values.