COMP201 Software Engineering I
Lecture 23 – UML Interaction Diagrams

Lecturer: Dr T. Carroll

Email: Thomas.Carroll2@liverpool.ac.uk

Office: G.14

See vital for all notes

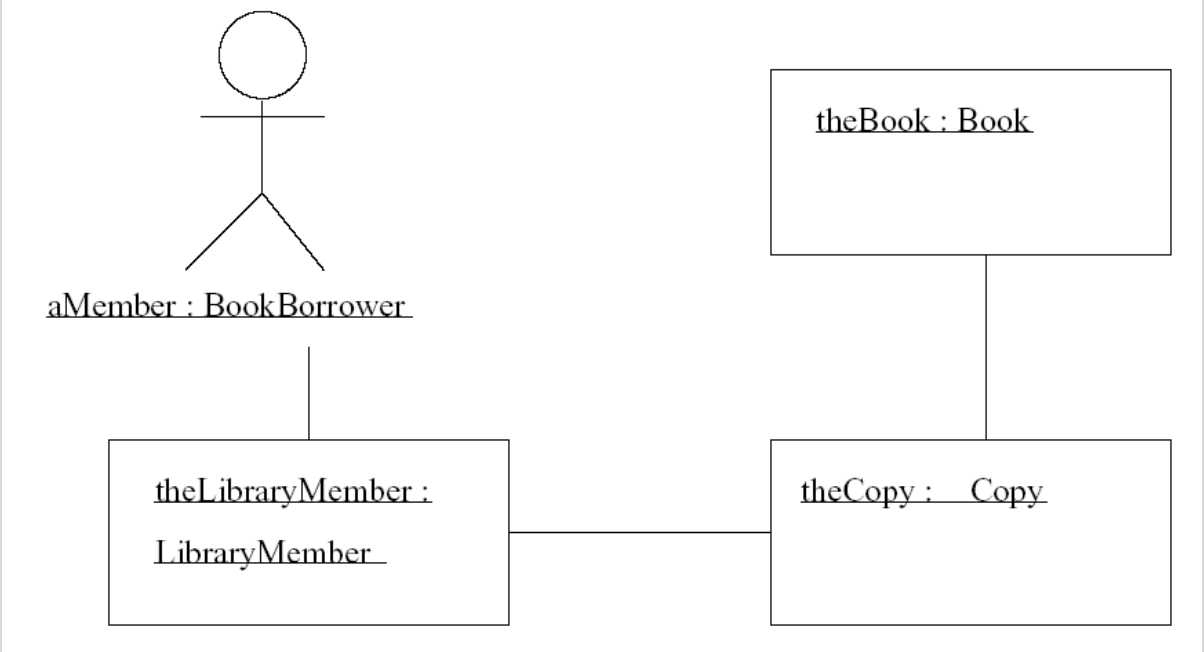
Today…

Important UML Models

- We have now seen the two most important UML models:
  - The ***use case model***, which describes the tasks which the system must help to perform
  - The ***class model***, which describes the classes which are to be implemented and the relationships between them
- UML's ***interaction diagrams*** allow us to record, in detail, how objects interact to perform a task
  - Sequence Diagrams
  - Collaboration Diagrams
  - Activity Diagrams

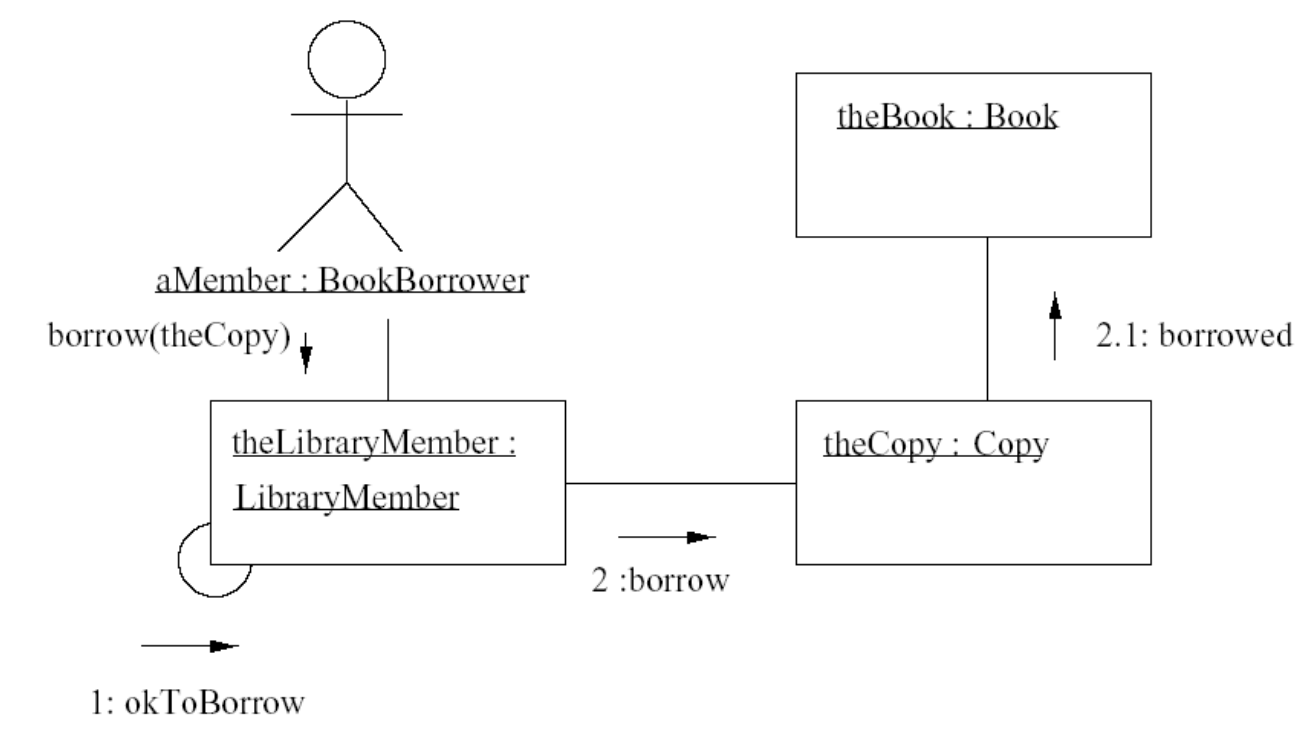
Collaborations – Objects linked to perform a task

- **Objects** - Each object is shown as labelled rectangle (name:Type)
- **Links** - Links between objects are shown like associations in the class model.
- **Actors** - Actors can be shown as on a use-case diagram



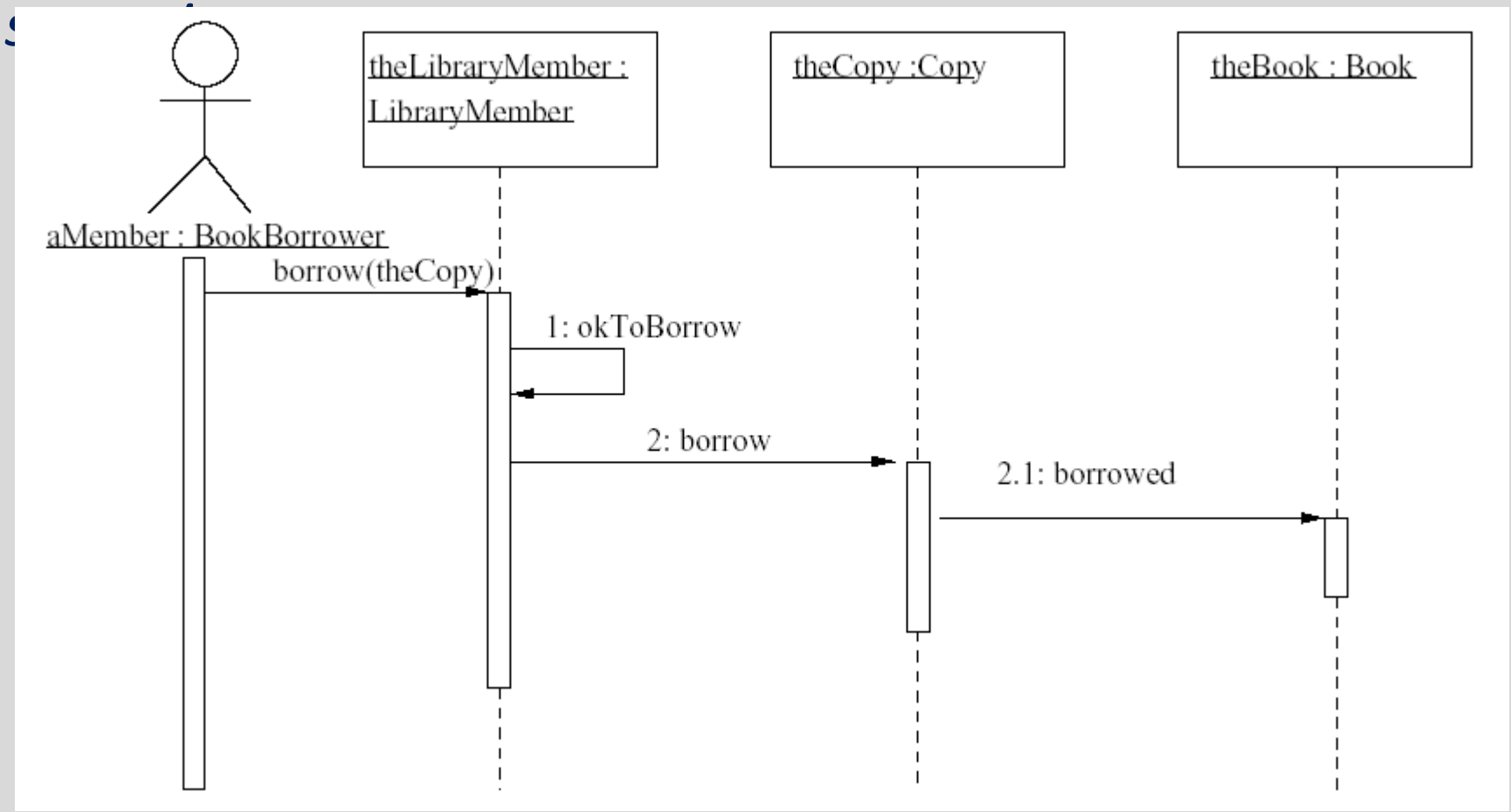
Interaction on Collaboration Diagrams

- Each labelled arrow represents a message being sent
- The target object must understand the message



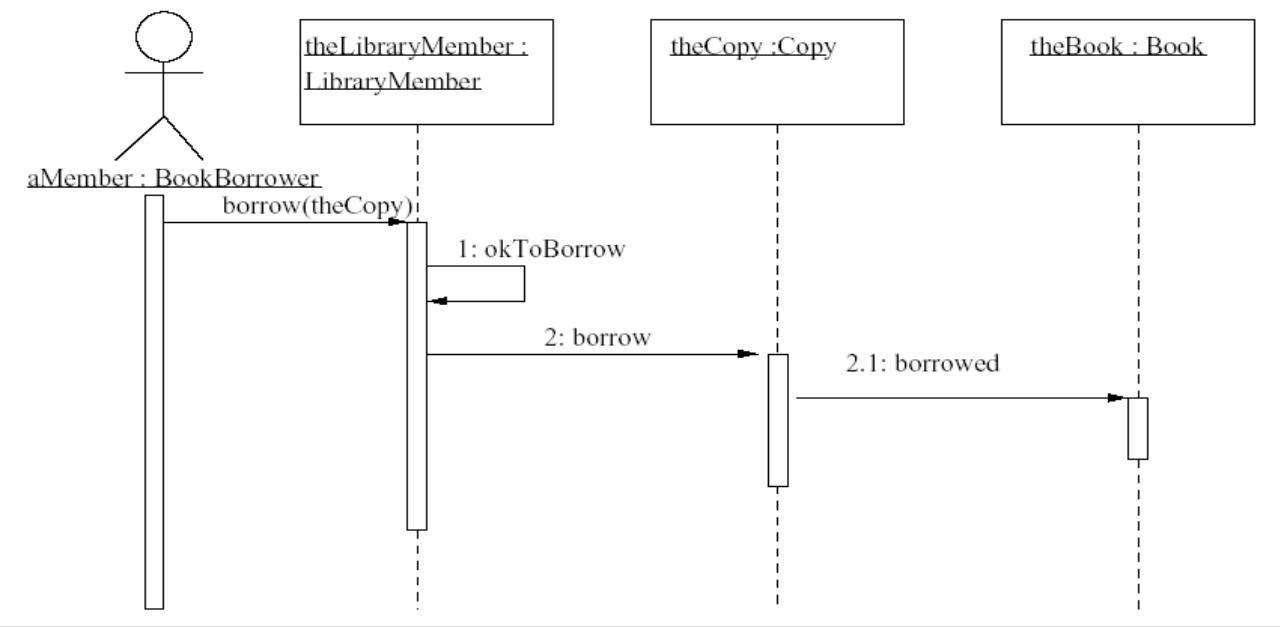
Sequence Diagrams

- **Sequence diagrams** are applicable to modeling *real-time interactive systems* or *complex s*



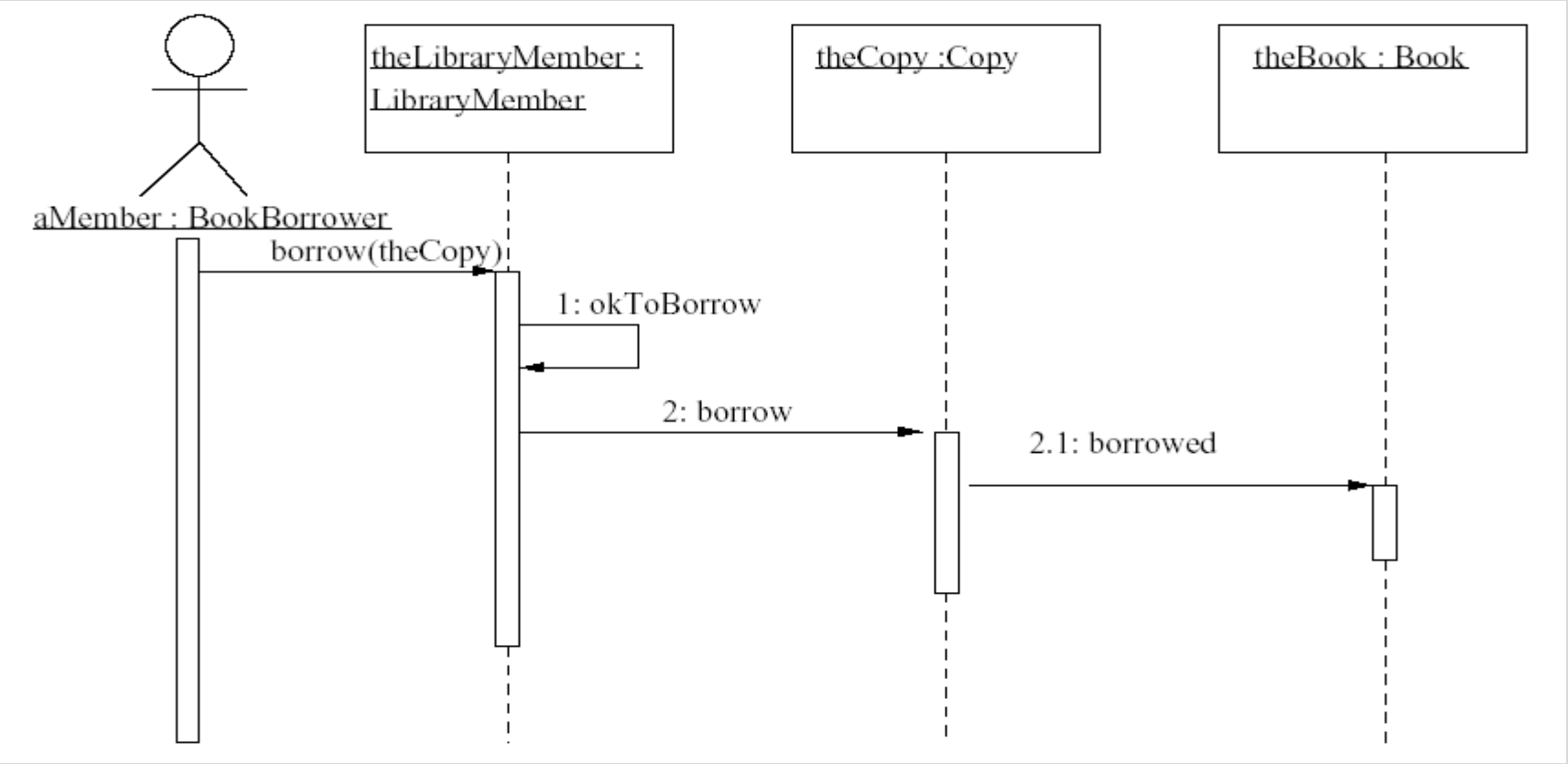
Sequence Diagram

- A ***sequence diagram*** shows the objects and actor which take part in a collaboration at the top of dashed
- The *vertical dimension* of a sequence diagram represents ***time***
- The *horizontal dimension* represents the different ***objects*** or roles that participate in the interactive sequence.
- An object's ***lifeline*** is shown as a *narrow vertical bar*.



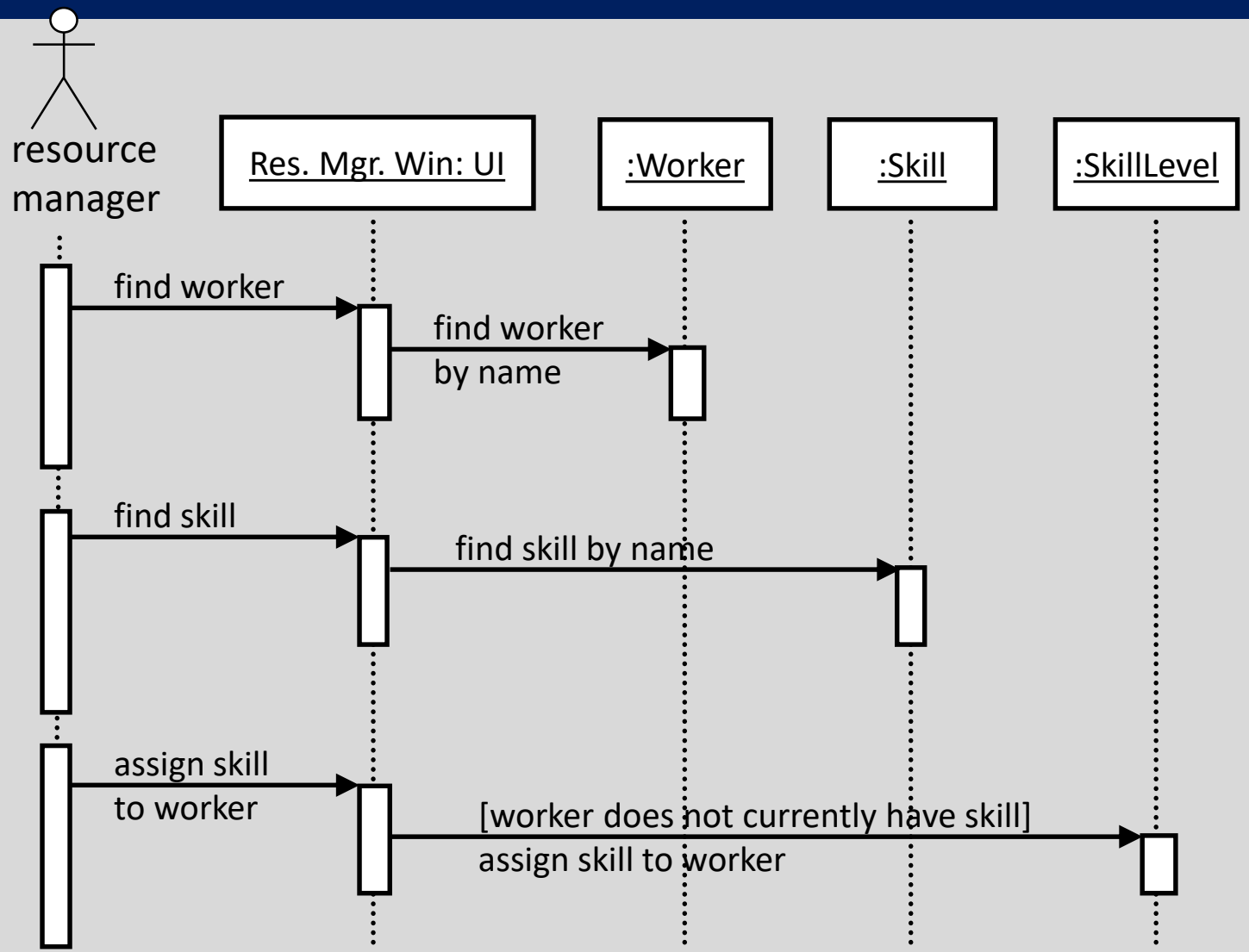
And Time Goes By….

- ***Time*** is assumed to pass as we move from the top to the bottom of the diagram.
- ***Messages*** between objects are shown as solid line arrows, and their ***returns*** are shown as *dashed line* arrows.



Task

1) List all the pairs of classes that can communicate directly with each other.
2) For each class, list all the method signatures that need to be included, based on this sequence diagram



Messages from an Object to Itself

- An object may, and frequently does, send a message to itself
  - (i.e. An object calls another method on itself; Java uses keyword “this”).
- On a collaboration diagram you show a link from the object to itself, and messages pass along that link in the usual way
- On a sequence diagram, you show a message arrow from the object’s lifeline back to itself.

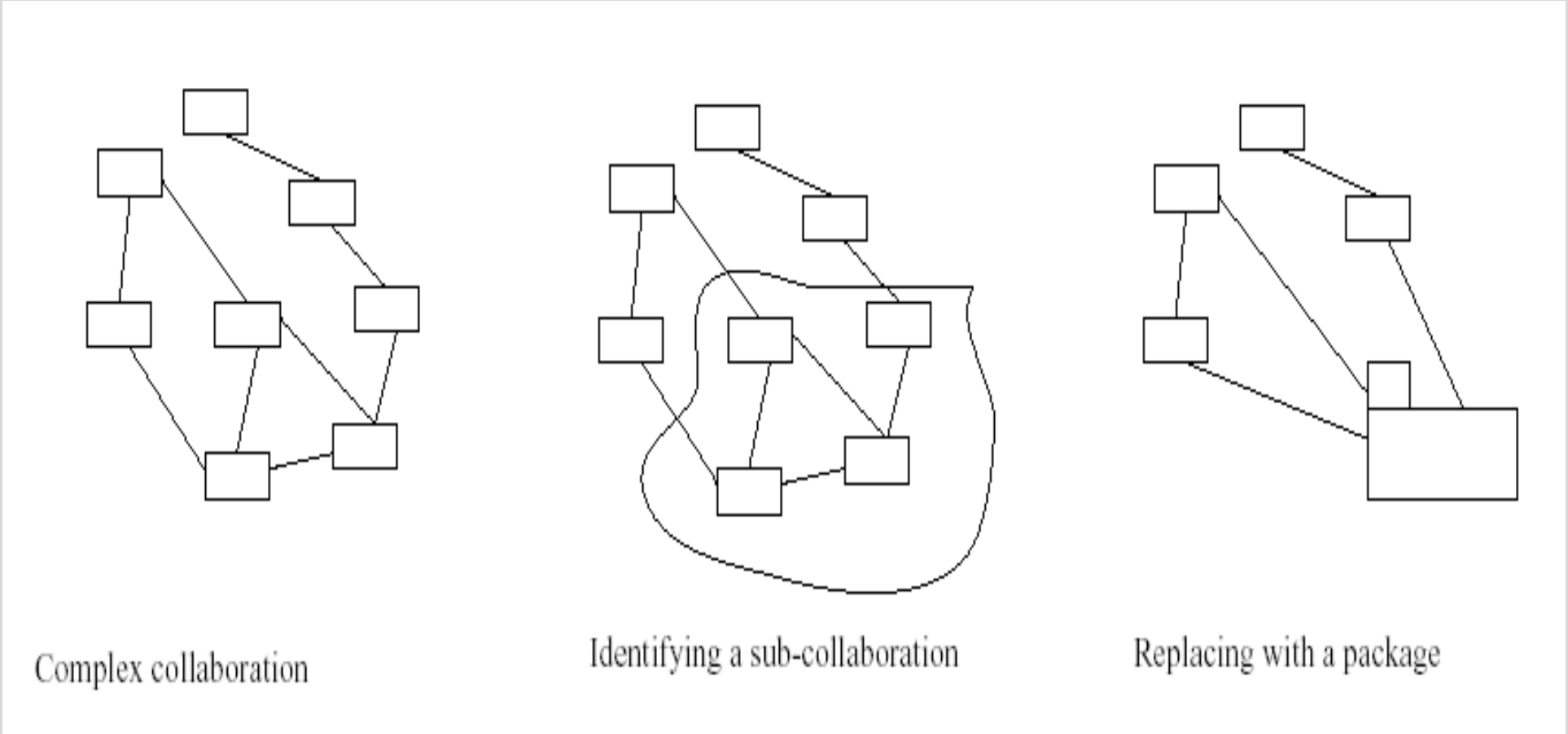
Messages from an Object to Itself

- In pure object oriented programming,
  - every function invocation is the result of a **message**
  - objects may send messages to themselves so often that an interaction diagram becomes cluttered
- You might choose to omit messages from an object to itself, counting such things as *internal computation* within the object.
  - This is a type of **abstraction**.

Suppressing Detailed Behaviour

- It is often sensible to describe interaction at a higher level, rather than showing every message between every pair of objects.
- To do this we define a (full) sub-collaboration of a collaboration
  - Collaboration is a collection of objects and links between them
  - Sub-collaboration is a subset of the objects, together with the links connecting those objects.

Using a Package to Simplify a Collaboration



Complex collaboration

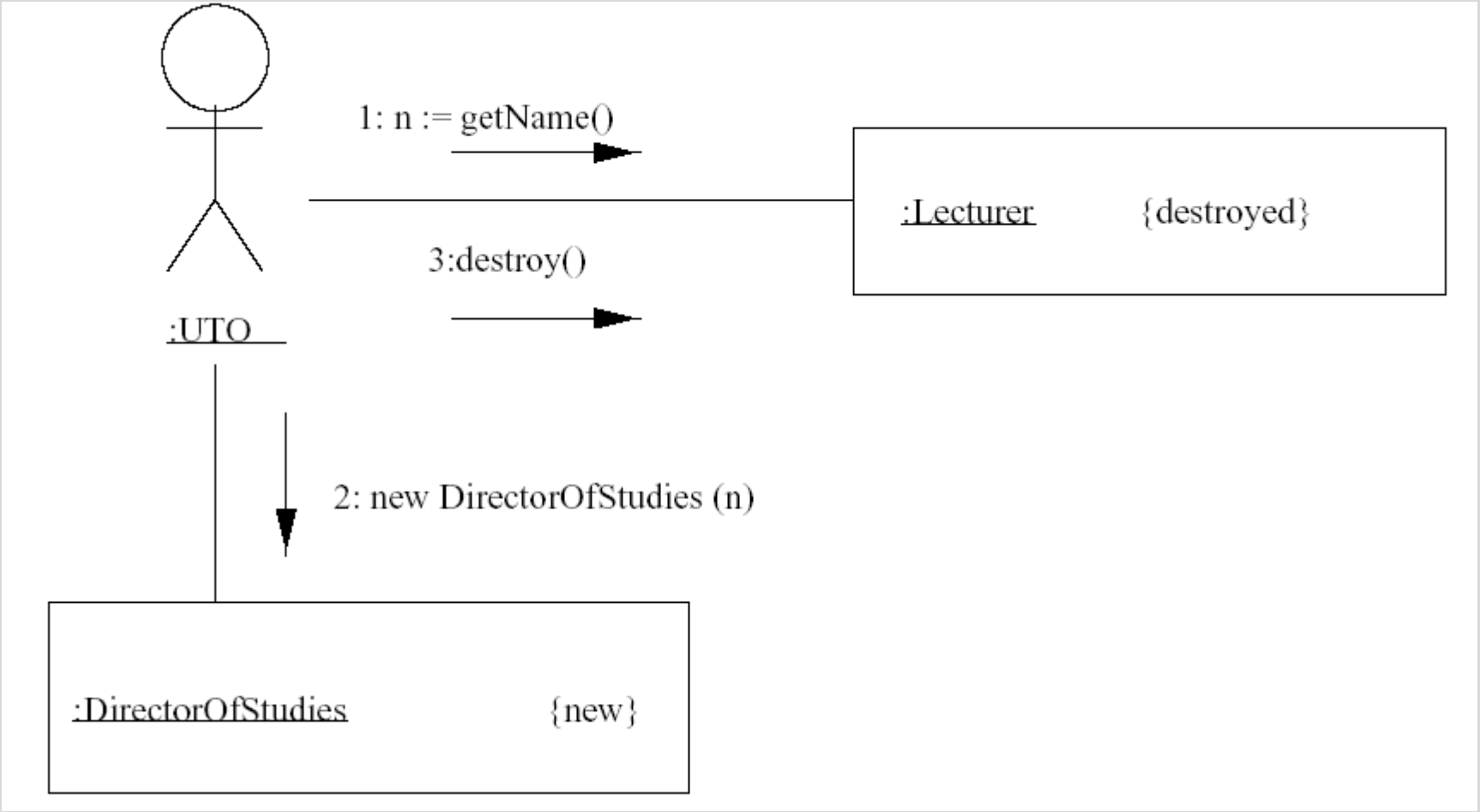
Identifying a sub-collaboration

Replacing with a package

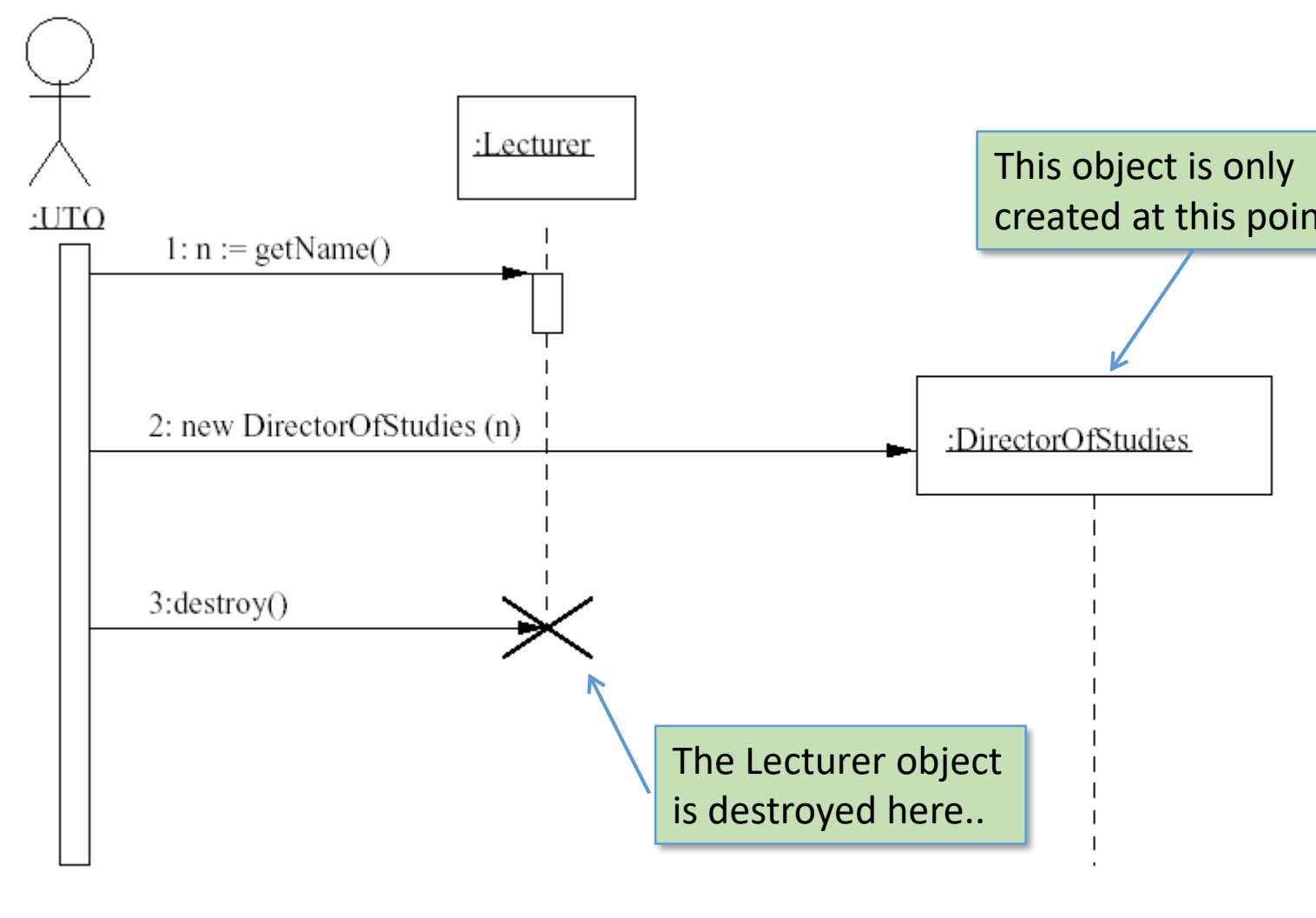
Creation and Deletion of Objects

- The set of objects involved in an interaction is not always static
  - objects may be *created* and *deleted* during an interaction.
- Collaboration diagram
  - Show which objects are created and destroyed during an interaction by {new} {destroyed}.
  - If the object is both created and destroyed in the same interaction, it can be labelled {transit}
- Sequence diagram
  - These show an object being created by putting its object box part-way down the page
  - Destruction of an object is shown by its activation ending with a large X.

Example Collaboration Diagram



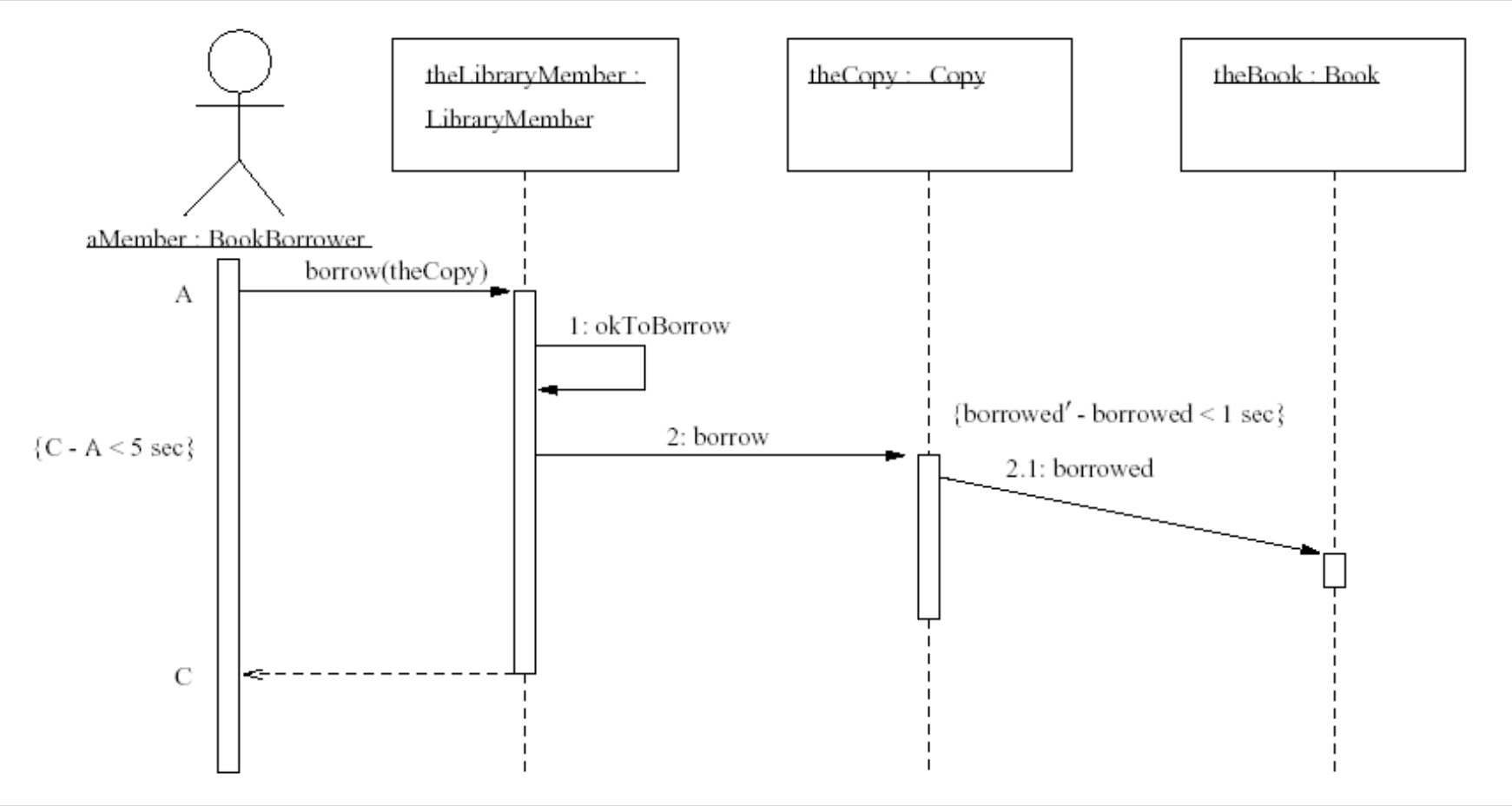
Example Sequence Diagram



Timing

- The major advantage of sequence diagrams over collaboration diagrams is their ability to represent the passage of time graphically.
- So far we have let the diagram indicate only the relative ordering of messages.
- Sometimes the actual times are important (eg: in real time systems).

Timing Constraints on a Sequence Diagram



Activity Diagrams

- Activity diagrams describe how activities are coordinated.
  - An activity diagram may be used **(like an interaction diagram)** to show how an operation could be implemented
- An activity diagram is particularly useful:
  - when you know that an operation has to achieve a number of different things, and
  - you want to model what the essential dependencies between them are, before you decide in what order to do them
- Activity diagrams are much better at showing this clearly than interaction diagrams.

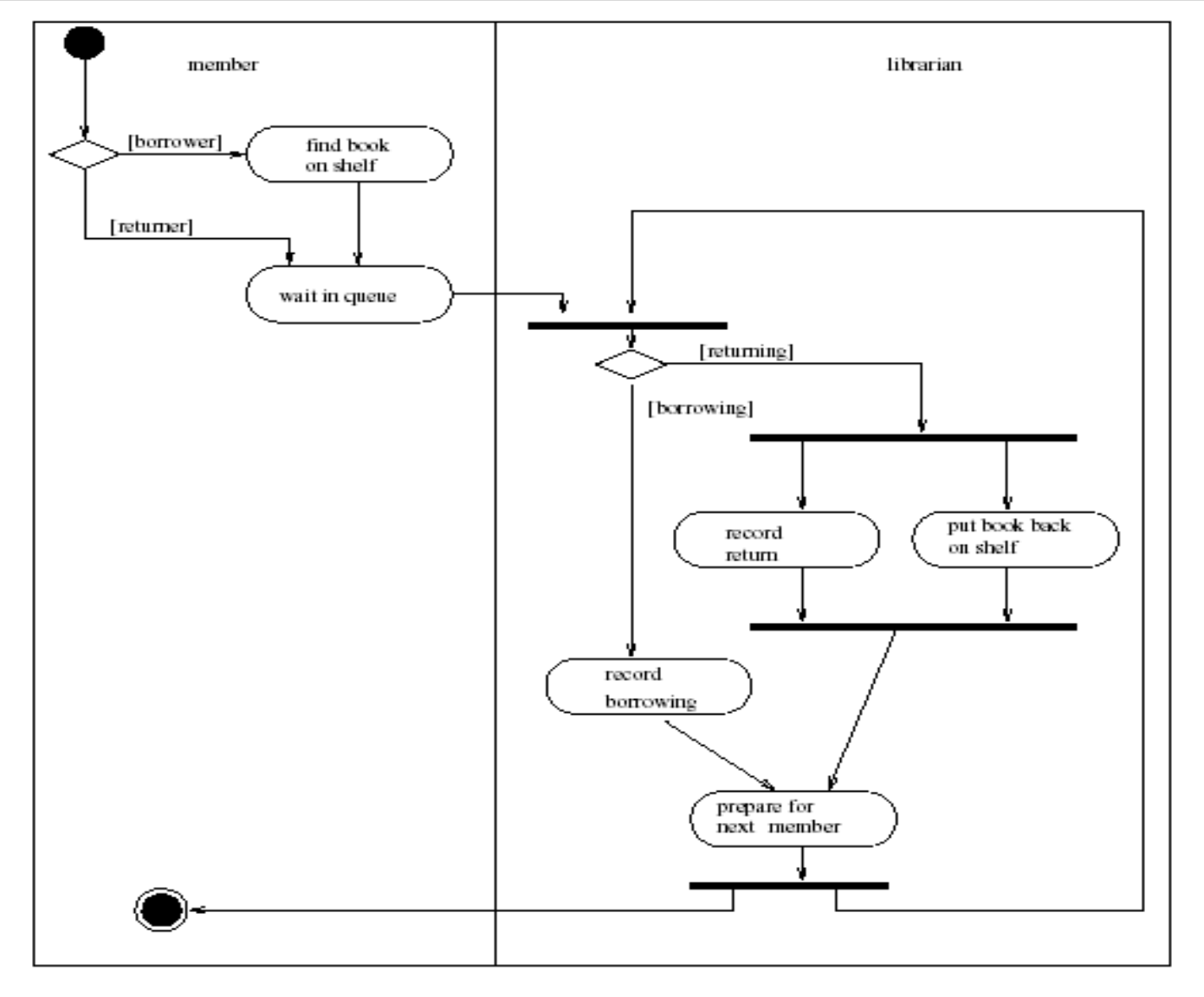
Activity Diagrams

- Some **use-cases** must be completed before others can begin.
- Recording of this can be aided by using an activity diagram
- Activity diagrams record the dependencies between activities:
  - which things can happen in parallel?
  - which must occur sequentially?
- UML activity diagrams are **state diagrams** with extra notation
  - Activity
  - Transition
  - Synchronization bar
  - Decision diamond
  - Start and stop markers

Activity Diagrams

- **Activity** – An activity is recorded like the notation for a state. However, we do not have transitions as a result of event, rather as the finishing of an activity.
- **Activity edge** – an arrow to indicate where to move in the diagram after an activity finishes. These can be labelled with a guard condition.
- **Synchronisation bar** – a thick horizontal bar describing the co-ordination of activities which must all be completed before the activity edges leading from the bar are fired.
- **Decision Diamond** – can be used to show decisions as an alternative to guards on separate states leaving the same activity.
- **Stop/Start markers** – are used in the same way as in a state diagram (initial/final states).

Business Level Activity Diagram of a Library



Activity Diagrams and State Diagrams:

- Differences between activity diagrams and state diagrams:
  - Activity diagrams do not normally include **events**
  - Activity is intended to proceed, following the flow described by diagram, without getting stuck.
  - Thus usually one of the guards of an edge leaving an activity should be satisfied
  - Concurrent activities can be modelled by using the synchronisation bar notation.

Lecture Key Points

- We have seen Collaboration Diagrams and Interactions on them.
- We have also studied sequence diagrams to represent the passage of time graphically and timing constraints that may be imposed upon them.