

Unified Modelling Language

- visual language
- analysis and design using object-oriented approach
- visualise
- construct
- document
- it is not a methodology

Modelling

- Why model ?
 - visualise system
 - structure of system
 - behaviour of system
 - construction template
 - record/document

Modelling

“A model is a simplification of reality”

“We build models so that we can better understand the system we are developing”

“We build models of complex systems because we cannot comprehend such a system in its entirety”

Principles

- choice of model - effects approach to problem and solution obtained
- models support different levels of precision
- good models are connected to reality
- no single model is sufficient
- number of separate models needed

Principles

- abstraction
 - focus on relevant properties of system
- relationships
 - structural, inter-dependency of elements
 - organisational, packaging of elements
 - temporal, sequence of events
 - cause and effect, pre-conditions and results
 - evolutionary, inter-relationships between models

Overview

- visualising
- specifying
- constructing
- documenting

UML Diagrams

- activity - operation as a set of actions
- class - static structure
- collaboration - objects links and interactions
- component - physical components
- deployment - map components to hardware
- object - objects and relationships

UML Diagrams

- sequence - temporal behaviour of objects
- statechart - behaviour in a particular state
- use case - function from user's viewpoint

Class Diagram

- static structure
- relationship between classes
- attributes (data)
 - initial value, =
 - derived attribute, /, attached note
 - multiplicity, [m..n]
 - default 1

Class Diagram

- operations (method)
 - parameters, parameter name:type=value
 - in (value), out, inout (reference)

Window
origin size
open() close()

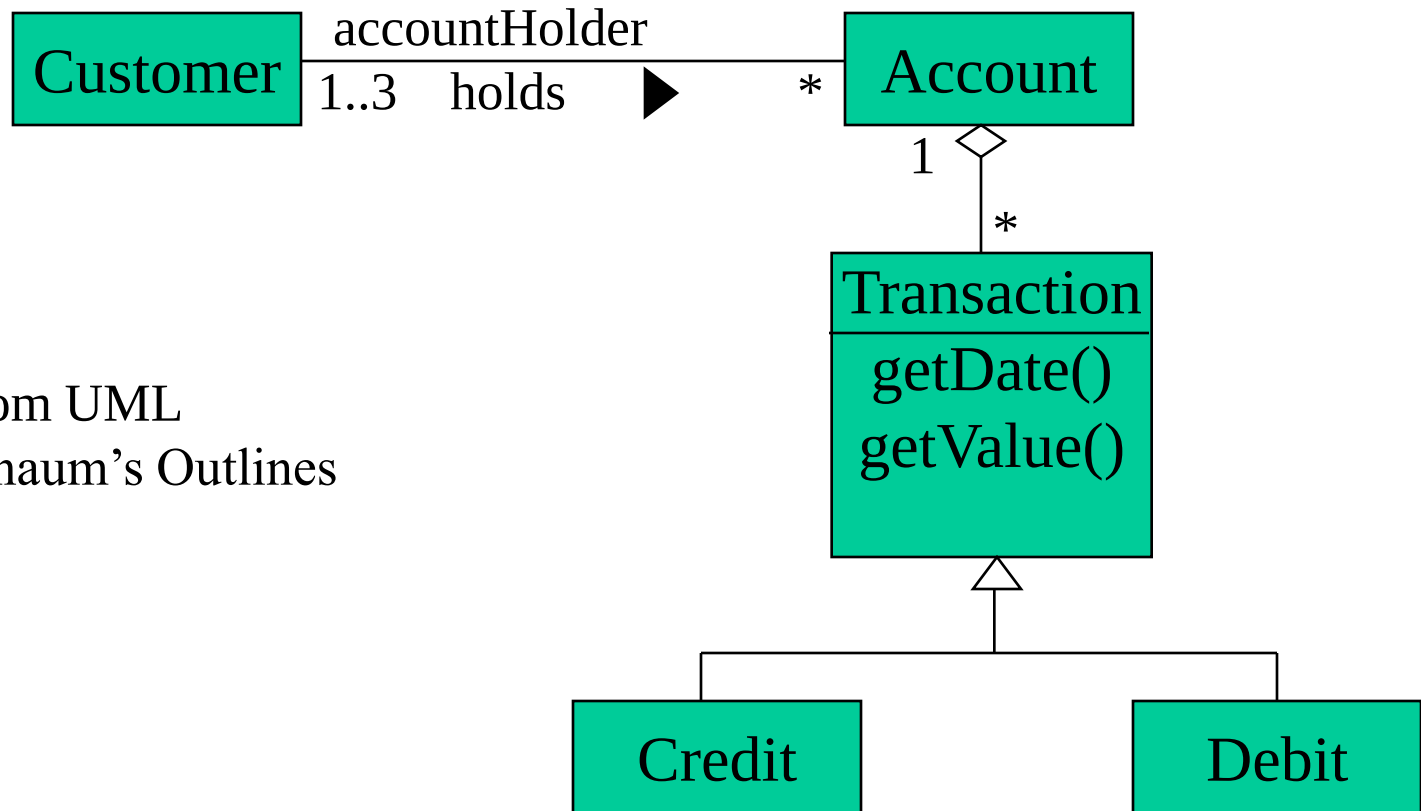
Class Diagram

- relationships with other classes
 - association
 - solid line
 - arrow head
 - label (verb)
 - multiplicity
 - number of object instances, m..n
 - default, not specified
 - role names
 - clarify role of class in association

Class Diagram

- aggregation
 - class includes instances of another class
 - unfilled diamond connected to aggregate
- composition
 - coincidental aggregation
 - filled diamond
- generalisation
 - “kind of” relationship
 - unfilled arrow head

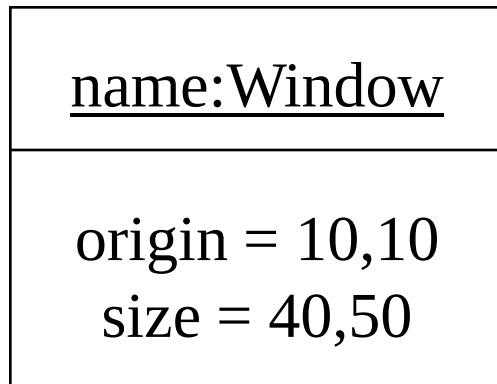
Class Diagram



From UML
Schaum's Outlines

Object Diagrams

- static structure
- snapshot of instances
- show context



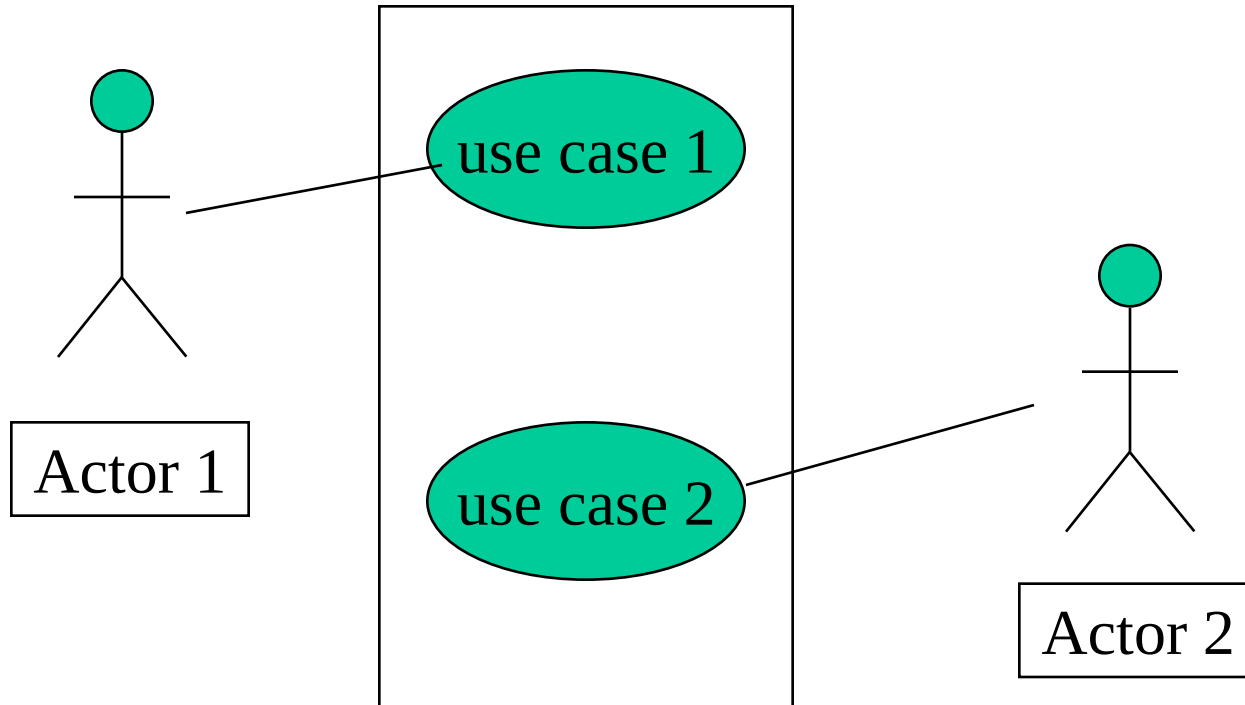
Use Cases

- system from user's standpoint
- relationship between system and environment
- image of system's functionality
- actor represents user or other system e.g. external hardware
- analysis technique

Use Cases

- sequence of actions carried out by system with an observable outcome
- scenarios within use case
- labels
 - active verb and noun
 - noun phrase
- actor
 - role performed
 - many physical entities

Use Cases



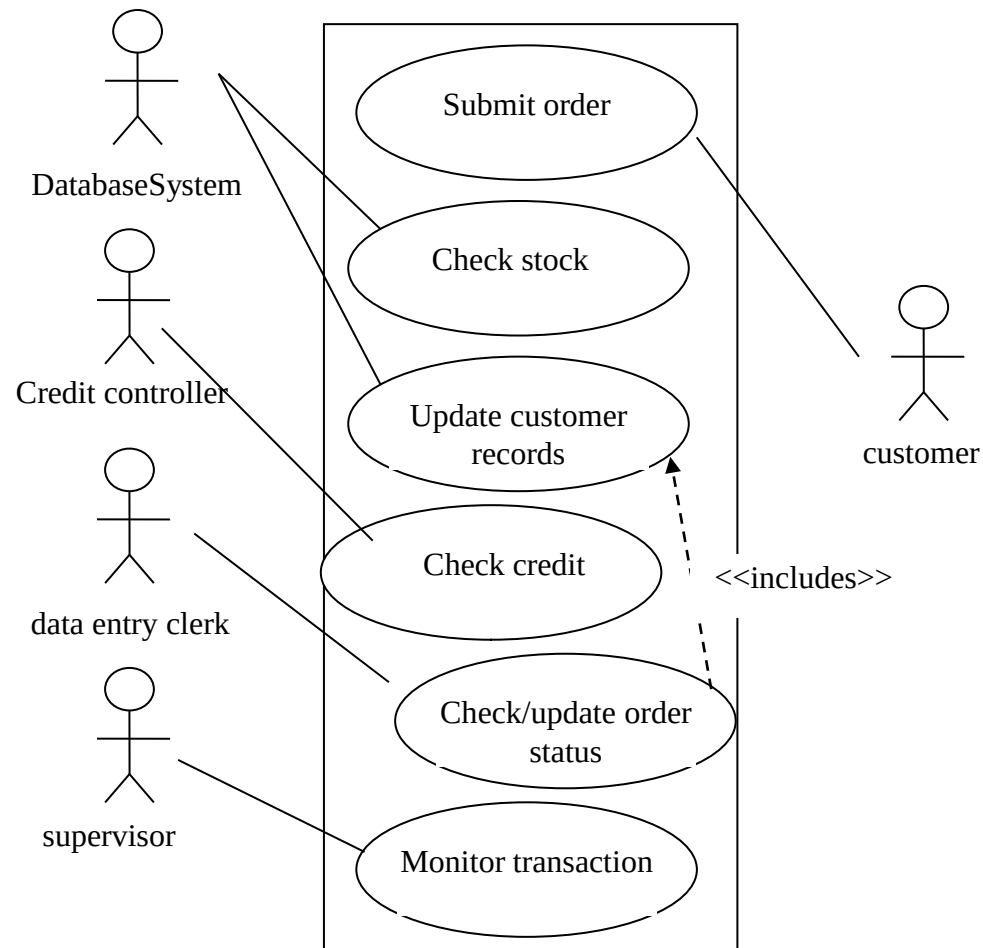
Use Cases

- textual description, use case description
 - describe sequence of activities
 - activity of actor, response of system
- associations
 - generalisation, use cases and actors
 - shared actions
 - unfilled arrow head
 - abstract use case, italicised

Use Cases

- relationships
 - include
 - functionality of one included in another use case
 - dashed line, open arrow, guillemets <<...>>
 - extend
 - optional inclusion

On-line book store



Interaction Diagrams

- Interaction diagrams show the flow of control and data among the elements of the system.
- Collaboration diagram (also known as Communication diagram in UML 2)
- Sequence diagram

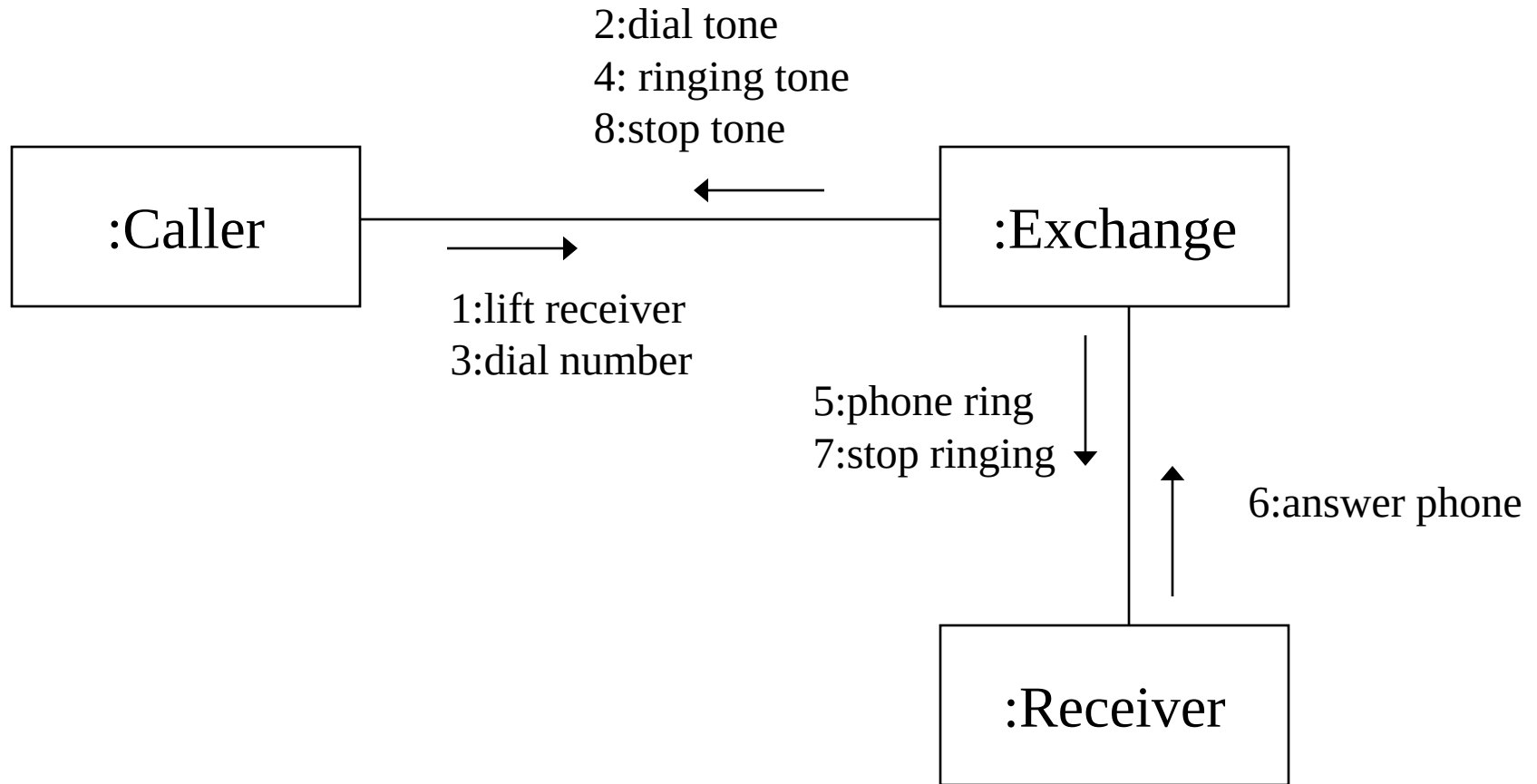
Collaboration Diagrams

- illustrate the relationship and interaction between classes/objects, basic associations (relationships) between classes
- show the message flow between objects
- focus on object roles
- models the exchange of messages between objects to achieve required functionality
- best suited for simple interactions among relatively small numbers of objects

Collaboration Diagrams

- objects work together to provide functionality
- realisation of use case
- message passing
- interactions - messages
- alternative scenarios
- specification level - classes
- instance level - objects
- links are labeled to show the order of events

Collaboration Diagrams



Sequence Diagrams

- Sequence diagrams shows the same information as the collaboration diagram but from a different viewpoint
- models interaction between objects, sequence of actions that occur in the system
- dynamic behaviour of system, emphasising order of events

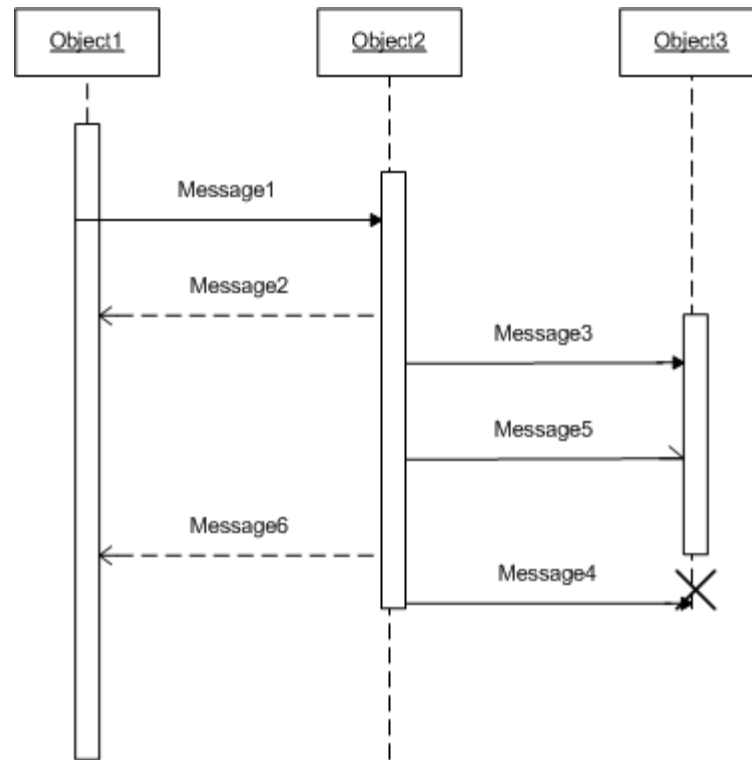
Sequence Diagrams

- timing of the messages, but not the relationships among the objects.
- describes how the system will behave in a particular scenario
- detailed flow for specific use case, describes a way the system may be used by its actors.
- order of interaction on vertical axis
- objects arranged horizontally

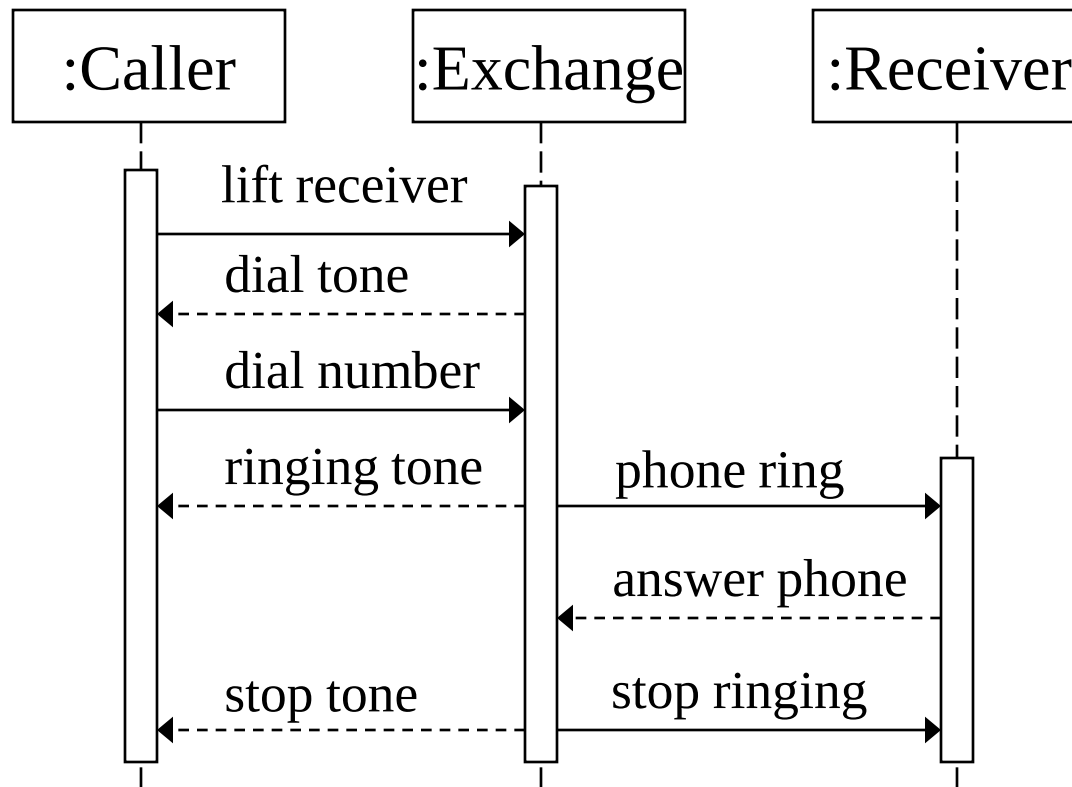
Sequence Diagrams

- lifeline (dashed vertical line)
- focus of control (narrow rectangle)
- messages
- object creation (by message)
- object destruction (cross)

Sequence Diagrams



Sequence Diagrams



Activity Diagrams

- activities - units of work
 - rectangle with rounded ends
 - made up of actions
 - on entry
 - do
 - on event
 - on exit
 - parallel operation (synchronisation bar)
 - fork, join

Activity Diagrams

- states - waiting for an event
 - rectangle with rounded corners
 - start (black dot)
 - one only
 - end (black dot with surrounding circle)
 - multiple
 - actions specified

Activity Diagrams

- transitions (arrowed lines)
 - from one activity to another
 - from one state to another
 - from state to activity
 - triggerless
 - triggered (exiting state)
 - multiple transitions allowed

Activity Diagrams

