School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

COMP2931 Software Engineering

Lecture 7: Use Cases & Behaviour Diagrams

Nick Efford

Twitter: @python33r
Email: N.D.Efford@leeds.ac.uk
Web: http://comp2931.info

Last Time...

UNIVERSITY OF LEEDS

- We discussed the origins of UML and the value of using diagrams to communicate
- We saw that UML provides seven different diagrams to capture structural aspects of a system, plus another seven to capture behavioural aspects
- We explored how classes are represented, as rectangles with optional compartments for attributes & operations, linked by lines to show how they are related
- We introduced package and component diagrams

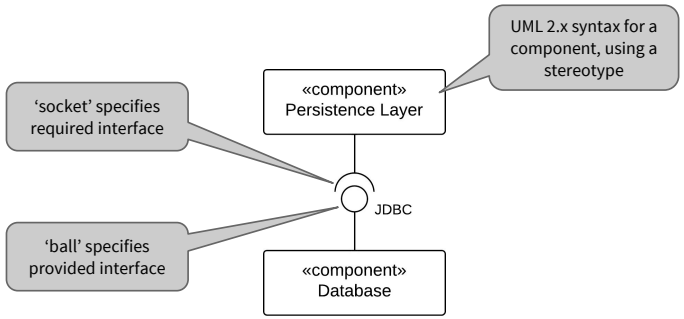
Today's Objectives

UNIVERSITY OF LEEDS

- To pick up where we left off, looking briefly at component diagrams and then at deployment diagrams
- To examine the role of **use cases** and use case diagrams
- To see how the flow of events in a use case can be represented using an **activity diagram**

Component Diagram Example

UNIVERSITY OF LEEDS



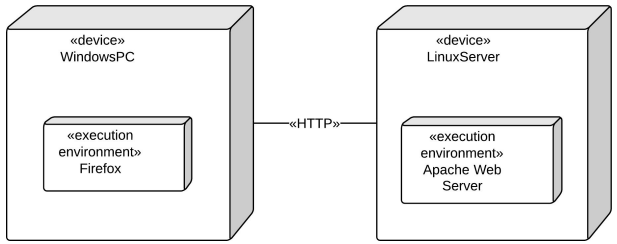
Deployment Diagrams

UNIVERSITY OF LEEDS

- Specify the physical environment in which a software system will execute, and how the software is deployed in that environment
- Environment consists of **nodes** - which can be physical hardware or an 'execution environment' of some kind (browser, VM, etc)
- Nodes are shown as '3D boxes' or as stereotyped icons that resemble the actual hardware
- Physical **artifacts** (`.exe` files, JAR files, documents, etc) are deployed on nodes

Deployment Example

UNIVERSITY OF LEEDS



What is a Use Case?

UNIVERSITY OF LEEDS

"A type of transaction between the system and one or more **actors**, which, when complete, yields a result of measurable value to the main actor."

Actors are **roles**, rather than specific human users of a system; an actor could even be another system!

Use cases specify functions required of the system

Use Cases...

UNIVERSITY OF LEEDS

- Describe what a system does or should do from the perspective of its users
- Show WHAT is required of a system, but not HOW that requirement is or will be realised
- Can be used to model what an existing system does or represent requirements of a new system
- Can be constructed for systems in their broadest sense - i.e., not just software but also businesses, etc

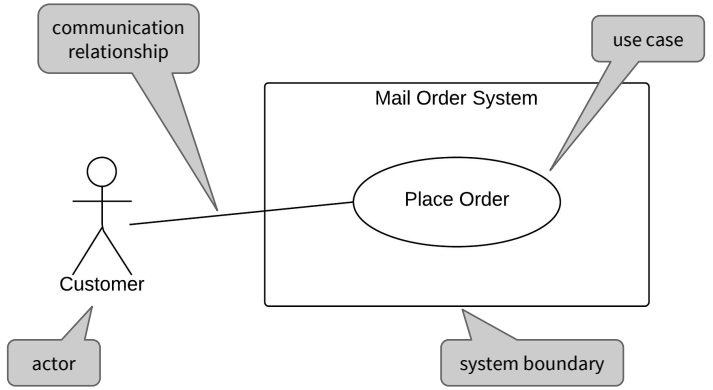
Use Case Model

UNIVERSITY OF LEEDS

- Use case diagram
  - Provides an overview
  - Identifies use cases and associated actors
- Use case descriptions
  - Provide detail of what happens in each use case
  - Format to be discussed later…

Basic Use Case Diagram Syntax

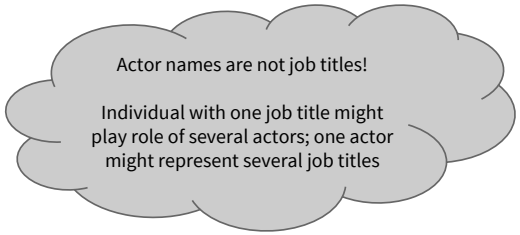
UNIVERSITY OF LEEDS



Actors

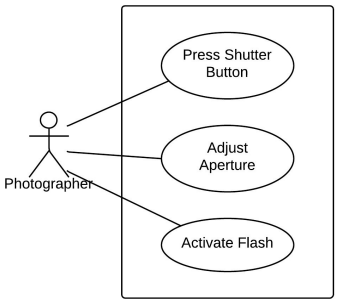
UNIVERSITY OF LEEDS

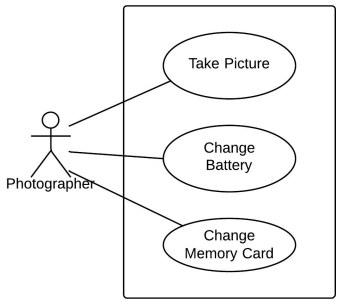
- Usually drawn as a ‘stick figure’ with a name underneath, more rarely as a class icon with «actor» stereotype
- Bad actor names: “Lecturers”, “Nick Efford”
- Good actor names: “Lecturer”, “Student”, “Customer”



Good & Bad Use Cases

UNIVERSITY OF LEEDS



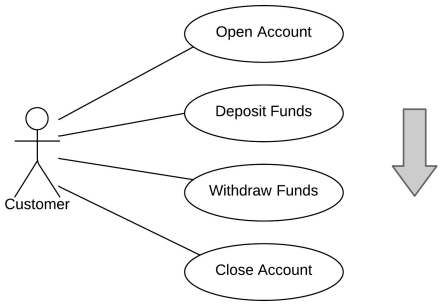


Which is the good diagram and which is the bad diagram? Why?

Good Use Case Style

UNIVERSITY OF LEEDS

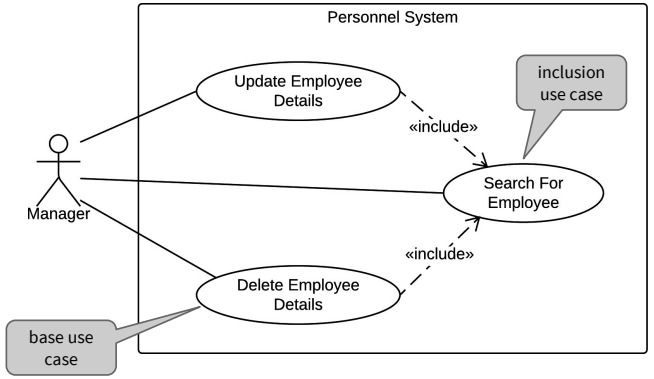
- Place **primary actors** (those initiating a use case) on left side, secondary actors on the right
- Where possible, arrange use cases in the order implied by timing considerations



«include»

UNIVERSITY OF LEEDS

Allows us to incorporate the behaviour of one use case into the flow of another



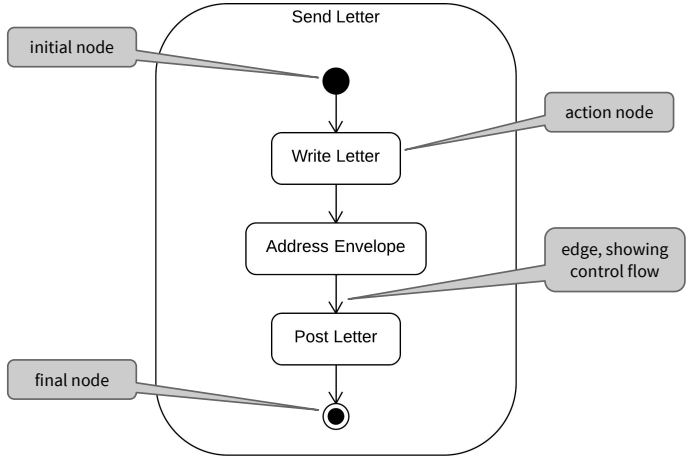
Activity Diagrams

UNIVERSITY OF LEEDS

- UML's version of the classic flowchart
- Focused on communication one specific aspect of a system's dynamic behaviour
- Typically used for modelling business processes and for modelling the logic of a single use case
- Consists mainly of **action nodes** and **control nodes**, linked by **edges** that represent flow of control

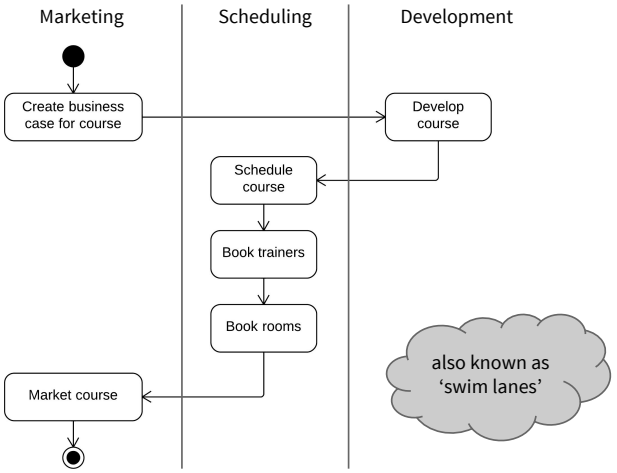
Simple Example

UNIVERSITY OF LEEDS



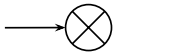
Activity Partitions

UNIVERSITY OF LEEDS

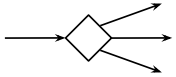


Other Control Nodes

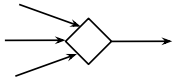
UNIVERSITY OF LEEDS



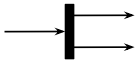
Flow final node: terminates a specific flow, leaving other flows in the activity unaffected



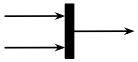
Decision node: output edge whose **guard condition** is true will be traversed



Merge node: combines flows into single output edge



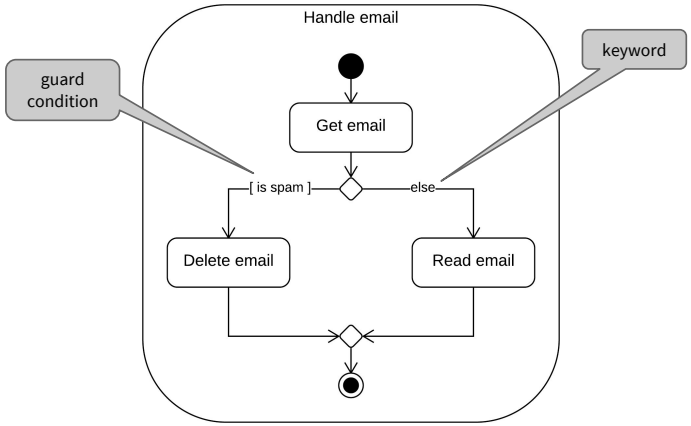
Fork node: splits flow into multiple concurrent flows



Join node: synchronises multiple concurrent flows

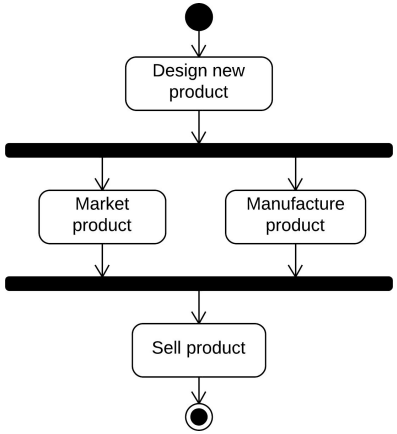
Decision & Merge Example

UNIVERSITY OF LEEDS



Fork & Join Example

UNIVERSITY OF LEEDS



Exercise 1

UNIVERSITY OF LEEDS

A user logs into a web application. If this is their hundredth login, it displays a customer survey, followed by the home screen; otherwise, it just displays the home screen.

Draw the activity diagram for this, omitting the initial & final nodes.

Exercise 2

UNIVERSITY OF LEEDS

A store receives an order from a customer. They fill the order, then deliver the order. While this is happening, an invoice is issued and payment is received. When the order has been delivered and payment has been received, the order is closed.

Draw the activity diagram for this, omitting the initial & final nodes.

Summary

UNIVERSITY OF LEEDS

We have

- Seen examples of component and deployment diagrams
- Explained what **use cases** and **actors** are
- Considered how use cases and actors are associated with each other in **use case diagrams**
- Explored how a flow of actions in a business process or IT system can be shown as an **activity diagram**

Follow-up / Reading

UNIVERSITY OF LEEDS

- Various UML books in the library - e.g., *Schaum's Outline of UML* (2nd ed.) by Bennett et al; see also ebook links on website
- http://www.uml-diagrams.org
- Sparx Systems UML 2 tutorial http://bit.ly/sparxuml2
- UML Quick Reference Guide http://bit.ly/umlquickref
- DZone Refcardz: Getting Started With UML http://bit.ly/umlrefcard
- Ambler's UML style guidelines http://bit.ly/umlstyle