



Use Cases

Stuart Anderson

Room 1610, JCMB, KB

0131 650 5191, soa@inf.ed.ac.uk

Reading/Activity

- Please read the Volere template that is linked off the notes page on the course web page.
- Please read Alistair Cockburn's paper *Structuring Use Cases with Goals* which is also available off the notes page.
- Read the outline of the practical activity in preparation for Wednesday's tutorials.

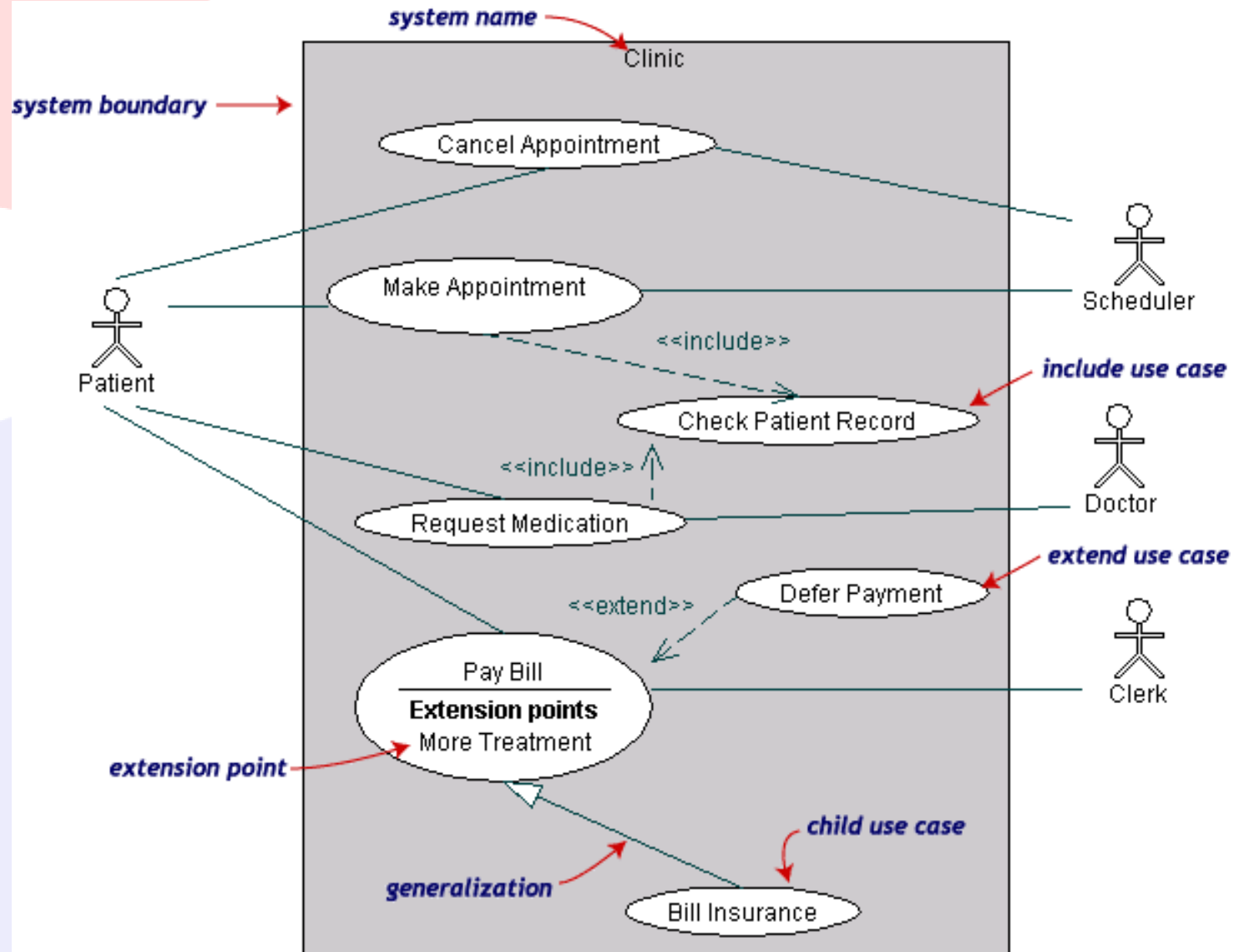
Use Case Diagrams

- Intended to support Requirements Engineering
- Strengths: capture different actors views of the system; comprehensible by naïve users; capture some elements of structure in requirements.
- Weaknesses: not particularly strong in capturing non-functional aspects; doesn't support analysis particularly well;

Why Use Case Diagrams?

- Model actions of the system at its external interface
- High level view of the system
- Capture how the system coordinates human action
- Rapid change allows exploratory approach
- Link to scenarios keeps the activity concrete
- Comprehensible by users.
- Capture some structure.

Specimen Use Case Diagram



Anatomy of a use Case Diagram

- Basic Diagrams: actors are represented as stick figures, use cases as ellipses, lines represent associations between these things. So basic use case diagrams show who is involved with what.
- Can be used to help in structuring systems: e.g. the scheduler and patient more or less form a sub-system – look at delegating appointment management to a single component or sub-system.
- Take care to identify *generic* actors who do a particular task – don't get confused with job titles etc.
- Aim for reasonably generic use cases – try not be too detailed at first.
- Use case diagrams should not be too complex.

Attaching Use Cases

- Use cases should be attached to each case in the diagram. Use case is a generic sequence of actions undertaken in using the system, e.g. :
 - Patient: request appointment to scheduler
 - Scheduler: queries System for available times
 - System: responds with times
 - Scheduler: negotiates with Patient on suitable time
 - Scheduler: confirms time with system
 - System: responds with confirmation of appointment (e.g. booking number).
 - Scheduler: communicates confirmation to Patient
- Provided generic test scenarios for the full system.

Structure in Use Cases

- Generalisation: between use cases; between actors – in use cases, see the example: **pay bill** is a generalisation of **bill insurance**. A **“health worker”** is a generalisation of **“nurse”, “doctor”** etc.
- *Include* relationships hold when one use case is included in others. For example, looking up medical records is included in many other use cases.
- One use case extends another when it has the same function but is more particular, e.g. deferring payment as a means of paying.

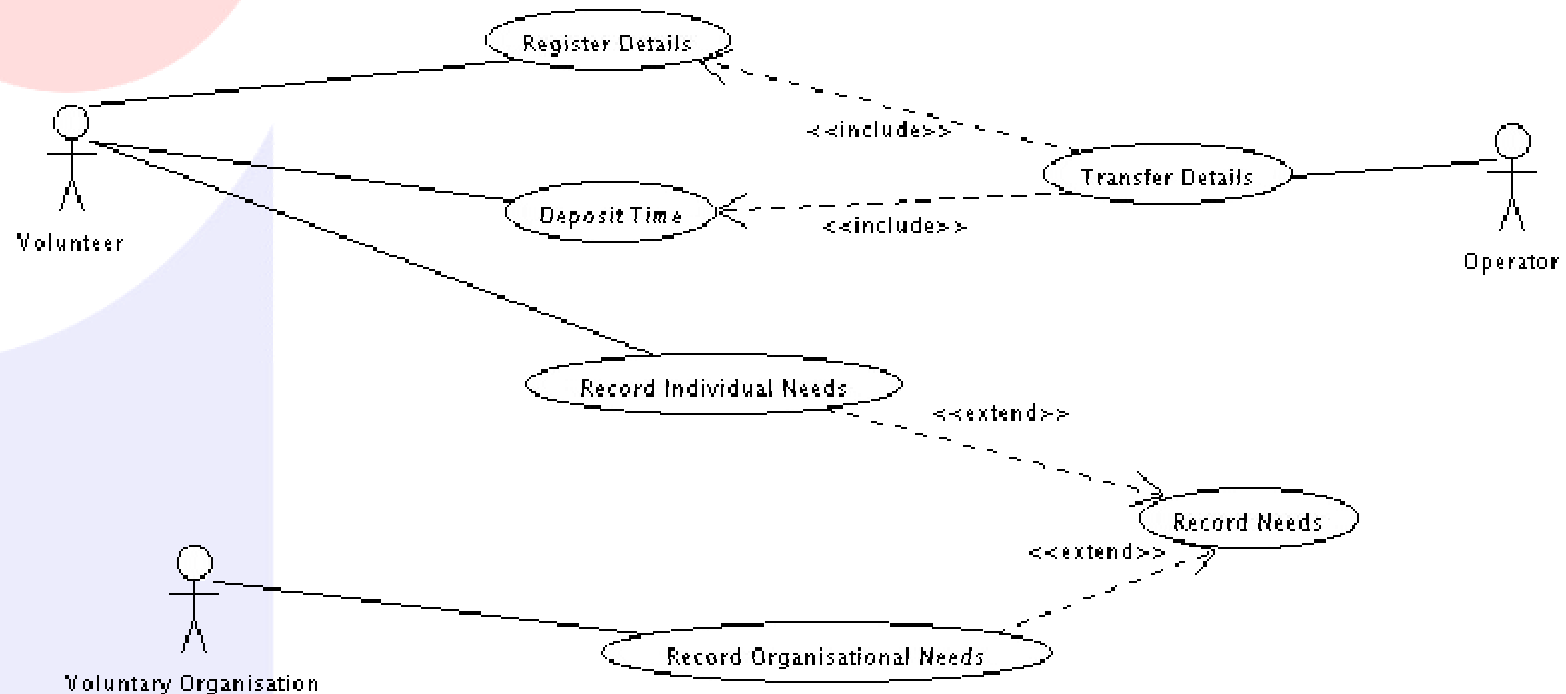
Software Requirements

- Main activities involved in Software Requirements engineering:
 - Elicitation: Identify sources; Elicit requirements
 - Analysis: Classify requirements; Model; Top-level architecture; Allocate requirements to components; Negotiate requirements
 - Specification: Requirements Definition Doc; Software Requirements Specification; Document Standards; Document Quality
 - Validation: Reviews; Prototypes; Modelling; Test definition
 - Management: Traceability; Attributes; Change/Evolution
- The pattern, sequence and interaction of these activities is orchestrated by a Requirements Engineering Process.

Creating Use Cases

- Find the actors and use case names: can use checklists – who uses the system, who manages the system, who maintains the system etc.
- Prioritize the use cases on the basis of utility or frequency of use – depending on the process this may be closely linked to what is needed in the process
- Develop each use case: develop the associated use case.
- Work to find structure in the use case: identify relationships introduce these to the model

VolBank: Incomplete Diagram



VolBank: Activity

- In class, or afterwards if it is not completed in class:
 - Who are the main actors in the VolBank example?
 - Can you identify all the main use case names in the system?
 - What opportunities are there to structure the use case diagram?
 - Can you see any non-functional requirements that are present in the specification?
 - How well are non-functional requirements represented in the use case diagram?