MANCHESTER 1824

The University of Manchester

COMP23420: Introduction to Software Engineering

The Staff

- Robert Stevens (sem 1)
- Caroline Jay (sem 1)
- Kung-Kiu Lau (sem 1)
- John Sargeant (sem 2 and course leader)
- Liping Zhao (sem 2)

Outline

- What is software engineering?
- How does the course work?
- What the course expects from you
- Functional Modelling: Activity Diagrams

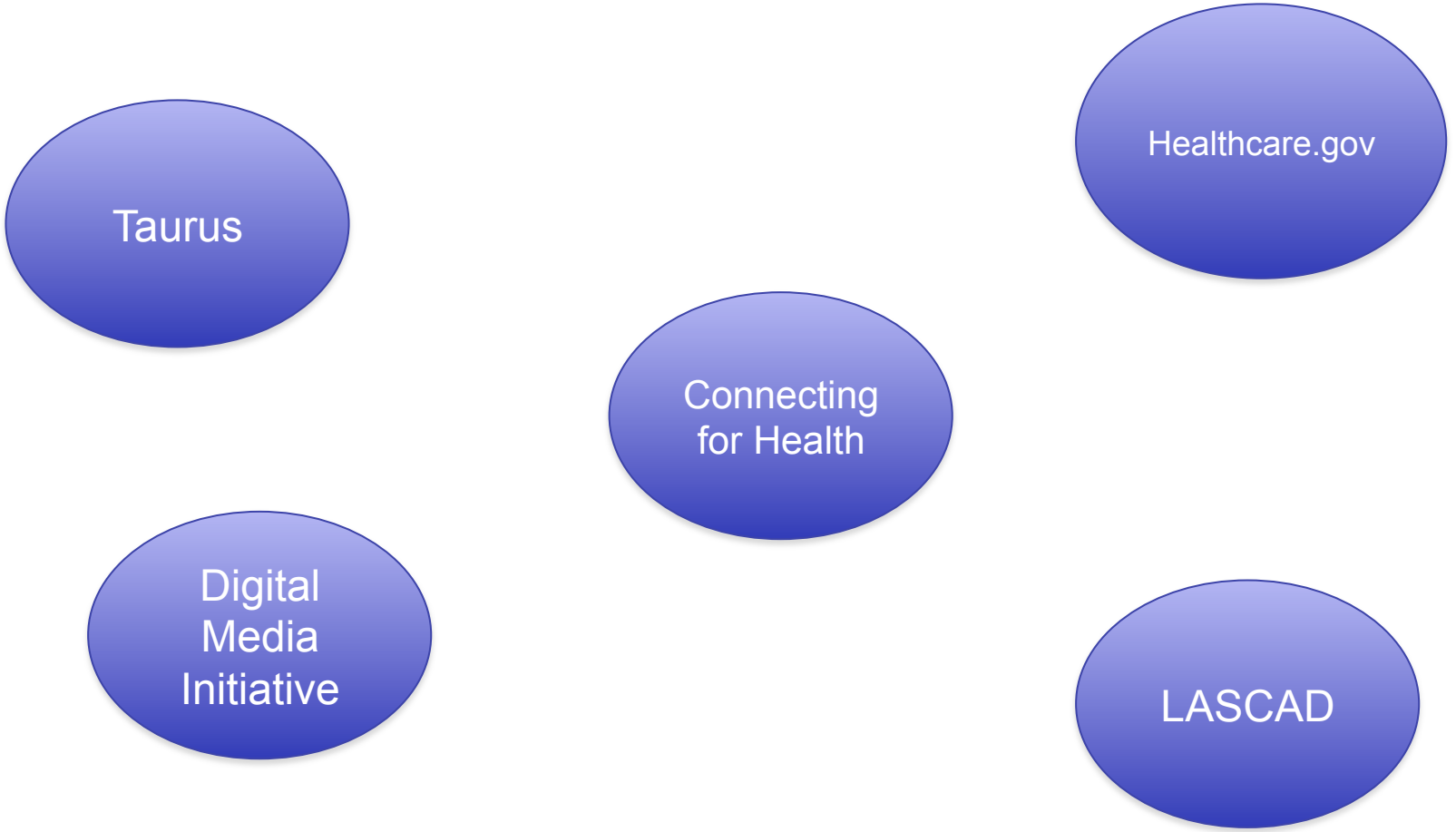
The software crisis

- In the late 1960s increasingly complex machines and software process led to an observed “software crisis” caused by:
  - Projects running over-budget.
  - Projects running over-time.
  - Software being very inefficient.
  - Software being of low quality.
  - Software often not meeting requirements.
  - Projects being unmanageable and code difficult to maintain.
  - Software never being delivered.

  It is now the 2010s…

Source: http://en.wikipedia.org/wiki/Software_crisis

“Failed” software projects



Source: http://en.wikipedia.org/wiki/List_of_failed_and_overbudget_custom_software_projects

Why do software projects fail?

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Unmanaged risks
- Poor communication among customers, developers, and users
- Use of immature technology
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management

Source: http://spectrum.ieee.org/computing/software/why-software-fails

Is software engineering engineering?

- Is software engineering equivalent to programming?
- Is HCI part of software engineering?
- Is budgeting time and money part of software engineering?
- Are ethical considerations part of software engineering?
- Are people part of software engineering?

Engineering: Building bridges…

Scientific Principle

Economics

Social Organization

Robust solution to crossing a gap

Newton's third law of motion: Every action has an equal and opposite reaction.

http://stock-free-images.com

Engineering: Building software

- The economics and social organisation aspects of trad engineering have obvious counterparts.

Economics	
Social Organization	
Robust solution to route-finding	

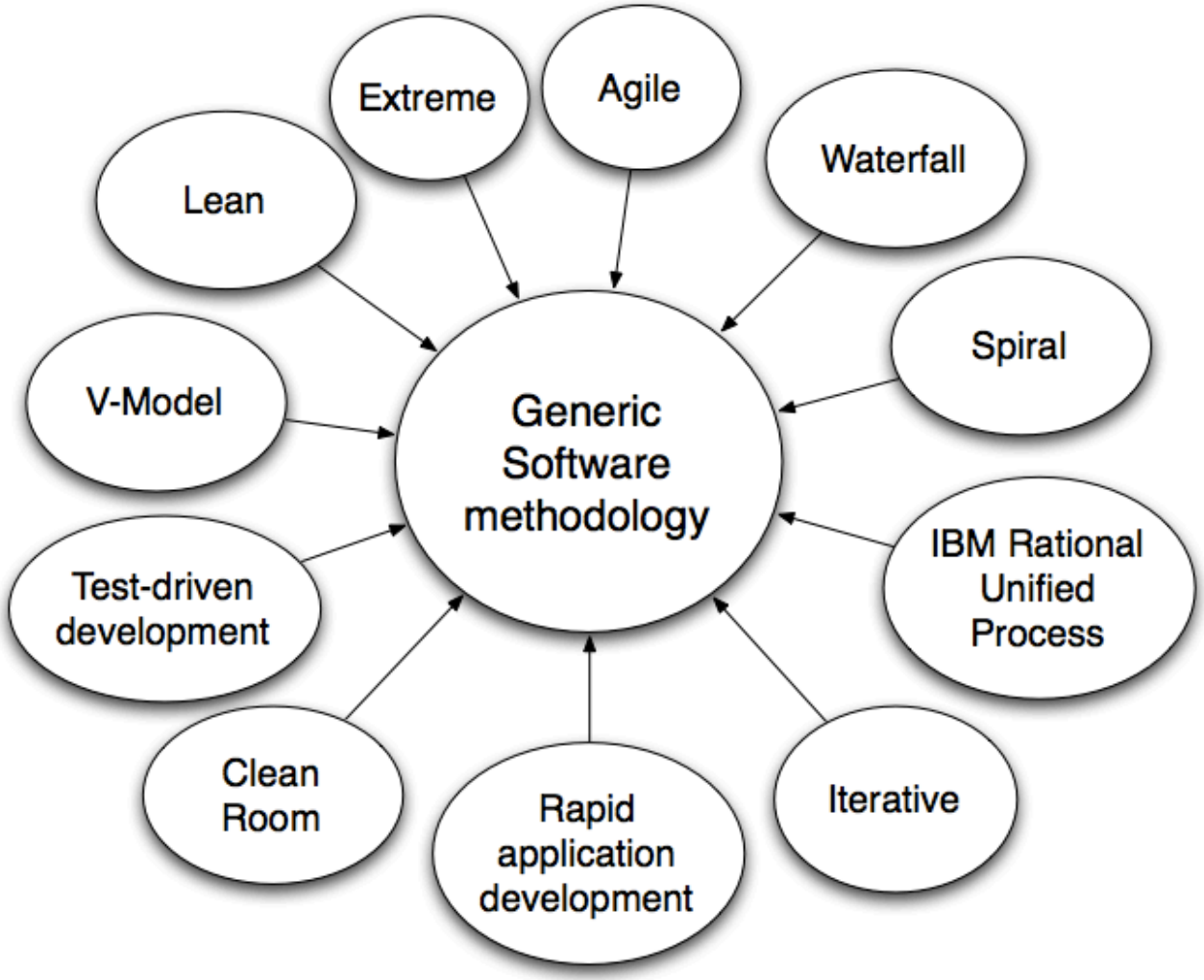
http://www.clubic.com/

- What about the scientific principles bit?

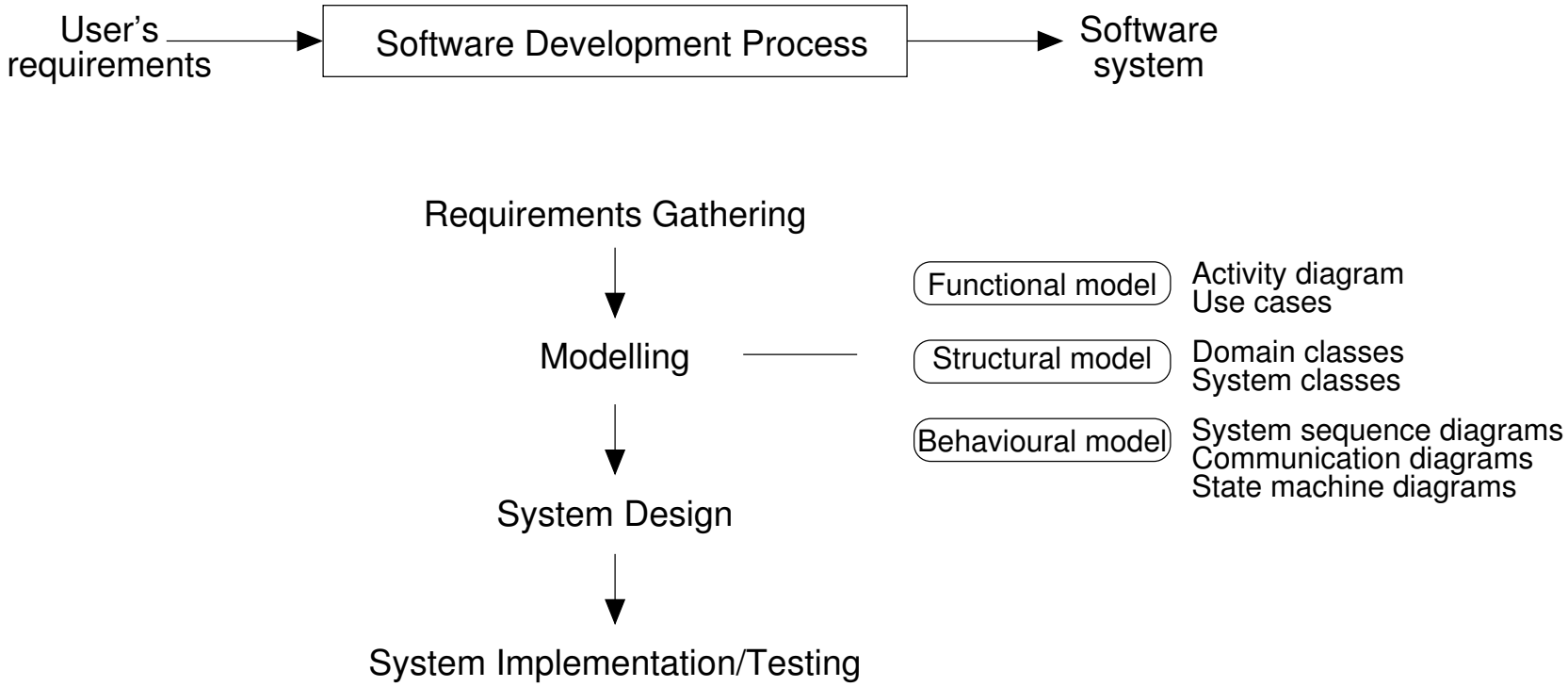
How is software engineering different?

- Scientific principles can be embedded in software – models of Newton's laws of motion etc
- We use principles from computer science to deliver robust, maintainable solutions to problems
- For example, separation of concerns in software, encapsulation,…
- Engineering can have analogues of increments and iterations
- We'd like to think the time-scales are different in software engineering
- Software is short-lived compared to a lot of engineered things

Lots of Software Development Methods



A Generic Process



A Generic Software Development Process.

Semester one and Semester two

- Semester one: requirements gathering and modelling (functional, structural and behavioural)
- Semester two: System design and system implementation.
- Semester one: Individual work
- Semester two: Team work, building a significant software system.

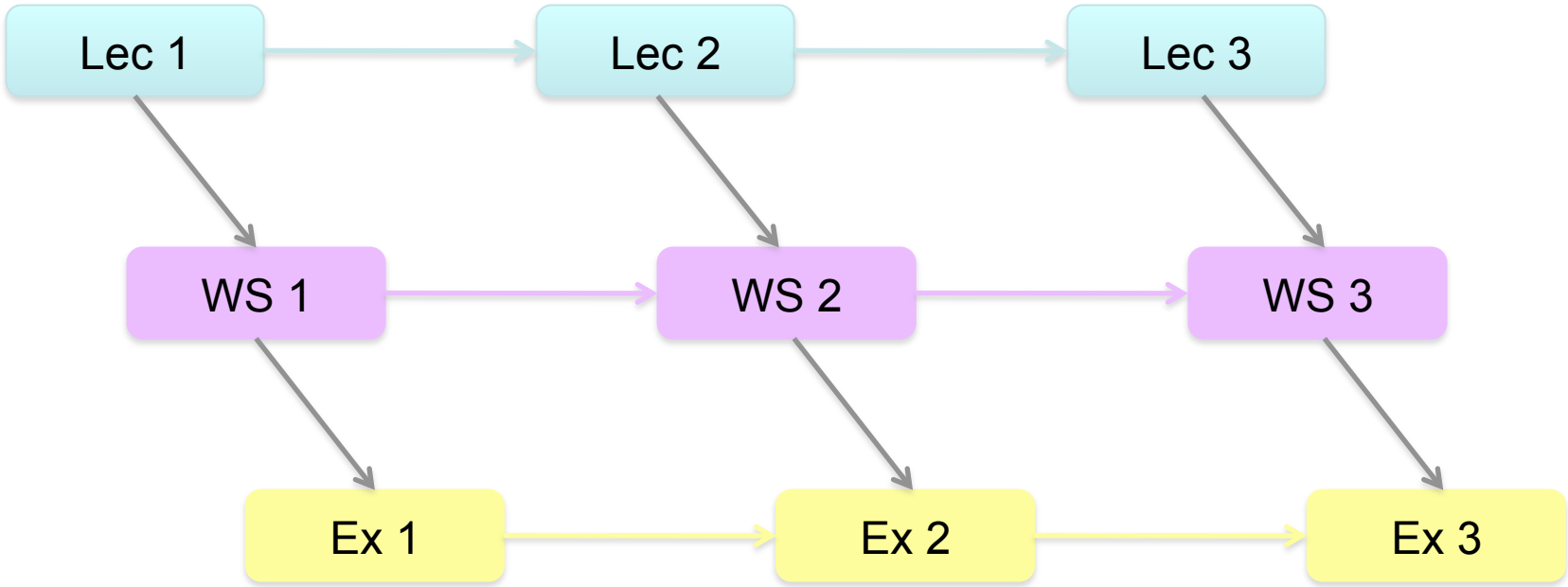
Team Working

- In semester two there are teams.
- Performance in teams will be marked – probably by yourselves.
- Performance and attendance in the first semester may be used to help us form the teams in Semester Two.

Why Workshops

- Your first year project should have raised issues
- Some learning is best done via lectures
- Some learning is best done by you finding out
- Much software engineering falls in to this category
- COMP23420 workshops are set up to enable you to learn how to do some of the aspects of SE difficult to learn via lectures

Lectures, Workshops and Labs



Moodle

- Everything is on COMP23420's Moodle
- There is no printed lab book (though there is a printable file on Moodle)
- Important documents (scenarios; templates; UML are in the "Supporting documents" section).
- Before each workshop, print off workshop material or use a Colab computer or a mobile device

Assessment

- One exam at the end of each semester
- Exams will be on paper
- Semester one has a mixture of MCQ and long answer
- Both semesters labs contribute marks, but semester two makes more contribution in marks…
- Details are on Moodle and the course unit syllabus page.

Marking in the Sem 1 Coursework

- Submission via the `submit` and `labprint` mechanism.
- Exercises are due at 11:59 PM the day before the marking lab session.
- Marking will be by presentation to a demonstrator.
- You will show him/her your deliverables.
- You will narrate and illustrate your understanding and be marked face-to-face.
- Feedback is given at marking time, but do take your own notes too.

The Dog Ate My Homework

- Deadlines are hard; in sem1 11:59 PM the day before the “marking” one second late is late.
- Extensions on request (but please don’t)
- There are consequences for being late
- There are no (few) excuses

Starting Functional Modelling

- In workshop one you'll learn the most basic interviewing techniques.
- One of the main means for *requirements gathering*
- More on requirements gathering in lecture 2.
- You will gather both *functional* and *non-functional* requirements.

Functional and Non-Functional requirements

- Functional modelling relates directly to a specific process the system should perform or information it should contain “search for publications by year”
- Non-functional requirements relate to how the system should operate as a whole, e.g., performance, ethical standards
- Functional: “The system should ***do*** <requirement>”
- Non-functional “the system should ***be*** <requirement”;

Functional and Non-Functional requirements

- Customer chooses DVD
- Customer manages DVD list
- System should be able to work on any Web browser
- System should be compliant with standard Web accessibility guidelines
- Manager creates sales figures
- FlicksULike needs to present film ratings
- FlicksULike operator updates DVD database
- System should be able to respond in under one second
- System should be compliant with data protection act

Non-functional requirements and software qualities

- Many sorts of non-functional requirements are "*software qualities*"
- These are the attributes that help one judge the *quality* of a software artefact
- There's a whole ISO standard on software qualities
- https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

- It's very long, but are a lot of "-abilities" or "-alities"
- Learnability, reliability, usability, inter-operability, ….

Software quality checklist

- Conceptual integrity
- Maintainability (Supportability)
- Reusability (Availability)
- Interoperability
- Manageability
- Reliability
- Scalability
- Security
- Testability
- Usability

Subsets of the ISO standard: see e.g. http://msdn.microsoft.com/en-us/library/ee658094.aspx

Business Process

- A business process is a set of structured tasks that gives rise to a service or product
- Your system captures some business process
- Requirements gathering uncovers the business process
- Without an understanding of the BP you're sunk
- Functional modelling describes the business process

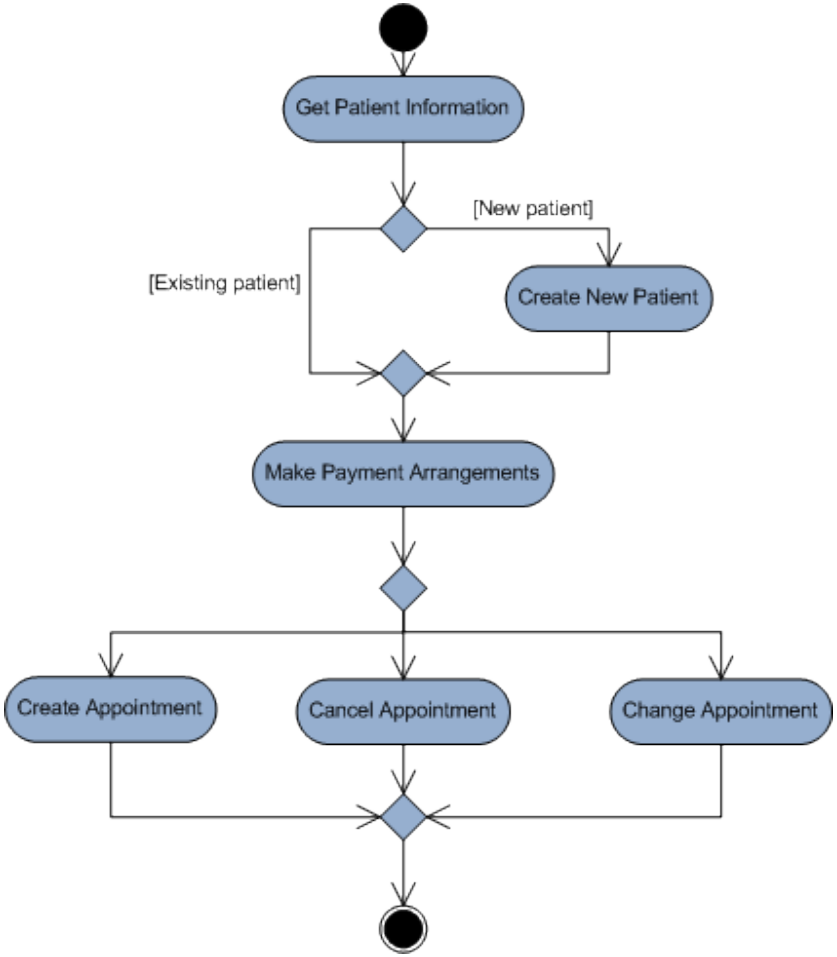
Capturing the Business Process

- Your interviews help you understand the “is-as” and the “to-be” systems.
- Need to capture the *business process.*
- *No point in any design without having the basics right*
- *…, though you will have to return to the basics again and again.*
- The major activities/actions and their relationships to form a “flow”.
- Captured in UML activity diagrams.
- You will learn activity diagrams in workshop one.

Activity Diagrams

- Can model both the “is-as” and the “to-be”
- Capture the activities and how objects flow between them
- An activity diagram is a *logical model*; it describes what it does, not how it does it
- A *physical model* (see structural and behavioural modelling)
- An *action* is a non-decomposable piece of bvehaviour
- An *activity* is a composed set of actions
- Typically we only use activities in activity diagrams….
- An *object node* represents objects in an object flow
- We will not use object flows at this stage; that is more of a system view for later on
- A *control flow* shows the sequence of execution

Sample Activity Diagram



An ATM Case Study

- Activity diagrams part of UML, the Unified Modelling Language
- UML is the main *concrete* thing you'll learn this semester
- The real thing to learn is how UML helps you capture various aspects of software
- We're using UML as a means to get you to think about software design
- There is a fully worked case study based on an Automatic Teller Machine available on Moodle
- It is a work in progress; do give us feedback

What do you Now Know

- What the course is about
- A very high-level view of the nature of software engineering
- You must go and read about it….
- The basic stages of a software project
- Modelling business processes with activity diagrams