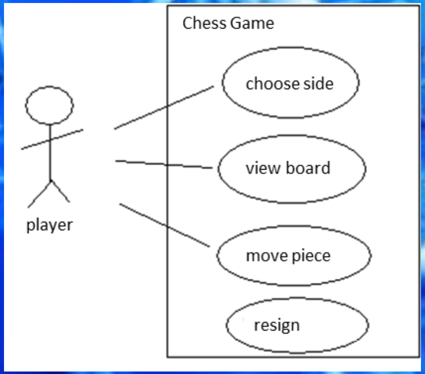
EEE3096S



Design Cycle & UML (1)

Lecture

Embedded Systems II

Dr Simon Winberg

Electrical Engineering
University of Cape Town

Outline

- Berger’s 7-phase ES lifecycle model
- Brief introduction to UML
- Use case modelling
- Intro to BOUML
- What is SysML (brief aside)
- Short UML relations exercise

Phases of an ES Design Lifecycle?

See the first part of the handout, write down what you would think are the main phases of development?

This is the main ES specific design cycle to remember

7 Phases of the Design Cycle

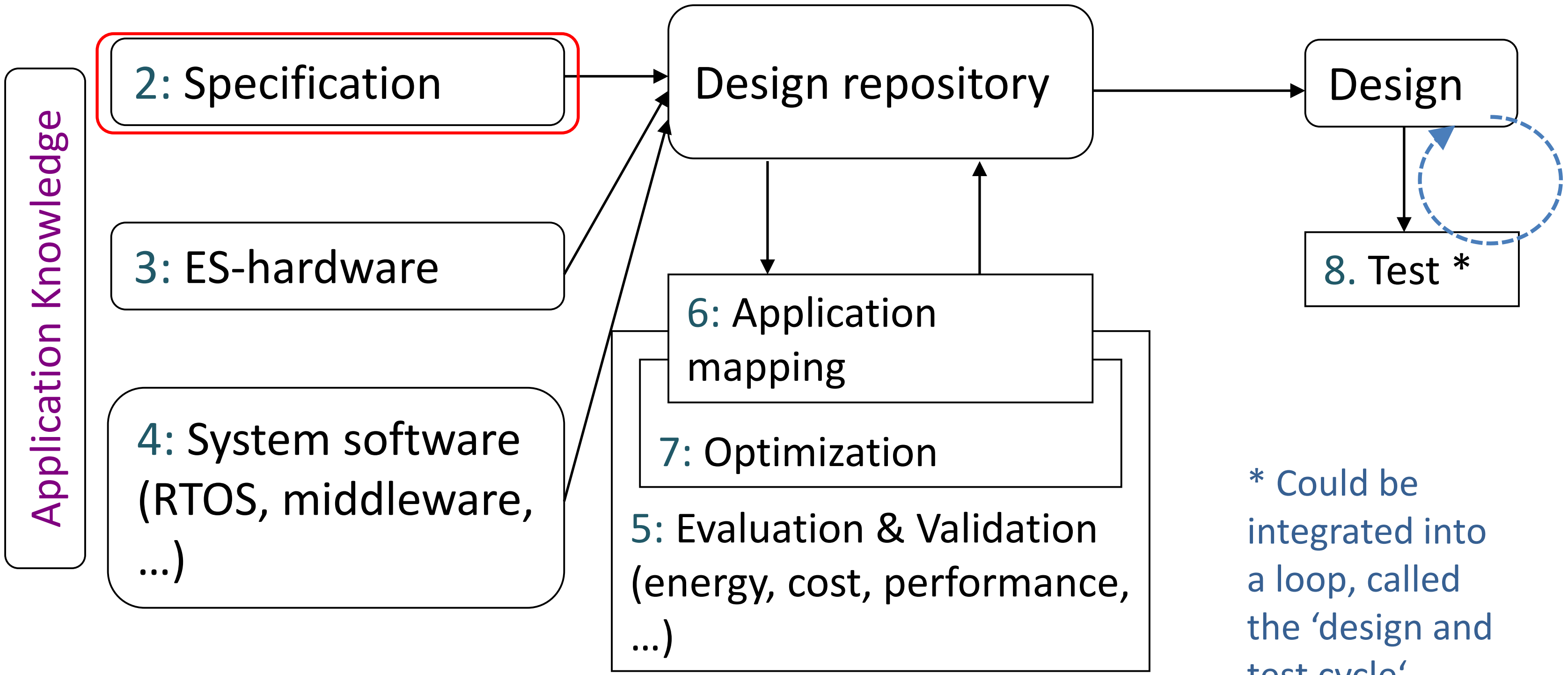
1. Specification
2. Partitioning into HW and SW components
3. Iteration and Implementation
4. Design of SW and HW done independently
5. Integration of SW and HW components
6. Acceptance Testing and Release
7. Maintenance and Upgrade

*Remember: **SPIDIAM***

This is the model I as guidance. But usually combined with the spiral model in practice.

Prof. Arnold Berger's (University of Washington) Model

Hypothetical design flow



* Could be integrated into a loop, called the 'design and test cycle' (D&T cycle)

Generic ES design cycle
The number in the diagram denote chapters of the textbook

NB: For this course consider this model (i.e. Marwedel's hypothetical model) a reference structure that considers the range of issues relevant to ES and some association between the actions. It considers some more specific issues e.g. system software and optimization. Berger's model (previous slide) is an easier model that is very useful in planning ES projects more broadly.

A Look at Berger’s Phases

- Requirements and Specifications
  - Learning about the client
  - Learning about the client needs
  - The ‘customer tour’
  - Developing the requirements
  - Progressively developing specifications

A Look at Berger's Phases

- Partitioning
  - Dividing the specifications into:
    - What will be done in hardware and
    - What will be done in software

A Look at Berger’s Phases

- Iteration and Implementation (I&I)
  - Designers work on aspects of design, refining the system towards acceptance
  - (*when you actually get to this stage* the spiral model or Marwedel's model essentially starts to take over in practice)
- Design of SW and HW done independently
  - The system at some point generally splits into I&I of design and I&I of hardware (if there is any hardware development)

A Look at Berger’s Phases

- Iteration and Implementation (I&I)
- Design of SW and HW
- Integration of SW and HW components
  - The SW and HW are brought together and I&I of the system continues

A Look at Berger's Phases

- Iteration and Implementation (I&I)
- Design of SW and HW
- Integration
- Acceptance Testing and Release
  - System is taken through its paces to demonstrate to the client it is complete
  - (jump back 1 or more steps if not)

A Look at Berger's Phases

- Iteration and Implementation (I&I)
- Design of SW and HW
- Integration
- Acceptance Testing and Release
- Maintenance and Upgrade
  - Changes are made to the system after the client has accepted it

Later on we go into details of validating the system, which generally culminates in a big acceptance test when the developers think the system is ready for the client.

Brief Introduction to UML & Why UML? & RT-UML?

What is UML?

- UML stands for:
  Unified Modeling Language
- Language for modeling and specifying systems
- Mostly used in Software Engineering, but applicable elsewhere (esp. Embedded Systems)
- Useful in both analysis (or requirements) and design phases the design cycle

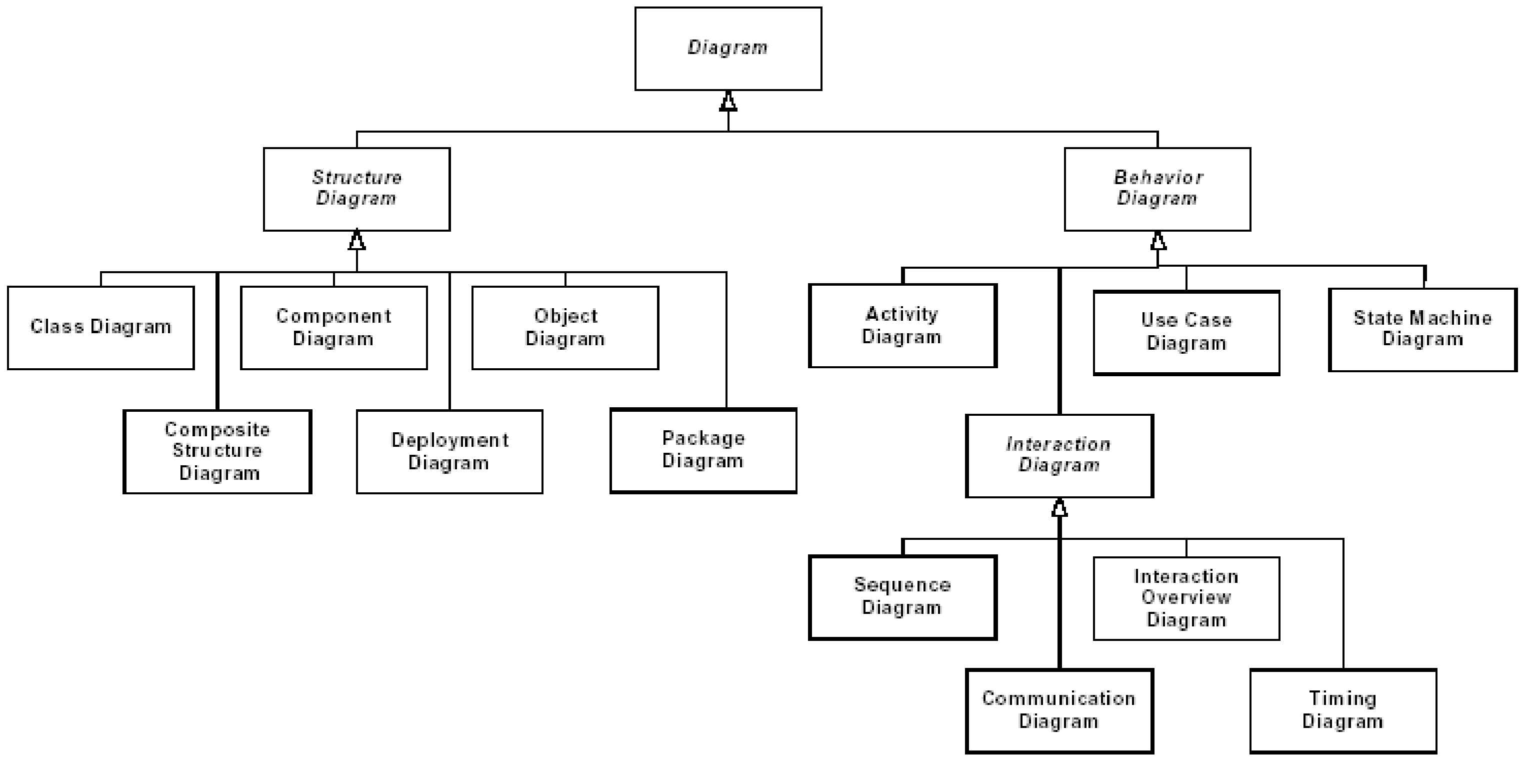
The History of UML

- Called "Unified" as it is based on a combination of three different precursors, sometimes called the 'Three Amigos' (of UML). They are:
  - OOD by Grady Booch's from Rational
  - OMT by James Rumbaugh from Rational
  - OOSE and Use-Cases by Ivar Jacobson from Objectory
- Development of UML 1.1 was completed in 1997
- UML 2.5 is current standard, but 2.0 still popular

I.e. clearly Jacobson was invited to join Rational later once the other two realized how useful his models were.

But don't confuse this with the 'Three Amigos' of Agile, which refers to agreement between the Product Owner (business/who will own the product), Developer and Tester. I refer to these as the Three Agile Amigos but that is the topic of a different course.

Structure of UML 2.0



from Wikipedia: UML

Diagram Types

- Structural diagrams
  - Describe the structure of design artefacts
- Behavior diagrams
  - Describe behavior of design artefacts
- Organization diagrams
  - Organize the other diagrams

Requirements:
Use Case Modeling

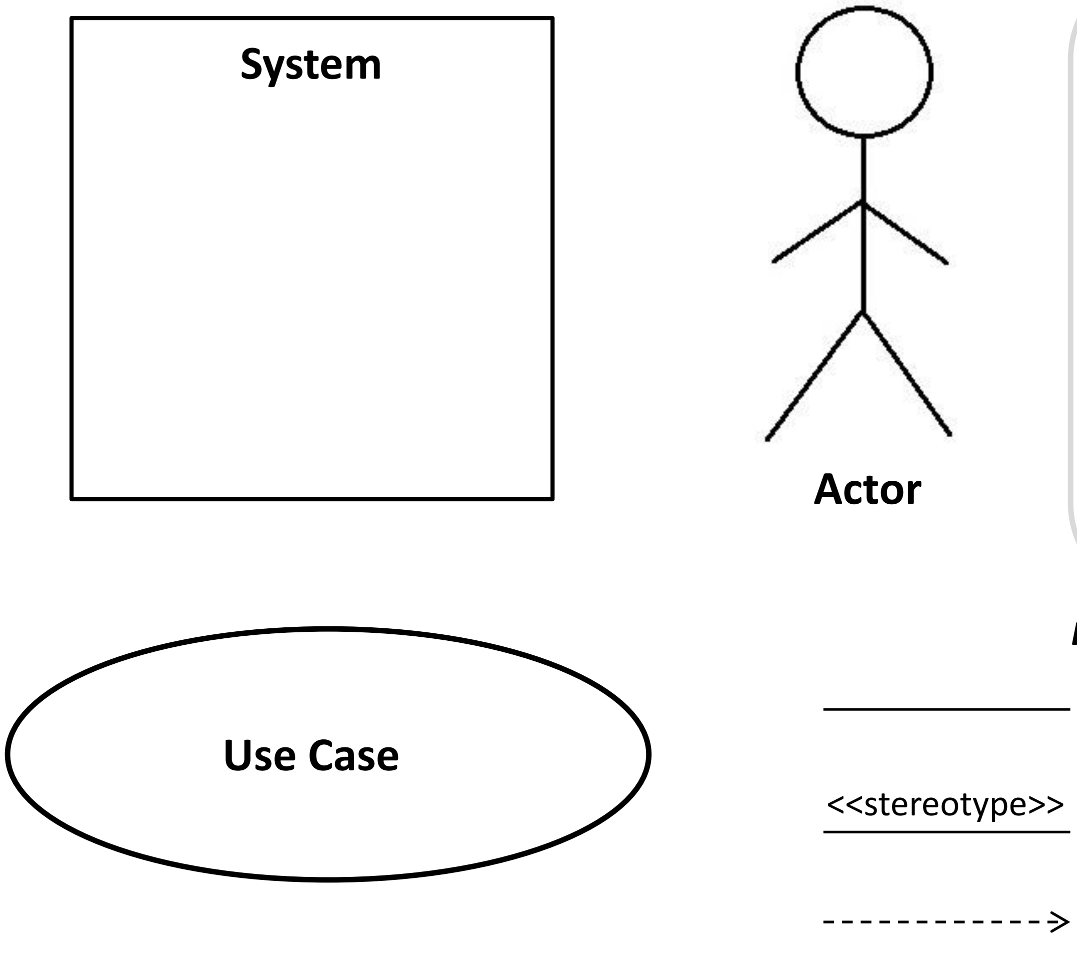
(A type of behavior diagram)

Use Case Models

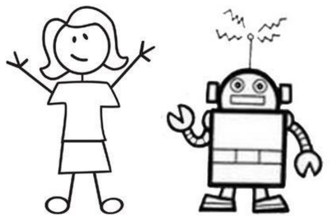
- Use case modelling is generally best used for requirements modelling
- These are diagrams simple enough for client or end-users to stand

Use Case Models

- A use case has four modelling primitives



Can you use different shapes for the actors, such as:



It's not technically recommended because it should be kept clear and consistent. But if it improves understanding I don't see why not!

How the diagram is structure

I’m going to take you through doing a very simple use case for a product.

Product to develop requirements for:

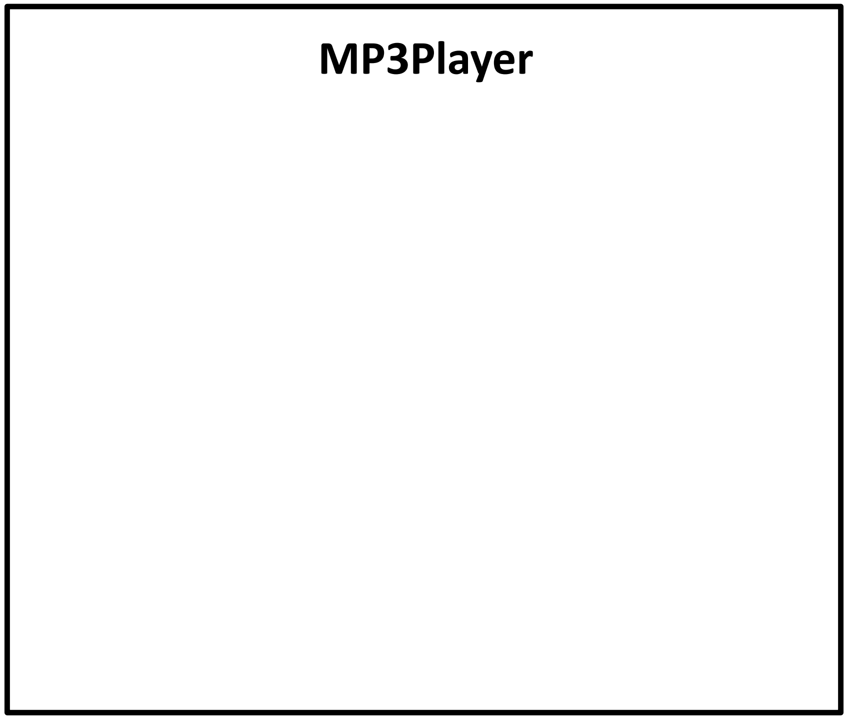
Imagine that we are developing a simple MP3 Player. What are the requirements? We can base it on the example given in an earlier lecture:
- List songs
- Select a song
- Play a song

And we might add (for later)
- Preview a song

Great, now we know what the basic requirements are, let’s model it…

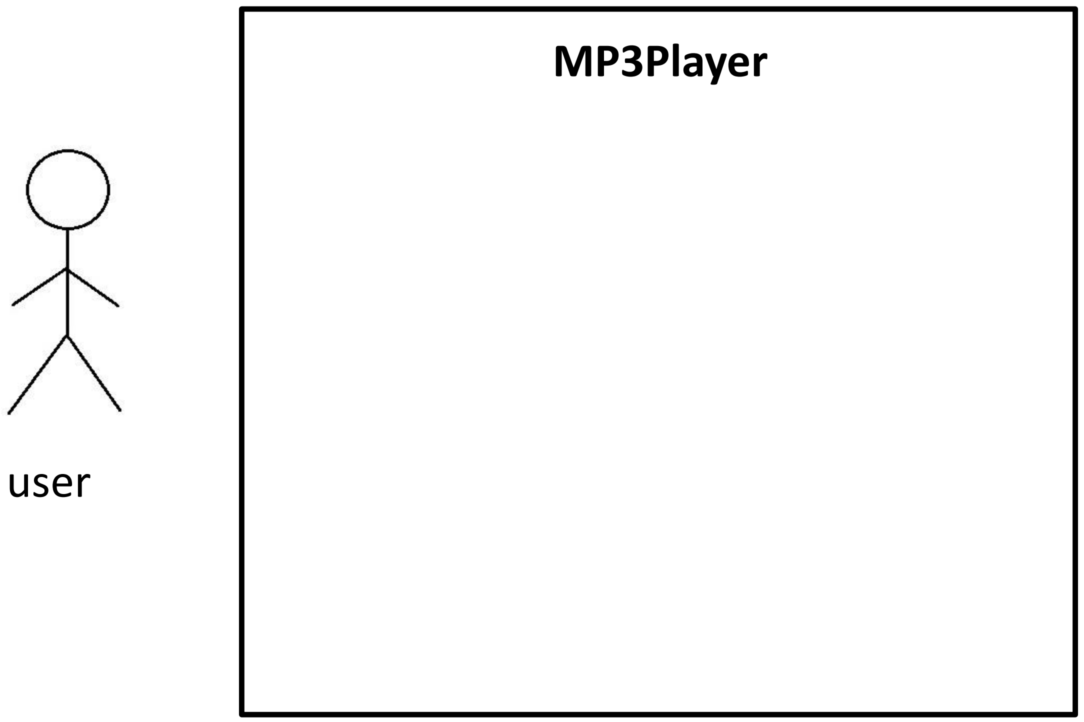
How the diagram is structure

Start with adding the system…



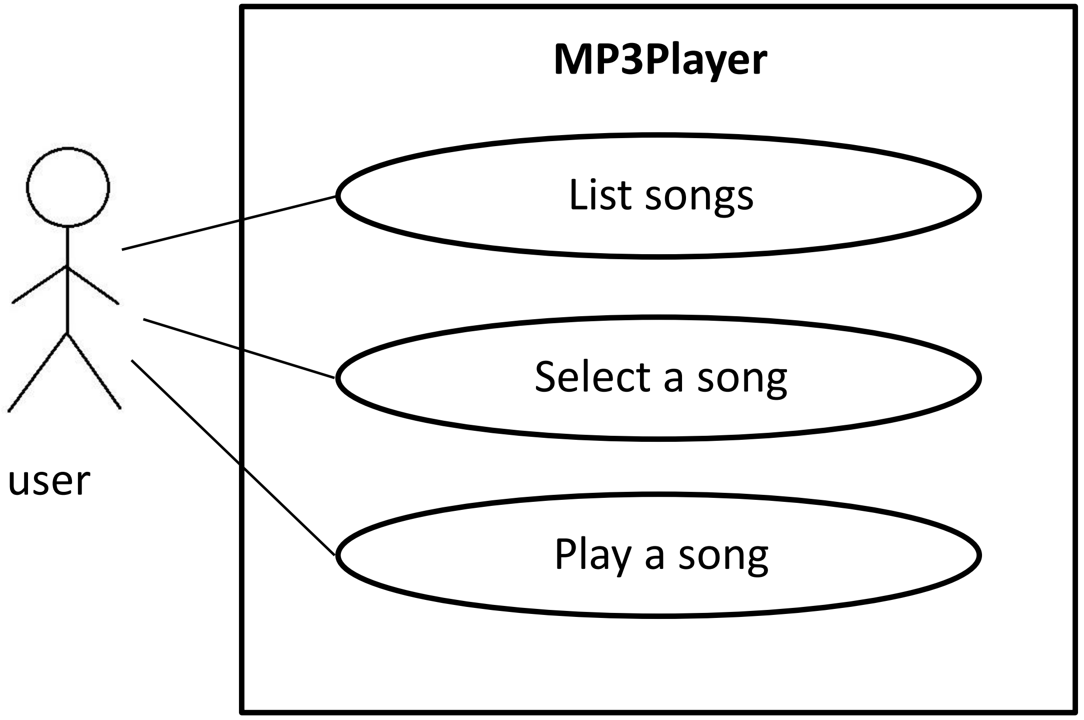
How the diagram is structure

Now add the actors



How the diagram is structure

Now add the use cases…



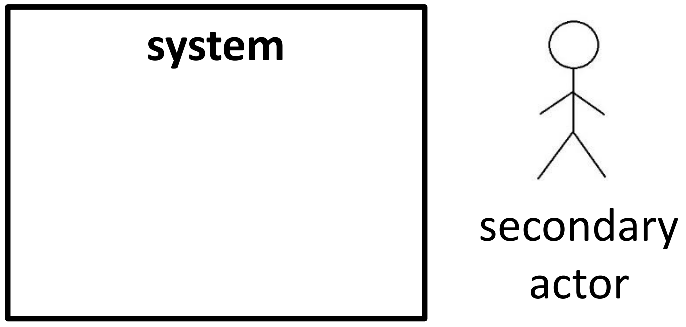
That is the main thing done!

This is a association relation (in this case it is undirected)

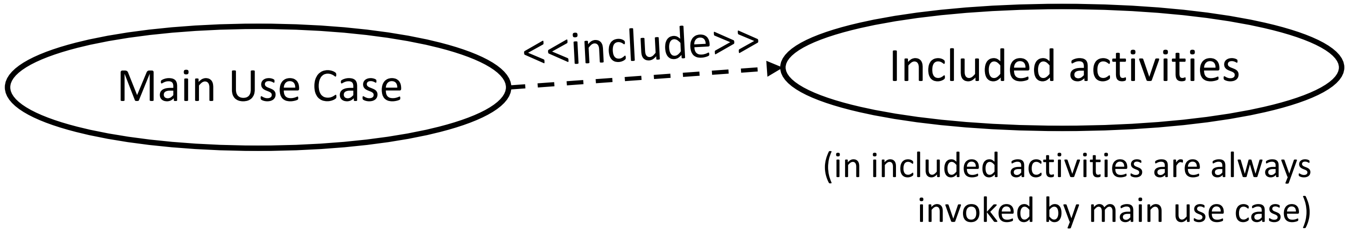
Refining the Use Case further

We have seen the basic use case but there's more…

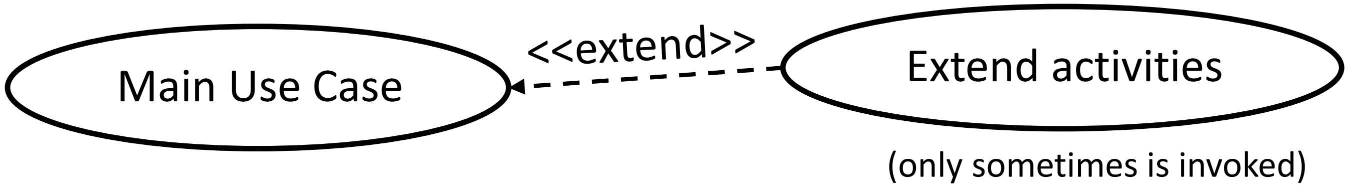
Adding a *secondary actor* (one that is involved when a certain use case activates)



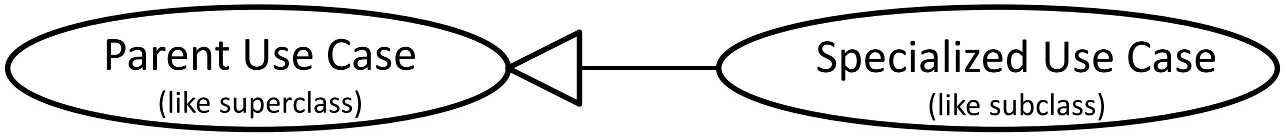
Include relation


Main Use Case
<<include>>
Included activities
(in included activities are always invoked by main use case)

Extend relation



Generalization relation



BOUML

BOUML is a neat and efficient, also fairly powerful UML modelling language that supports UML 2.
It can generate: C++, Java, Idl, Php, Python and MySQL code

It was developed by Bruno Pagès (in France) and detailed can be found on the website: https://www.bouml.fr/index.html

Since the release 7.0 BOUML is available as free software.

It is available for Raspbian(!) maybe not latest ver but you can compile from sources.

There appears to be no limitations to the size or other aspects of the diagrams you create.

Let’s have a brief preview… in a few moments (!)

(see Vula site for link to BOUML software)

Preview of BOUML

- Let’s make a more complicated product to show more Use Case relations…
- Product description:

CloudSongs is an embedded product used to access MP3 media files on the cloud, which are stored on SongSrv. To use CouldSongs the user must log in, which in turn performs an authorization request on the SondSrv. If the login fails a Invalid User message is displayed. Once logged in, the user can List Songs or Select a Song; both these operations access a Song Directory. Once the user has selected a song, the user uses Play Selected Song, to start playing the song. This operation uses the Access Song operation to get the audio file data from the server. If no song was selected then an Not Selected Error occurs.

A Preview mode can be enabled in which the Play Song operation instead previews songs, plays them for 15 seconds each time a song is selected to play.

I’ll run through how you make a start in BOUML and then if you want to experiment further with the tool you can finish off the design as homework (the completed design is given at the end of the lecture)

Worksheet Activity

You have noticed in your worksheet there is a blank space for drawing models

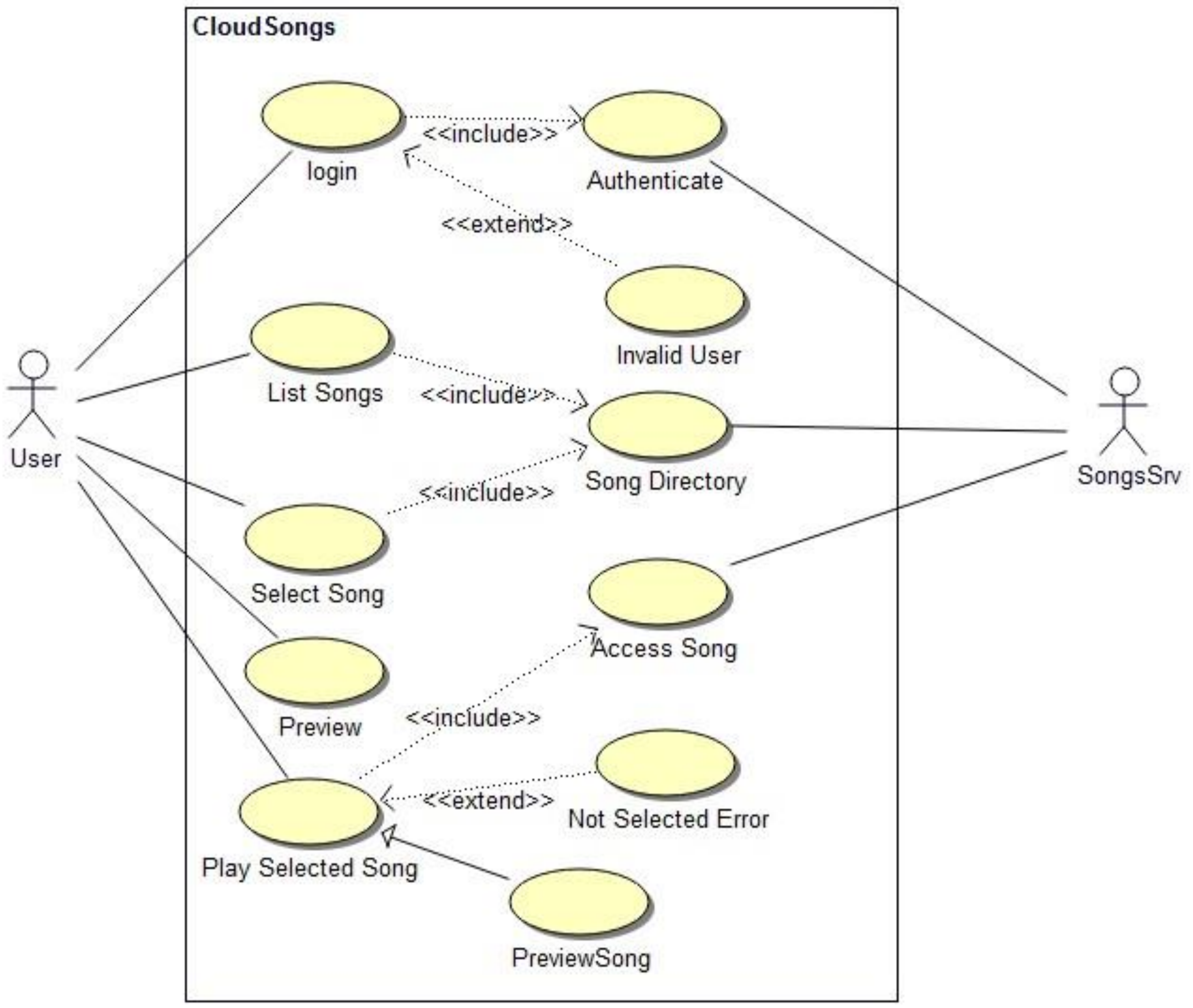
Please add a drawing this functionality…

… To use CouldSongs the user must log in, which in turn performs an authorization request on the SondSrv. If the login fails a Invalid User message is displayed.

What modelling elements will you use??

Take 5 minutes to give it a try … white I get BOUML set up

CloudSongs Completed Example



cloudsongs.zip

SysML

- Systems Modeling Language (SysML) is a general-purpose modelling language for systems engineering applications.
- It supports: specification, analysis, design, verification and validation of a broad range of systems and systems-of-systems, and is quite well suited for (large) embedded systems!
- SysML is an ‘extended subset’ of UML (using UML's profile mechanism). The language’s extensions support systems engineering.

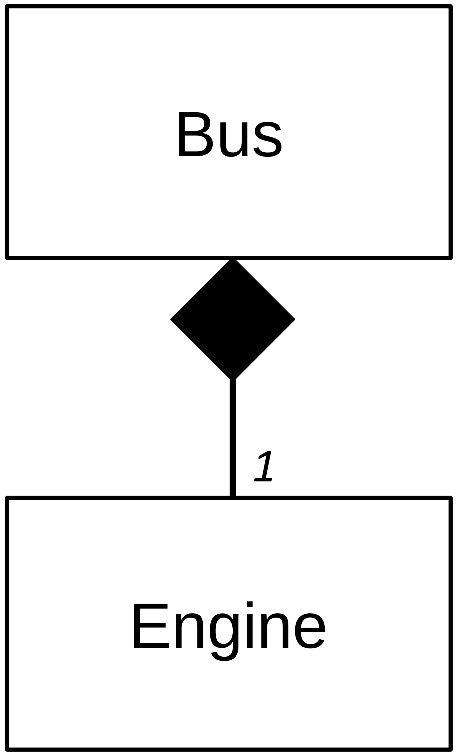
Types of Relations

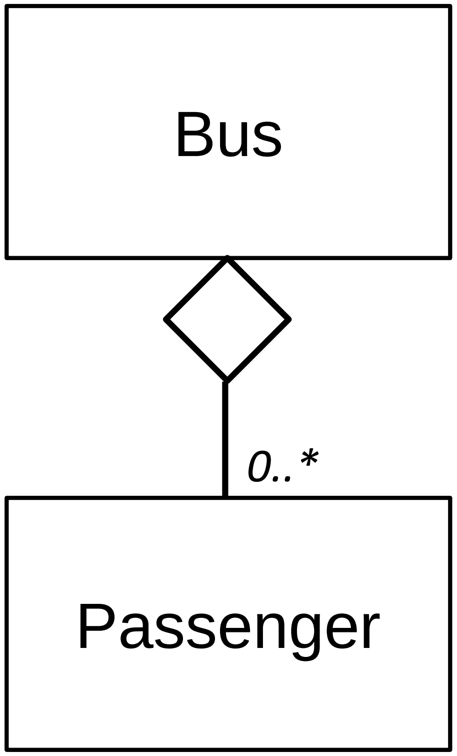
- Similar types of relations crop up frequently in UML diagrams. Before going on to more diagrams, let's see if you know these…

Look at the bottom of the worksheet… and draw for me

- A containment relation where a bus contains one engine.
- An aggregation relation showing that a bus can group zero or more passengers.

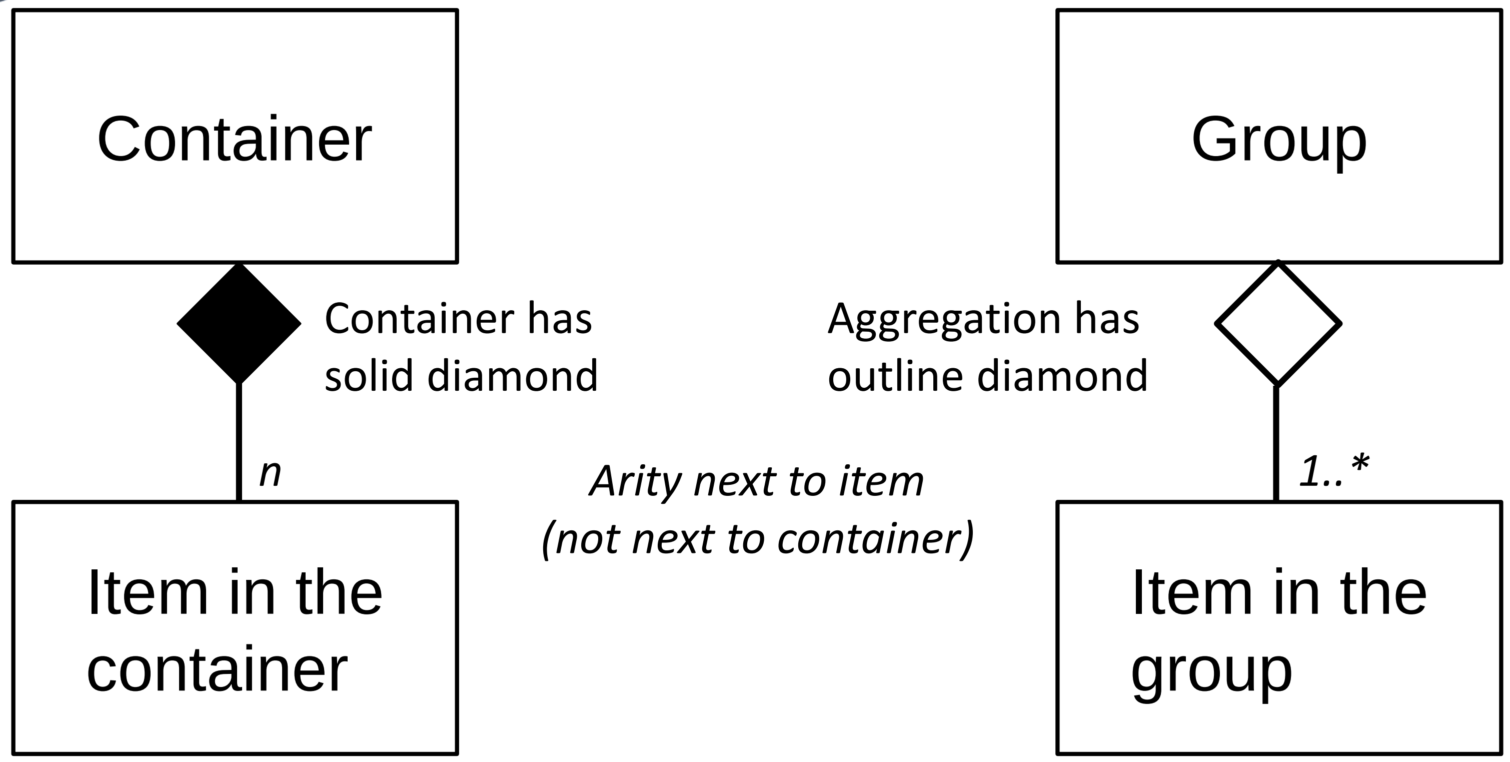
Solutions





- A containment relation where a bus contains one engine
- An aggregation relation showing a bus can group zero or more passengers

Containment vs. Aggregation


Container
Container has solid diamond
n
Item in the container
Group
Aggregation has outline diamond
1..*
Item in the group
Arity next to item (not next to container)

<u>Key to arity:</u> *n → exactly n.   1..* → 1 or more   * = zero or more*

End of Lecture

The Next Episode…

Lecture P02

P02: Operating systems & introduction to the embedded Linux design. (Part 1)