Software Design & Architecture

Engr. Abdul-Rahman Mahmood

DPM, MCP, QMR(ISO9001:2000)

armahmood786@yahoo.com
alphapeeler.sf.net/pubkeys/pkey.htm
pk.linkedin.com/in/armahmood
www.twitter.com/alphapeeler
www.facebook.com/alphapeeler
abdulmahmood-sss alphasecure
armahmood786@hotmail.com
http://alphapeeler.sf.net/me

alphasecure@gmail.com
http://alphapeeler.sourceforge.net
http://alphapeeler.tumblr.com
armahmood786@jabber.org
alphapeeler@aim.com
mahmood_cubix 48660186
alphapeeler@icloud.com
http://alphapeeler.sf.net/acms/

Advanced USE Cases

Use Case 2.0

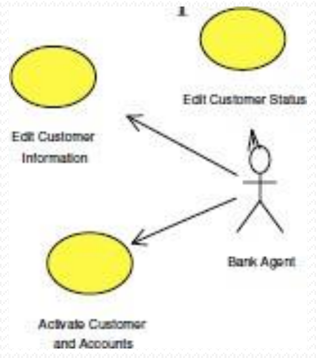
- **Agenda**
- Use case vocabulary
- Some use case key points
- Mis-Use cases
- Use Case Maps
- Summary
  - Use Cases benefits
  - What use cases cannot do

Vocabulary

- Actor – Role(s) external parties that interact with the system
  - stimulates the system to react *(primary actor)*
    - *You normally don't have control over primary actors*
  - responds to the system's requests *(secondary actor)*
    - *normally used by the system to get the job done*

Vocabulary

- Use Case
  - A formal way of representing how a business system interacts with its environment
  - A sequence of actions a system performs that yields a valuable result for a particular actor.
- Use Case Model
  - Bag that contains:
    - Actors list, packages, diagrams, use cases, views



Use case descriptions

Concepts

Constraints and requirements

Places to Look for Actors

- Who uses the system?
- Who gets information from this system?
- Who provides information to the system?
- What other systems use this system?
- Who installs, starts up, or maintains the system?

Finding Actors

- Focus initially on human and other primary actors
- Group individuals according to their common tasks and system use
- Name and define their common role
- Identify systems that initiate interactions with the system
- Identify other systems used to accomplish the system's tasks

Finding Use Cases

- Describe the functions that the user will want from the system
- Describe the operations that create, read, update, and delete information
- Describe how actors are notified of changes to the internal state of the system
- Describe how actors communicate information about events that the system must know about

Advance Use Cases

- Use-Case 2.0: A scalable, agile practice that uses use cases to capture a set of requirements and drive the incremental development of a system to fulfill them.
- <u>Drives development of a system</u> by first helping you understand how the system will be used
- Helping you <u>evolve an appropriate system</u>.

First Principles

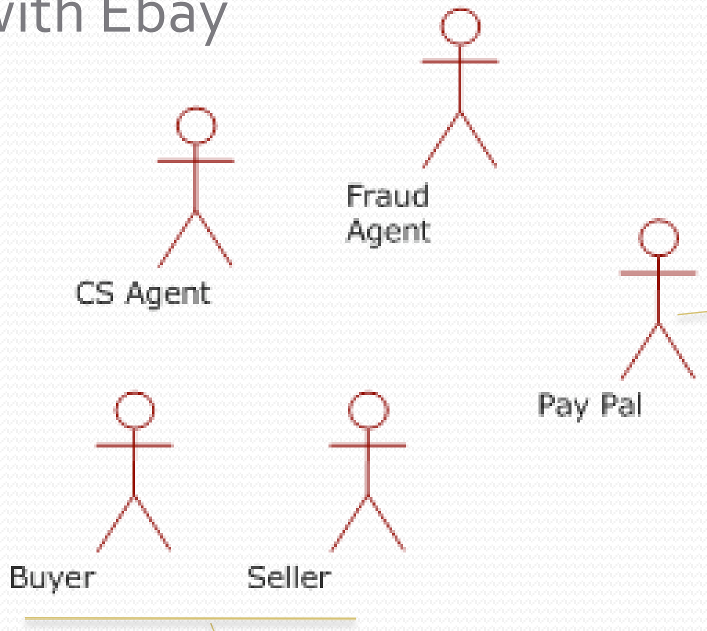
- There are six basic principles at the heart of any successful application of use cases:
- 1. Keep it simple by telling stories
- 2. Understand the big picture
- 3. Focus on value
- 4. Build the system in slices
- 5. Deliver the system in increments
- 6. Adapt to meet the team's needs

Key point 1

- Building use cases is an iterative process
  - You usually don’t get it right at the first time.
  - Developing use cases should be looked at as an iterative process where you work and refine.
  - Involve the stakeholder in each of the iteration

Key point 2

- Define use case actors
  - UML visual notations are commonly used
  - Start by defining key actors:
    - There are possibly over a dozen actors that interact with Ebay

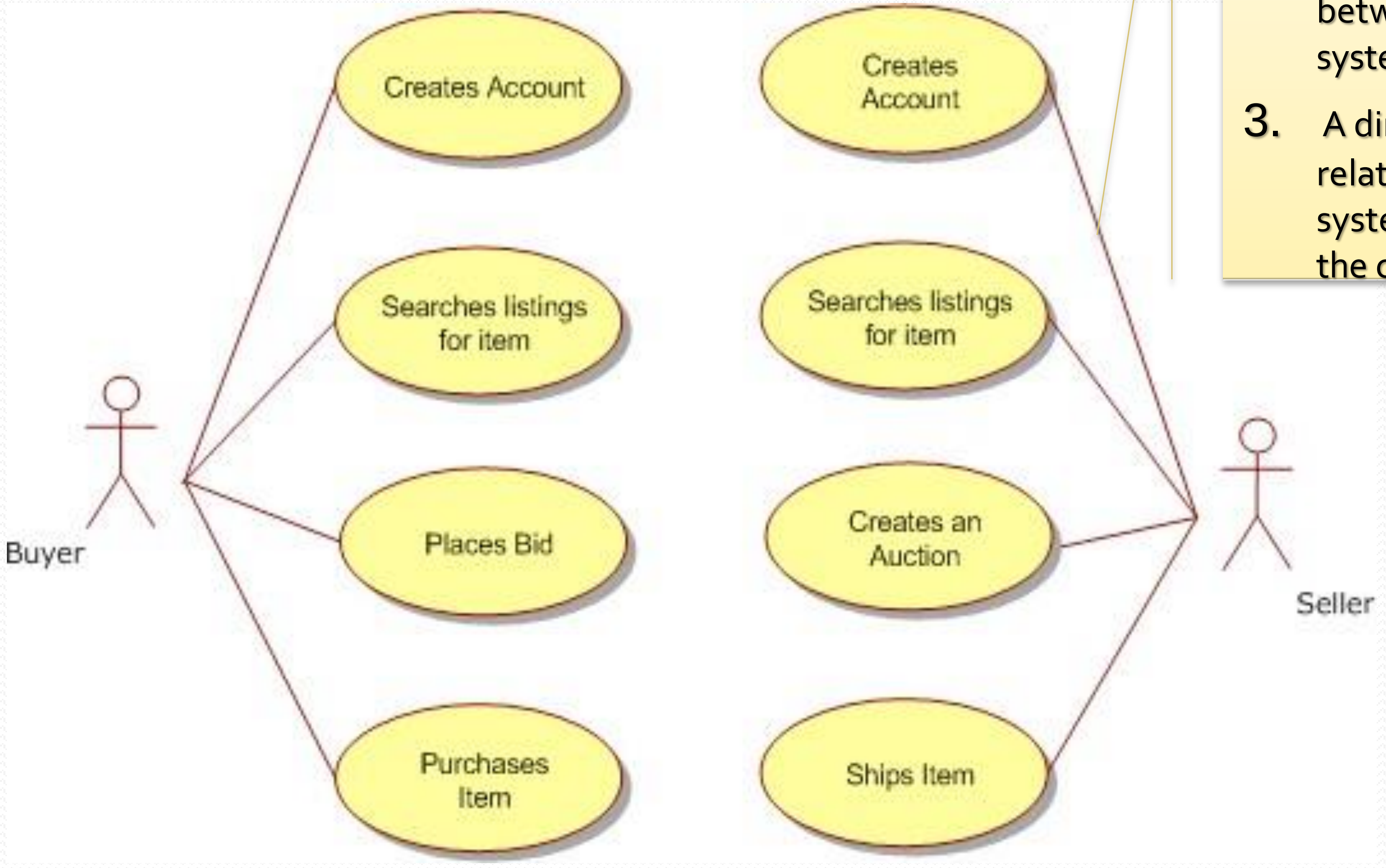


An actor can be a system because the system plays another role in the context of your new system and also interact with other actors

Key users

Key point 3

- Define your use case Actor Goals

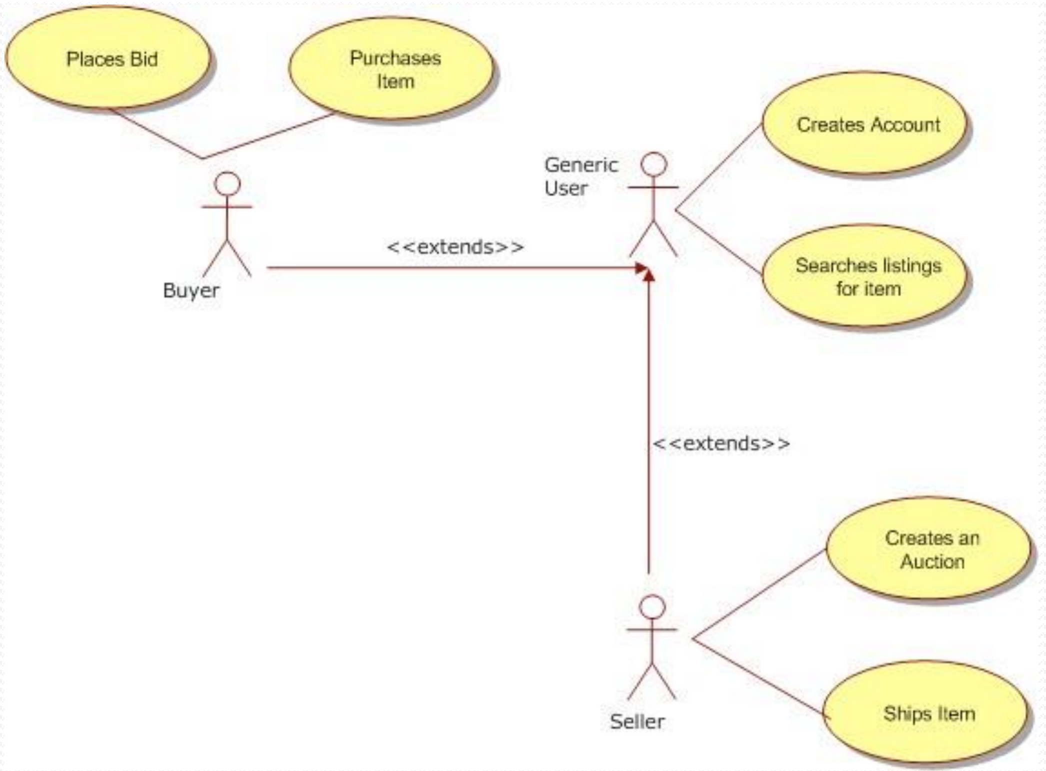


Note:

1. Association relationships only show which actors interact with the system to perform a given use case.
2. Association relationship **DO NOT** model the flow of data between the actor and the system.
3. A directed association relationship only shows if the system or the actor initiates the connection.

Key point 4a

- Identify reuse opportunity for use cases
  - There is **duplicate behavior** in both the buyer and seller which includes "**create an account**" and "**search listings**".
  - **Extract a more general user** that has the duplicate behavior and then the actors will "inherit" this behavior from the new user.



Key point 4b

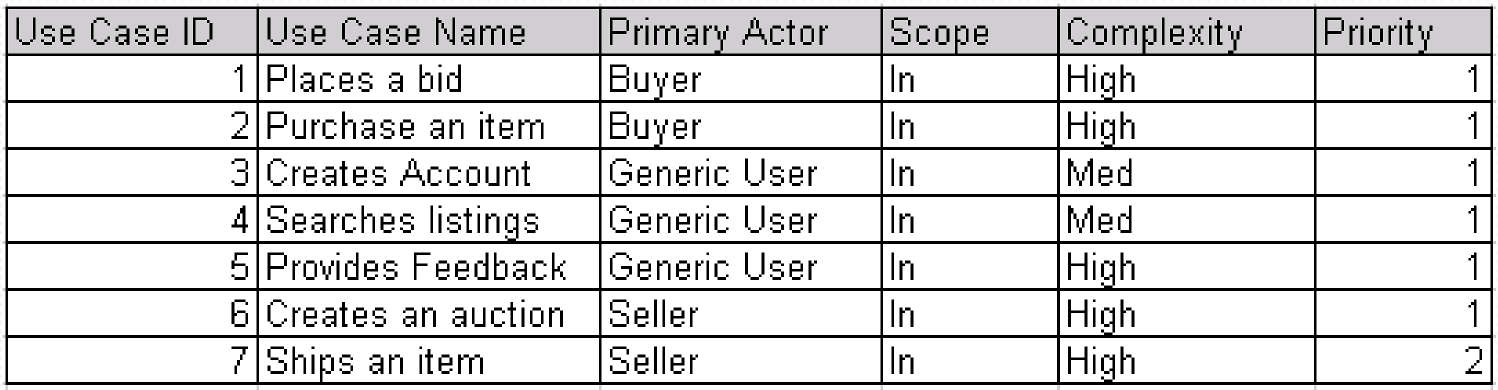
- Identify reuse opportunity for use cases
  - UML defines these relationships between use cases:
    - **Dependency**—The behavior of one use case is affected by another
      - Being logged into the system is a pre-condition to performing online transactions. **Make a Payment depends on Log In**
    - **Includes**— One use case incorporates the behavior of another at a specific point
      - **Make a Payment** includes **Validate Funds Availability**

Key point 4c

- Identify reuse opportunity for use cases
  - UML defines these relationships between use cases:
    - Extends— One use case extends the behavior of another at a specified point
      - **Make a Payment in installments** and **Make a Fixed Payment**
      - both extend **the Make a Payment** use case
    - Generalize—<u>One use case inherits the behavior of another</u>; it can be used interchangeably with its "parent" use case
      - **Check Password and Retinal Scan generalize Validate User**

Key point 5

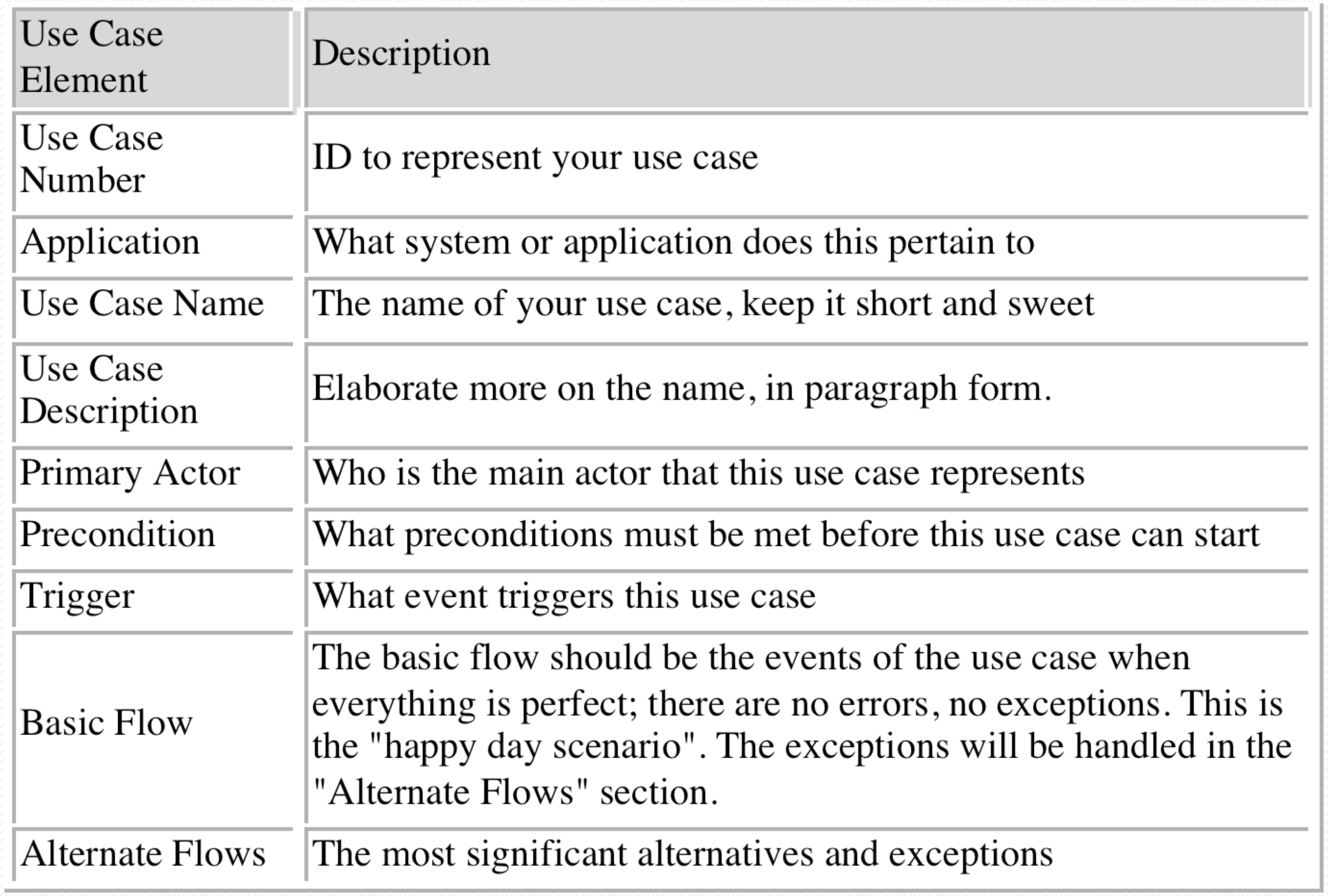
- Create a use case index
  - Every use case has various attributes relating both to the use case itself and to the project
  - At the project level, these attributes include scope, complexity, status and priority.



Use Case ID	Use Case Name	Primary Actor	Scope	Complexity	Priority
1	Places a bid	Buyer	In	High	1
2	Purchase an item	Buyer	In	High	1
3	Creates Account	Generic User	In	Med	1
4	Searches listings	Generic User	In	Med	1
5	Provides Feedback	Generic User	In	High	1
6	Creates an auction	Seller	In	High	1
7	Ships an item	Seller	In	High	2

Key point 6a

- Identify the key components of your use case
  - The actual use case is a textual representation



Use Case Element	Description
Use Case Number	ID to represent your use case
Application	What system or application does this pertain to
Use Case Name	The name of your use case, keep it short and sweet
Use Case Description	Elaborate more on the name, in paragraph form.
Primary Actor	Who is the main actor that this use case represents
Precondition	What preconditions must be met before this use case can start
Trigger	What event triggers this use case
Basic Flow	The basic flow should be the events of the use case when everything is perfect; there are no errors, no exceptions. This is the "happy day scenario". The exceptions will be handled in the "Alternate Flows" section.
Alternate Flows	The most significant alternatives and exceptions

Key point 6b

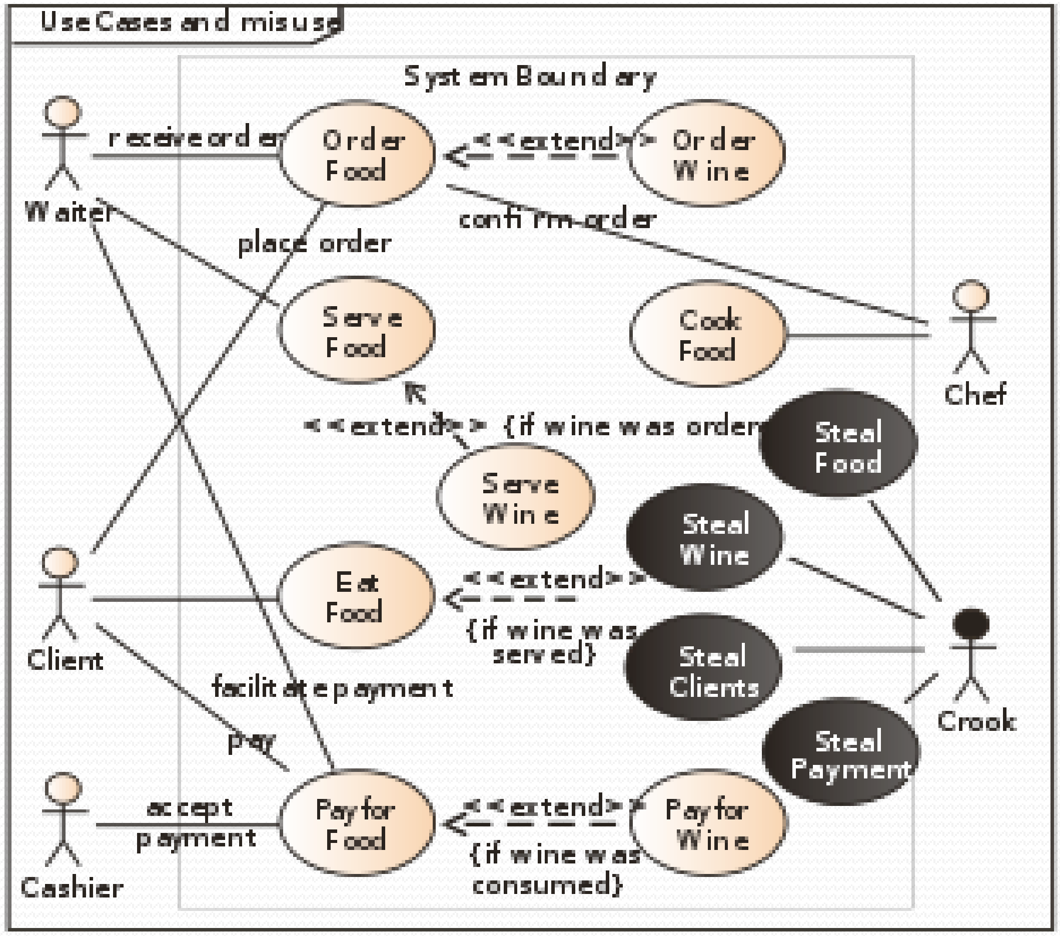
- Identify the key components of your use case
  - Examples of alternative flow:
    - While a customer places an order, their credit card failed
    - While a customer places an order, their user session times out
    - While a customer uses an ATM machine, the machine runs out of receipts and needs to warn the customer

Mis-Use cases

- Aims to Identify the possible misuse scenarios of the system
- The basic concept describes the steps of performing a malicious act against a system
- The process is the same as you would describe an act that the system is supposed to perform in a use case

Mis-Use cases

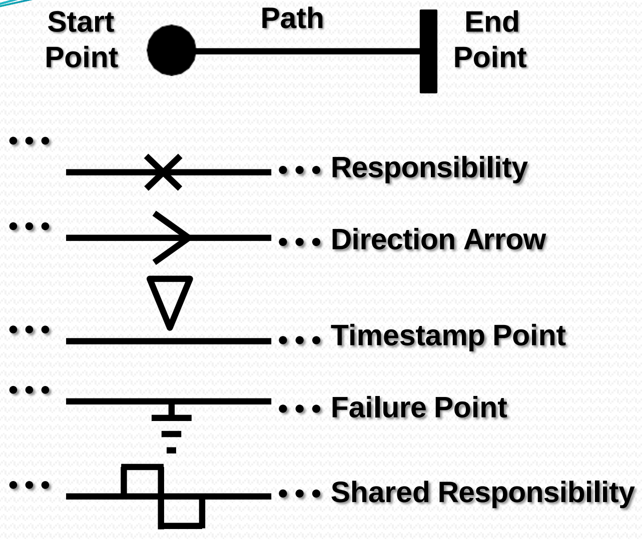
- Mostly used in capturing security requirements



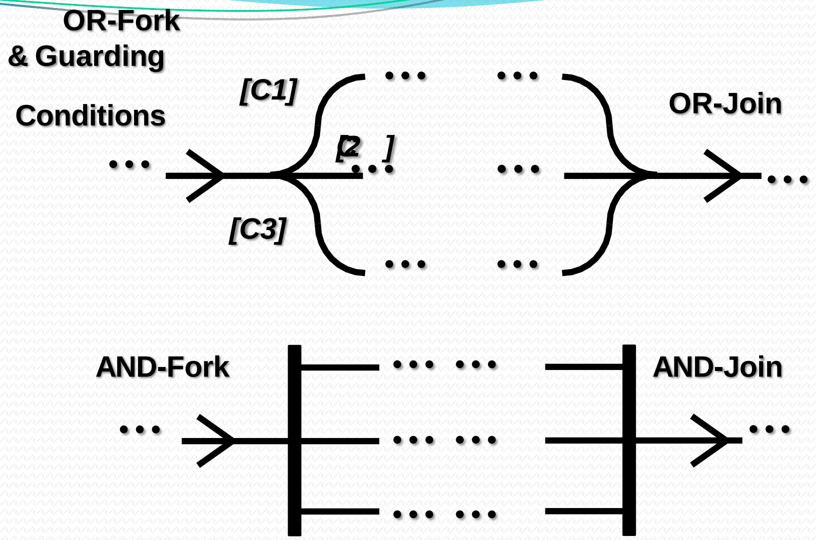
Use Case Maps

- *Definition*:
  - A visual representation of the requirements of a system, using a precisely defined set of symbols for responsibilities, system components, and sequences.
- Links behavior and structure in an explicit and visual way
- UCM *paths:*
  - Architectural entities that describe causal relationships between responsibilities, which are bound to underlying organizational structures of abstract components
  - UCM paths are intended to bridge the gap between requirements (use cases) and detailed design

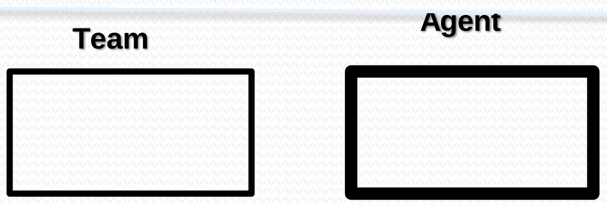
Use case Maps: *Notations*



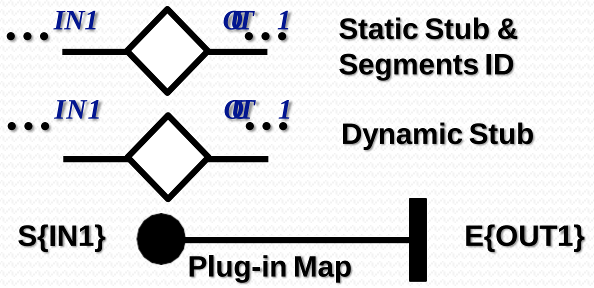
(a) UCM Path Elements



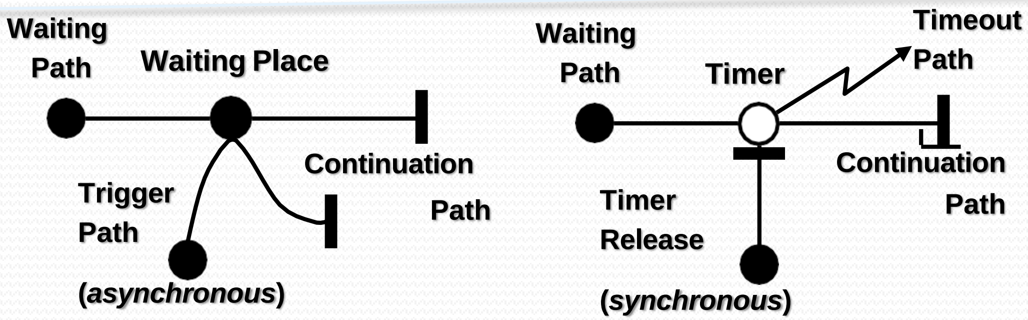
(b) UCM Forks and Joins



(c) UCM Components



(d) UCM Stubs and Plug-ins

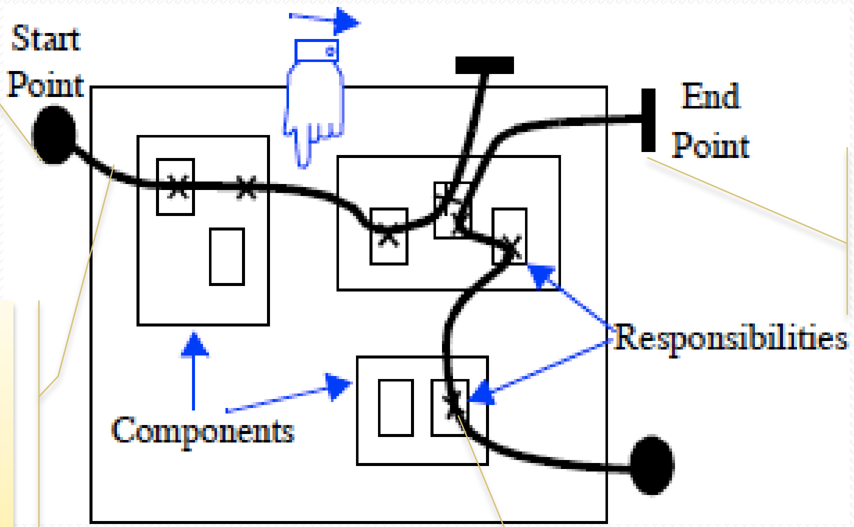


(e) UCM Waiting Places and Timers

Use Case Maps:

- Mainly consist of path elements and components

Contains pre-conditions or triggering causes)



Example of Use Case map

path traces through a system of objects to explain a causal sequence, leaving behind a visual signature.

bars representing post-conditions or resulting effects

- causal chains of responsibilities (crosses, representing actions, tasks, or functions to be performed)
- Responsibilities are normally bound to component when the cross is inside the component

- A component is responsible to perform the action, task, or function represented by the responsibility.
- Start points may have preconditions attached, while responsibilities and end points can have postconditions.

Summary

- Use Cases benefits:
  - Promote customer involvement
  - Use cases describe a system from an external usage perspective
  - They can be organized according to their relevance, frequency of use, and perceived value to the system's users
  - System features can be correlated with how they are used within specific use cases
  - Impacts of adding and/or removing features on system usability can be analyzed

What Use Cases Cannot Do

- Use Cases are best used to describe system functionality from a task-oriented perspective
- They do not describe:
  - user interfaces
  - performance goals
  - application architecture
  - non-functional requirements

References

- https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case_2_0_jan11.pdf