SOFTWARE ENGINEERING MODELS AND MODELING

ANTTI KNUTAS (D.SC.)

Open your mind. LUT.
Lappeenranta University of Technology

REQUIREMENTS AND BCE/MVC MODELS

LECTURE 4

Open your mind. LUT.
Lappeenranta University of Technology

UML 2 Chart types

Structure diagrams

- **Class diagram**
- **Component diagram**
- **Composite structure diagram** (added in UML 2.x)
- **Deployment diagram**
- **Object diagram**
- **Package diagram**

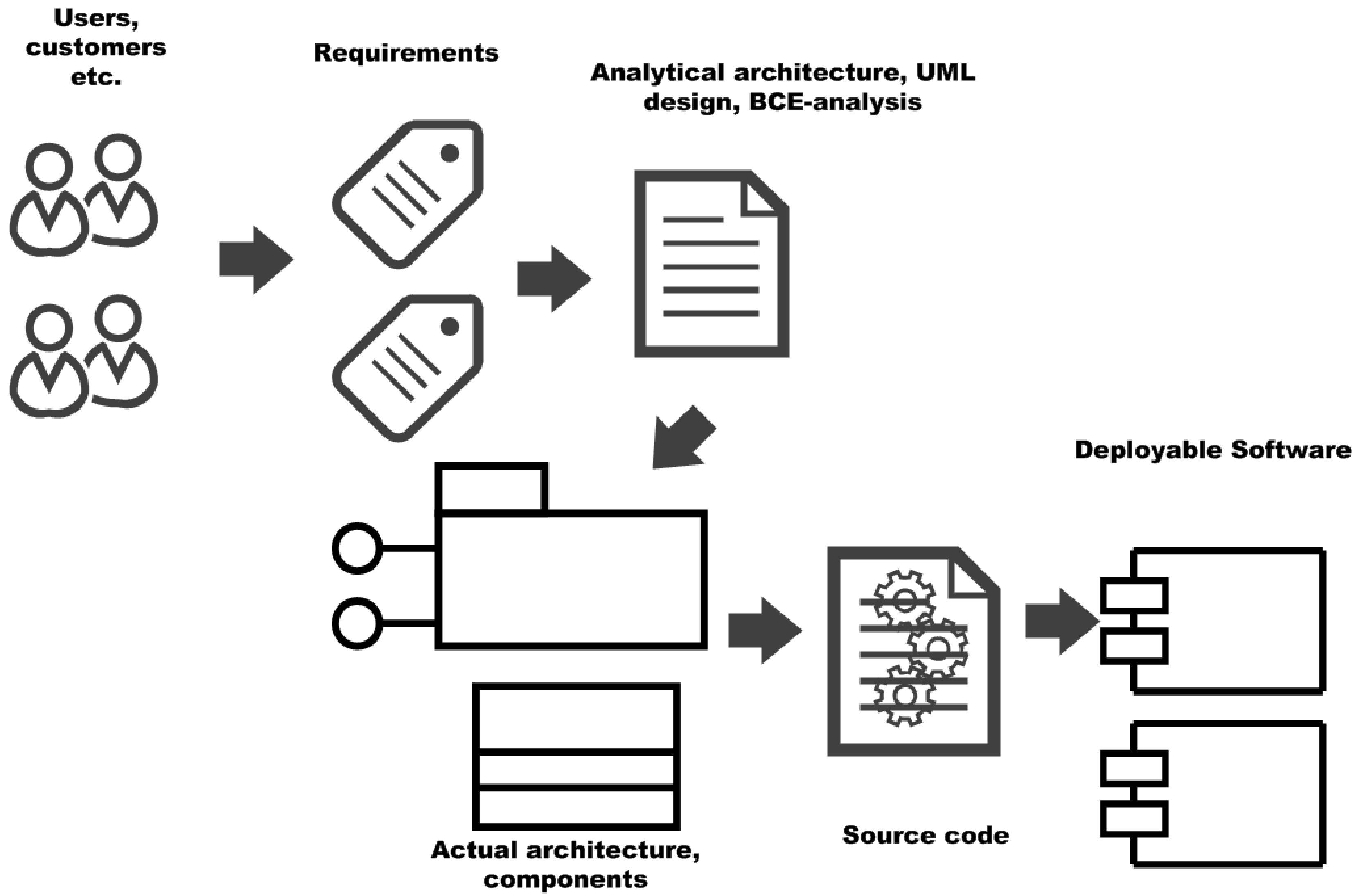
Behavior diagrams

- **Activity diagram**
- **State Machine diagram** (*was Statechart diag. in UML1*)
- **Use case diagram**

Interaction diagrams

- **Communication diagram** (was *Collaboration diag. in UML1)*
- **Interaction overview diagram** (added in UML 2.x)
- **Sequence diagram**
- **Timing diagram** (added in UML 2.x)

FROM REQUIREMENTS TO IMPLEMENTATION


Users, customers etc.
Requirements
Analytical architecture, UML design, BCE-analysis
Deployable Software
Actual architecture, components
Source code

SUMMARY

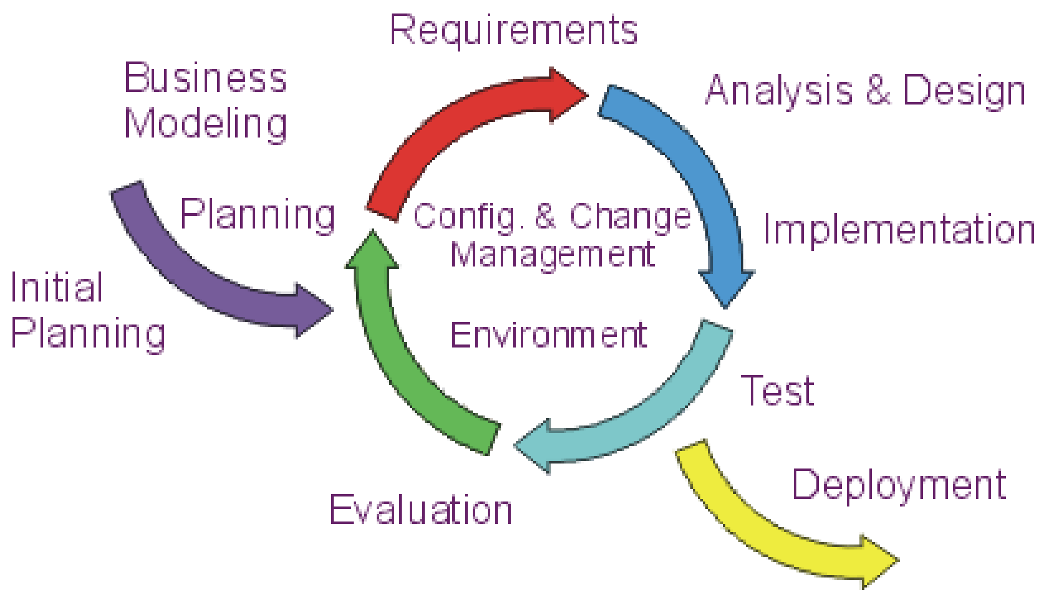
All models have similar ideas:

-Collect information before design

-Design before implementation

-Validate direction and quality during implementation

-Test before release



REQUIREMENTS

GATHERING AND ANALYSIS

WHAT IS A REQUIREMENT

A requirement is a condition or functionality, that the system must meet or implement

Requirements can be classified to functional and non-functional requirements

- Functional requirements define the functionalities needed by the user
- Non-functional requirements are more general requirements (e.g. performance, security, …)

TERMINOLOGY

Discovering the client's requirements

- *Requirements elicitation* (or *requirements capture*)
- Methods include interviews and surveys

Refining and extending the initial requirements

- *Requirements analysis*

METHODS OF REQUIREMENTS ANALYSIS AND ELICITATION

Typical examples

- Stakeholder interviews
- Joint sessions with stakeholders
    - Facilitator, business analyst
- Requirements list as a contract
    - Commercial contract between parties
- Prototypes
    - For illustrating requirements
- Use cases
- Requirements specification document

TYPICAL PROBLEMS OF REQUIREMENTS ANALYSIS

Users do not understand what they want

Users do not commit to the written requirements

Users want new requirements after the old ones are fixed

Users do not participate in elicitation or reviews

Users do not understand technology

Users do not understand the development process

Developers do not understand users' terminology

Developers want to force their own views

Analysis is done by technical experts instead of domain experts or persons with people skills

ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

Architecturally significant requirement: a requirement that has an effect on the basic structure and qualities of the system

- Often high-risk, high-priority or low-stability requirement

Examples

- Product localization
- Persistency management
- 7/24 operation
- Real-time low-latency operations
- Distributed, fault-tolerant global operation

CLASSIFYING REQUIREMENTS: FURPS+

Requirements can be further classified using FURPS+ model

- Functionality
- Usability
- Reliability
- Performance
- Supportability

+ in the model means requirements that relate e.g. to

- design constraints
- implementation requirements
- interface requirements
- physical requirements

FUNCTIONALITY

Functional requirements define those actions and functions that the system can execute

Functional requirements are presented with use cases

Functional requirements define how the system behaves with inputs and outputs

Table: examples of architecturally significant functional requirements

Function	**Description**
Auditing	Provide audit trails of system execution.
Licensing	Provide services for tracking, acquiring, installing, and monitoring license usage.
Localization	Provide facilities for supporting multiple human languages.
Mail	Provide services that allow applications to send and receive mail.
Online help	Provide online help capability.
Printing	Provide facilities for printing.
Reporting	Provide reporting facilities.
Security	Provide services to protect access to certain resources or information.
System management	Provide services that facilitate management of applications in a distributed environment.
Workflow	Provide support for moving documents and other work items, including review and approval cycles.

NON-FUNCTIONAL REQUIREMENTS

Many requirements are non-functional by nature

- They describe general properties, constraints or requirements of the system or its environment
- Often the non-functional requirements are described in a separate document
- Non-functional requirements are typically architecturally significant

Non-functional requirements classes

- Usability
- Reliability
- Performance
- Supportability

+

- design constraints
- implementation requirements
- interface requirements
- physical requirements

USABILITY

Usability requirements can be associated e.g. with

- Human attributes
- Esthetics
- Consistency of the user interface
- On-line helps
- Wizards and agents
- User documentation
- Training material

RELIABILITY

Reliability requirements may be related to

- Frequency/Severity of Failure
- Recoverability
- Predictability
- Accuracy
- Mean Time to Failure (MTBF)
- Software Criticality

PERFORMANCE

Performance requirements can be directly related to functions or they can be more general

- Speed
- Efficiency
- Resource consumption
- Throughput
- Response time
- Recovery time

SUPPORTABILITY

Supportability may include

- Testability
- Extensibility
- Adaptability
- Maintainability
- Configurability
- Serviceability
- Installability
- Localization

OTHER NON-FUNCTIONAL REQUIREMENTS (FURPS+)

Design Requirements

- General requirements about systems analysis and design (like the methods used)

Implementation Requirements

- Constraints for system implementation, like
  - Used standards
  - Required programming languages
  - DBMS
  - Resource limitations
  - Supported operating systems

Interface Requirements

- Formats, timing constraints, protocols, etc. related to external systems and devices

Physical Requirements

- Physical attributes of the system, like
  - Material, shape, size, weight
  - Also the constraints and requirements related to the hardware and physical network

LISTING OF REQUIREMENTS CANDIDATES

List of all properties required from the system

List may include wishes and ideas that will probably not be implemented

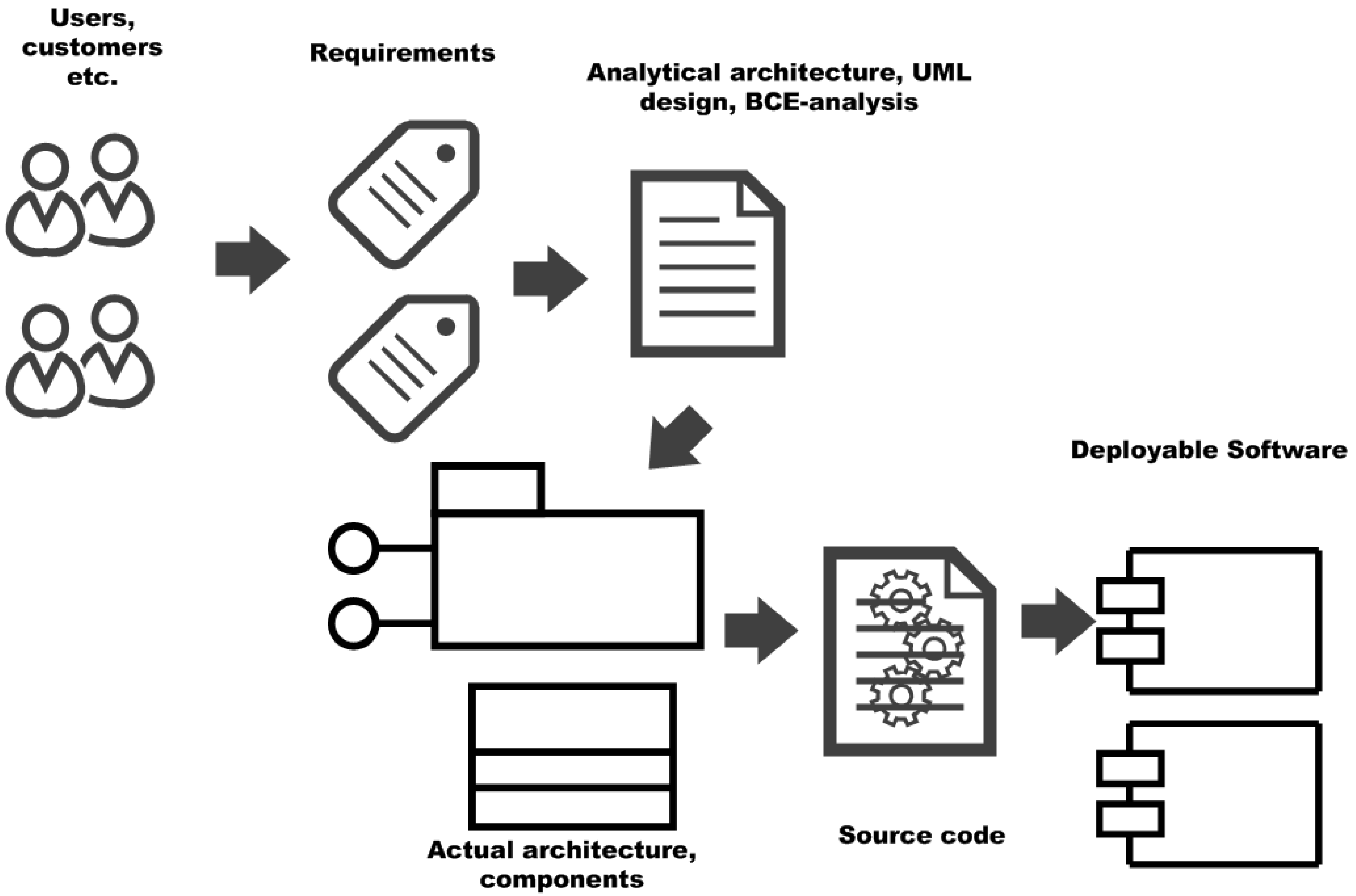
- Stakeholder needs
- System features
- Software requirements

In addition to the property itself, the list contains

- The status of the property (e.g. Proposed, accepted, etc.)
- Cost estimate (e.g. Working hours)
- Priority
- Risk evaluation

MAKING USE CASES OUT OF REQUIREMENTS

FROM REQUIREMENTS TO IMPLEMENTATION


Users, customers etc.
Requirements
Analytical architecture, UML design, BCE-analysis
Deployable Software
Actual architecture, components
Source code

SPECIFICATION OF THE ENVIRONMENT OF THE SYSTEM

Conceptual modeling (domain/business modeling)

- Description of business concepts

The result is a class diagram that describes business concepts

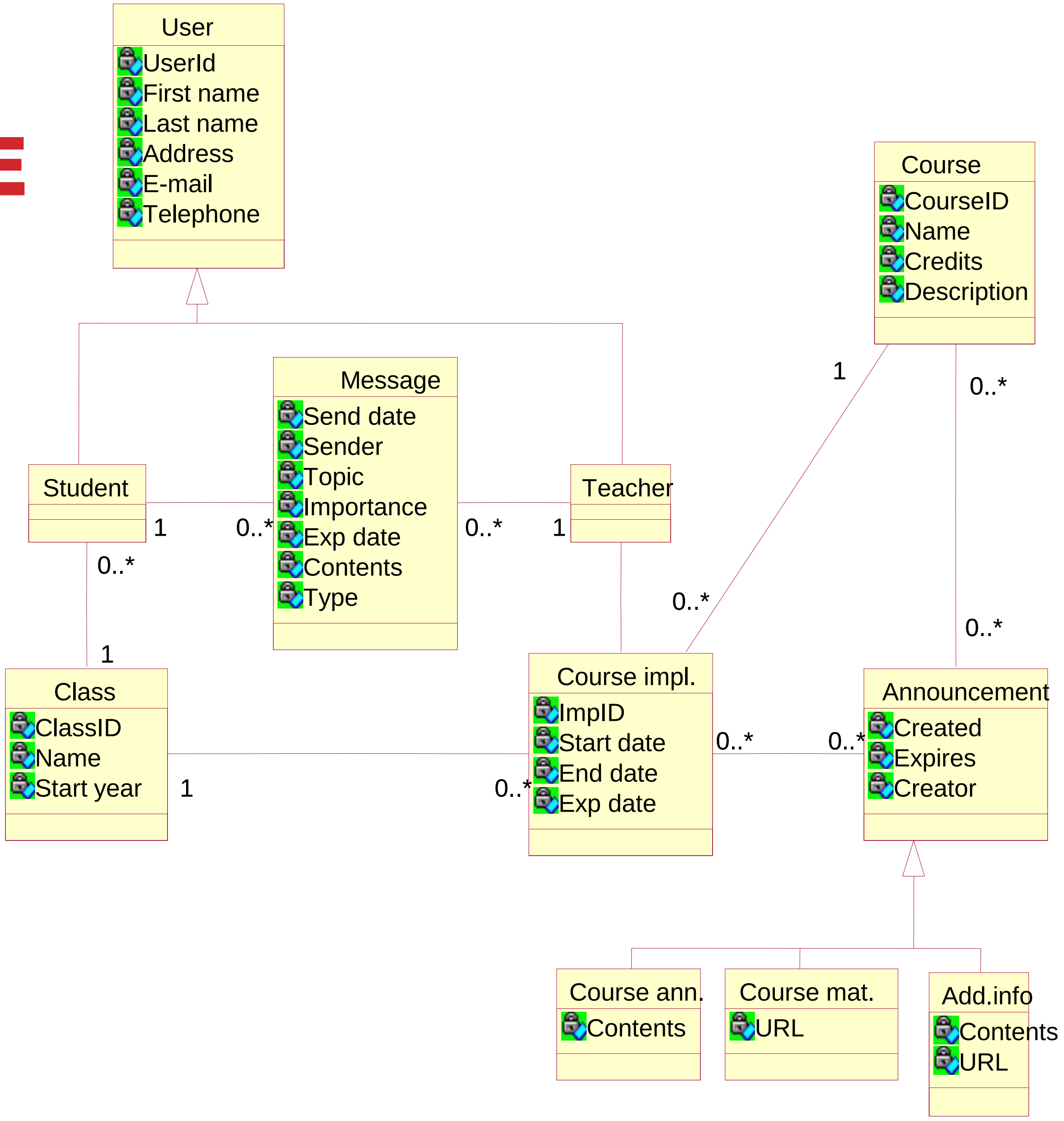
In addition, descriptions of concepts are made to the glossary

The domain/business model describes external concepts – not the internal data structures of the system

In addition a functional model of the business may be made

- Definition of business processes in system’s domain

EXAMPLE


User
UserId
First name
Last name
Address
E-mail
Telephone
Course
CourseID
Name
Credits
Description
Message
Send date
Sender
Topic
Importance
Exp date
Contents
Type
Student
Teacher
1
0..*
0..*
1
0..*
1
1
0..*
0..*
0..*
Class
ClassID
Name
Start year
1
0..*
Course impl.
ImpID
Start date
End date
Exp date
0..*
0..*
Announcement
Created
Expires
Creator
Course ann.
Contents
Course mat.
URL
Add.info
Contents
URL

GATHERING AND DESCRIPTION OF FUNCTIONAL REQUIREMENTS

Actors and use cases are identified

- Use cases are prioritized
- Use case descriptions are made
- User interface prototypes can be used

Identification of actors and use cases

- Input artifacts
    - Domain/business model and glossary
    - Non-functional requirements
    - Requirements candidates
- Output artifacts
    - Use case model and use case descriptions
    - Glossary

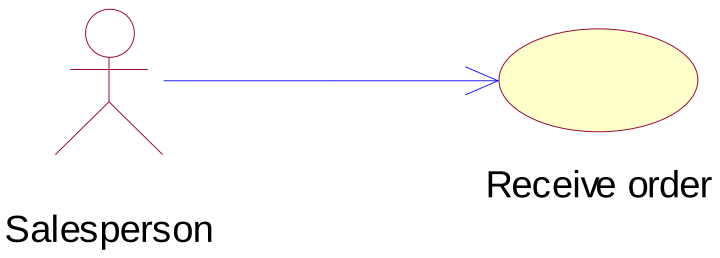
GATHERING AND DESCRIPTION OF FUNCTIONAL REQUIREMENTS (CONTINUED)

Purpose is to

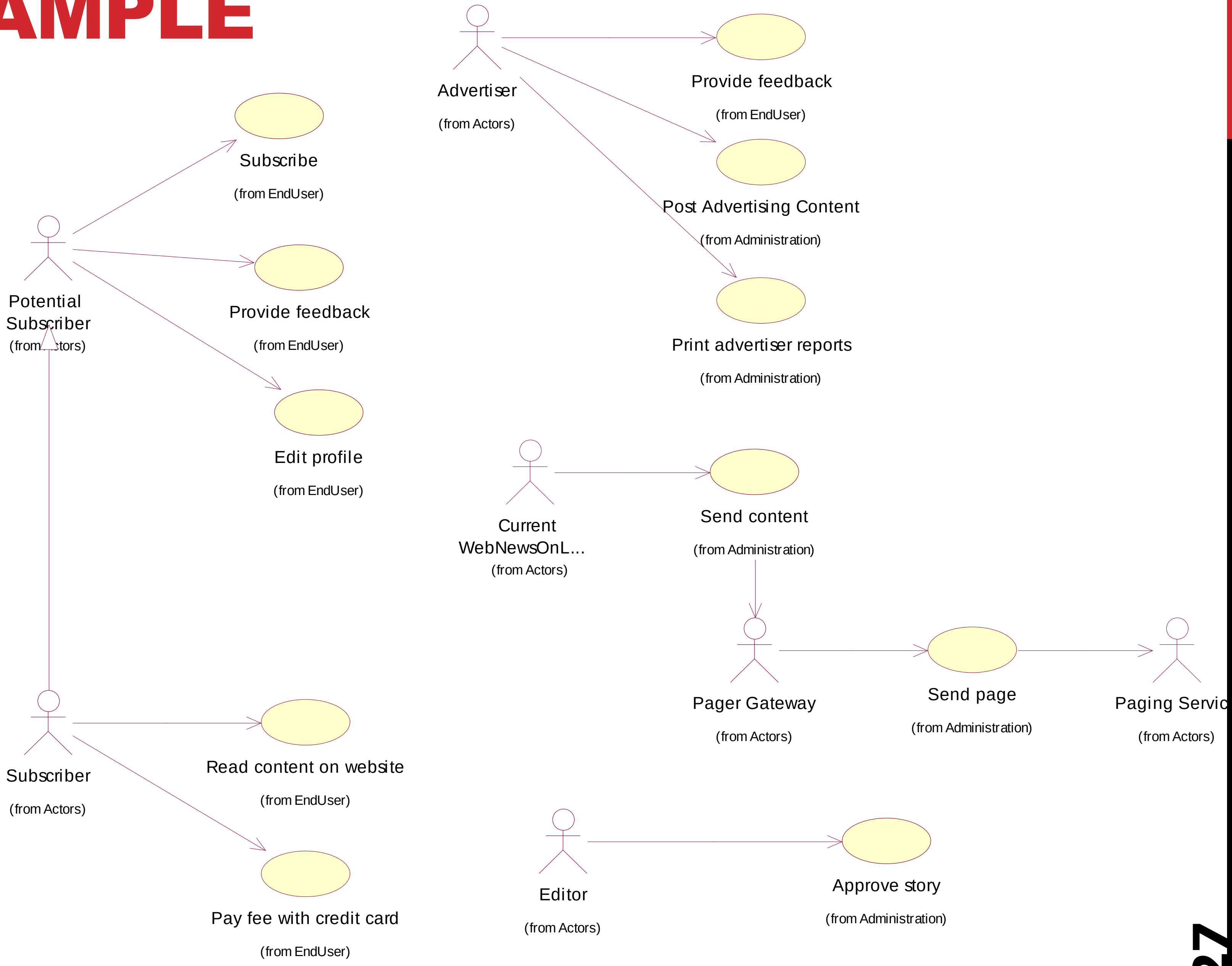
- Define the borders of the system
- Define who and what are interfacing the system and what is expected from the system
- Find out common terms and glossary of the system

Concepts

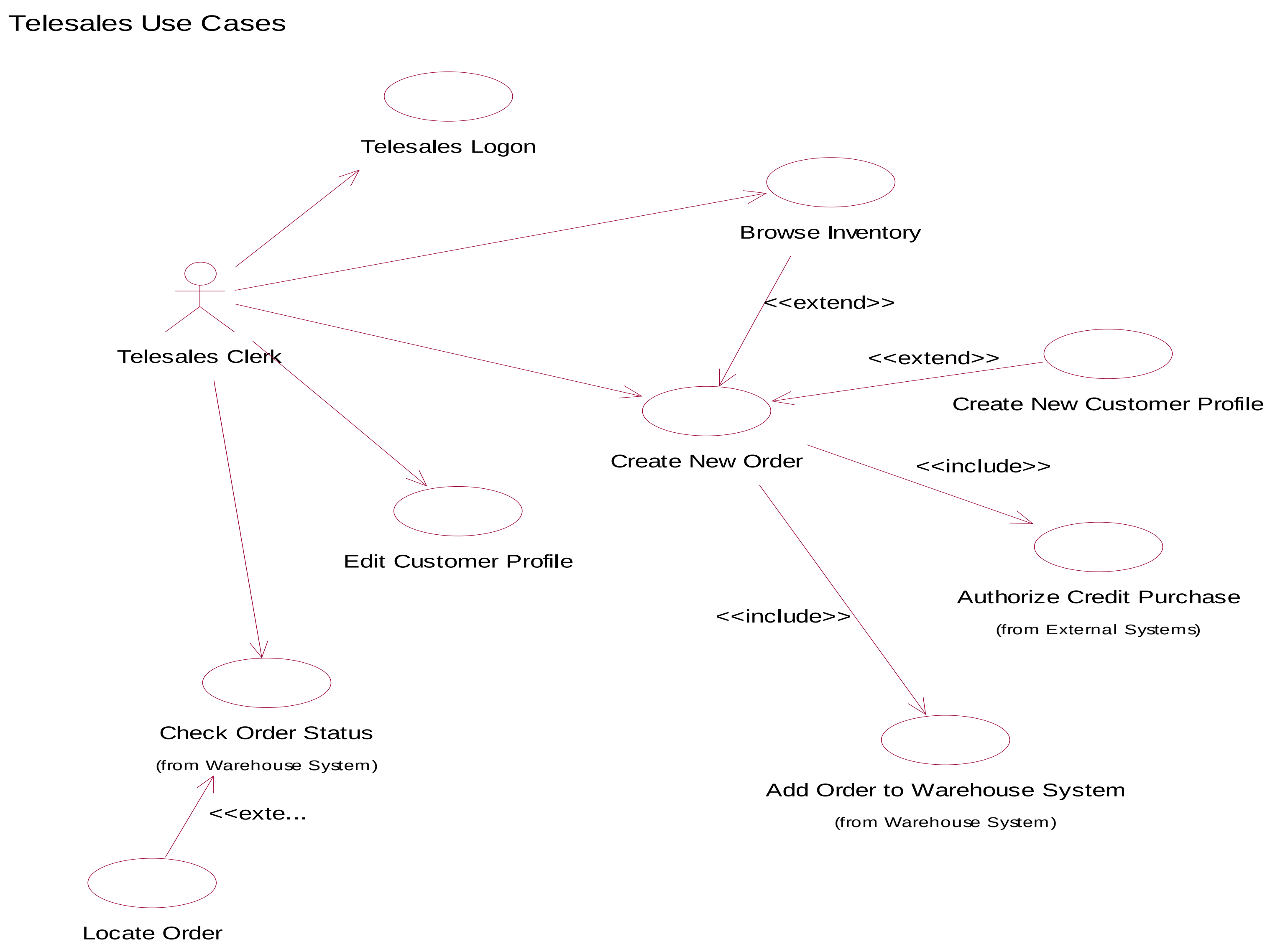
- Actor
  - Describes a role, not a particular person (a person may have several roles)
  - Can also be a device or another system that has interaction with the system
- Use case
  - A symbol for a certain flow of events
  - Typically a function of the system that is executed as a whole
  - Is connected to actors



EXAMPLE


Advertiser
(from Actors)
Provide feedback
(from EndUser)
Post Advertising Content
(from Administration)
Print advertiser reports
(from Administration)
Subscribe
(from EndUser)
Potential Subscriber
(from Actors)
Provide feedback
(from EndUser)
Edit profile
(from EndUser)
Current WebNewsOnL...
(from Actors)
Send content
(from Administration)
Pager Gateway
(from Actors)
Send page
(from Administration)
Paging Servic
(from Actors)
Subscriber
(from Actors)
Read content on website
(from EndUser)
Pay fee with credit card
(from EndUser)
Editor
(from Actors)
Approve story
(from Administration)

ANOTHER EXAMPLE



PRIORITIZATION OF USE CASES

Definition of what use cases are designed immediately and what can be done later

- Testability

Architecturally important use cases

- Define system architecture
- Must be designed in detail already in the beginning because changes to architecture are very expensive
- Also possible candidates for high-risk modules which are first to be implemented.

DESCRIPTION OF USE CASES

Use case diagram

- A free form description of the functionality of the system
- May be divided to several diagrams

Use case descriptions

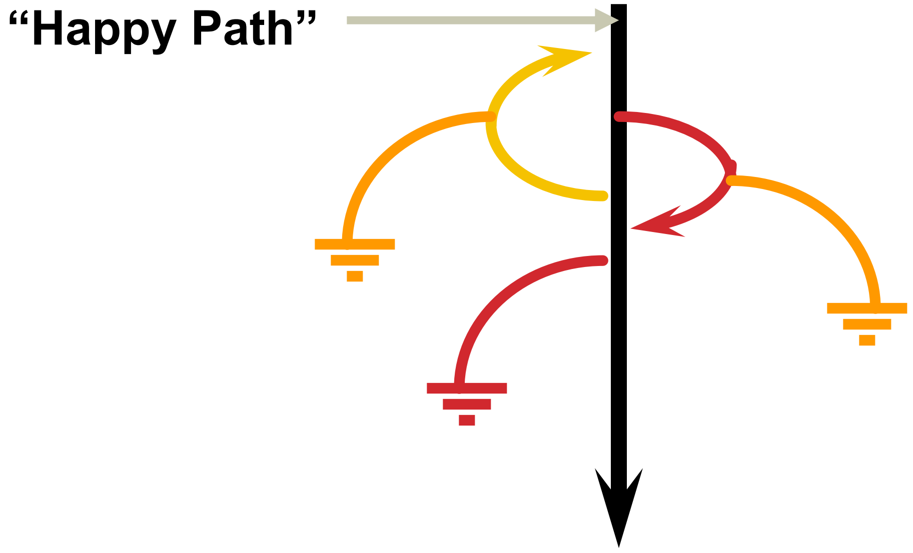
- First, a rough description about the use case is made
- Later the description is detailed so that the function is described explicitly step by step
- All variations of the flows of events are described

USE CASE FLOWS OF EVENTS

Typically a use case has one normal flow of event ("Happy Path")

Many alternative flows of events

- Recurring alternatives
- Exceptions
- Error situations



USE CASE DESCRIPTIONS

Contents of each use case description

- Start state: what are the preconditions of the function?
- How the use case starts?
- In what order the use case is executed?
- How the use case ends? All alternatives are described
- Alternative flows of events
- Illegal flows of events
- What actors participate and what information is exchanges?
- What kind of data and resources are used?
- Other non-functional requirements (e.g. Performance, response time, security, etc.)

RUP USE CASE TEMPLATE

1. Brief Description
A brief description of the use case is included here.

2. Flow of Events
The flow of events of the use case is included here. Only one level of sub flows is indicated, but you may add more levels if necessary.

2.1 Basic Flow of Events

2.1.1 <name of sub flow>

2.2 Alternative Flow of Events

2.2.1 <name of sub flow>

3. Special Requirements
Special requirements of the use case.

3.1 <name of special requirement>
A brief description of the special requirement.

4. preconditions
preconditions of the use case.

4.1 <name of precondition>
A brief description of the precondition.

5. postconditions
postconditions of the use case.

5.1 <name of postcondition>
A brief description of the postcondition.

6. Extension Points
Extension points of the use case.

6.1 <name of extension point>
Definition of the location of the extension point in the flow of events.

7. Relationships
The relationships involving the use case are included here. For communicates-associations, a brief description, multiplicity, and associated actors are included. Also, the navigability of the use-case role is included. For include- and extend-relationships, a brief description and associated use cases are included.

8. Use-Case Diagrams
Use-Case Diagrams local to the use case.

9. Other Diagrams
Other graphs that illustrate the use case.

COLLEGIATE SPORTS PAGING SYSTEM USE CASE SPECIFICATION: POST ADVERTISING CONTENT

Brief Description

This use case occurs when an advertiser wants to post advertising content (banner ads) on the web site and specify which subscriber profiles should be used for display.

Flow of Events

Basic Flow

1.Advertiser selects "Post Content"

2.System validates account billing information to ensure new content will be accepted. This consists of verifying that the advertiser account is in good standing (no outstanding balance over 30 days).

3.System prompts for name of content.

4.Advertiser uploads content in GIF format.

5.System stores GIF in a staging area and records location and name of file.

6.System acknowledges successful saving of content.

7.System displays potential categories for ad display. Categories are maintained as a reference list within the system.

8.Advertiser selects categories for which this ad should be shown.

9.System displays potential frequencies and prices for the ad. Frequencies are maintained as a reference list, prices are based on advertiser's contract with WebNewsOnLine. Pricing codes are maintained as part of the advertiser profile.

10.Advertiser selects desired frequency for this ad

11.System creates preliminary billing record for this ad based on number of categories selected, frequency, and advertiser's profile.

12.System places content in editor's "to-do" workflow queue for approval.   If more than one editor is defined, a round-robin approach is used to attempt to balance load.   Editors may be marked as unavailable (if, for instance, they are on vacation or sick or out of the office), in which case they will not be included in the round-robin process.

COLLEGIATE SPORTS PAGING SYSTEM: USE CASE SPECIFICATION: POST ADVERTISING CONTENT (2)

Alternative Flow

Invalid Account Information

1. Advertiser selects "Post Content"
2. System validates account billing information to ensure new content will be accepted
3. Account information is invalid, advertiser is prompted to contact WebNewsOnLine. System presents central phone number for advertising department (stored in system).

Special Requirements

None.

Preconditions

User is connected and validated as an advertiser.

Advertiser account exists.

Postconditions

When this use case is complete, advertising content can be displayed.

Extension Points

None.

EXAMPLE 2: COURSE REGISTRATION

Close Registration Use Case

Brief Description

This use case allows a Registrar to close the registration process. Course offerings that do not have enough students are cancelled. Course offerings must have a minimum of three students in them. The billing system is notified for each student in each course offering that is not cancelled, so the student can be billed for the course offering.

The main actor of this use case is the Registrar. The Billing System is an actor involved within this use case.

2. Flow of Events

The use case begins when the Registrar selects the "close registration" activity from the Main Form.

2.1 Basic Flow – Successful Close Registration

The system checks to see if a Registration is in progress. If it is, then a message is displayed to the Registrar and the use case terminates. The Close Registration processing cannot be performed if registration is in progress. .

For each open course offering, the system checks if three students have registered and a professor has signed up to teach the course offering. If so, the system closes the course offering and sends a transaction to the billing system for each student enrolled in the course offering.

EXAMPLE 2: CONTINUED

2.2 Alternate Flows

2.2.1 Less Than Three Students in the Course Offering

If in the basic flow less than three students signed up for the course offering, the system will cancel the course offering. The Cancel Course Offering sub-flow is executed at this point.

2.2.2 Cancel Course Offering

The system cancels the course offering. For each student enrolled in the cancelled course offering, system will modify the student's schedule. The first available alternate course selection will be substituted for the cancelled course offering. If no alternates are available, then no substitution will be made. Control returns to the Main flow to process the next course offering for the semester.

Once all schedules have been processed for the current semester, the system will notify all students, by mail, of any changes to their schedule (e.g., cancellation or substitution).

2.2.3 No Professor for the Course Offering

If in the basic flow there is no professor signed up to teach the course offering, the system will cancel the course offering. The Cancel Course Offering sub-flow is executed at this point.

2.2.4 Billing System Unavailable

If the system is unable to communicate with the Billing System, the system will attempt to re-send the request after a specified period. The system will continue to attempt to re-send until the Billing System becomes available

3. Special Requirements

There are no special requirements associated with this use case.

4. Preconditions

4.1 Login

The Registrar must be logged onto the system in order for this use case to begin.

5. Postconditions

There are no postconditions associated with this use case.

6. Extension Points

There are no extension points associated with this use case.

EXAMPLE 3:

Number: **UC2**	Name: **Check Out**	Last Updated: **September 20, 2000**
Release: **1.0**	Primary Actor: **SC User**	Owner:
Description: Upon entering the check out use case the user will begin a series of steps leading to receiving the product. This includes viewing cross sales opportunities (e.g. Would you like to biggie size that?).		
Scenario: 1. Calculate tax. 2. Select *payment option*. 3. Check out. 4. Receive notification. 5. Select distribution channel.		
Assumptions:		
Extensions: 1. View Privacy Policy. 2. View Cross Sales opportunities. 3. Complete Survey. 4. Change quantity.		
Variations: 1. Payment option. a. Purchase Order. b. Credit Card. c. Prepaid account. d. Vignette Employee.		
Notes:		
Priority: High	System: EIP Shopping Cart	Date Created: 9/19/00

RUP ”SUPPLEMENTARY SPECIFICATIONS” TEMPLATE

1. Introduction
1.1 Purpose
1.2 Scope
1.3 Definitions, Acronyms and Abbreviations
1.4 References
1.5 Overview
2. Functionality
3. Usability
4. Reliability
5. Performance
6. Supportability
7. Design Constraints
8. Online User Documentation and Help System Requirements
9. Purchased Components
10. Interfaces
10.1 User Interfaces
10.2 Hardware Interfaces
10.3 Software Interfaces
10.4 Communications Interfaces
11. Licensing Requirements
12. Legal, Copyright and Other Notices
13. Applicable Standards

USE OF PROTOTYPES

Logical user interface description

- What user interface elements are needed? How are they related to each other? How are they used in use case? How do they look like?
- What information the system shows/needs? What information the actor gives/needs? How are business concepts related to user interface and use cases?

User interface prototype

- An executable prototype can be made if
    - There is a good tool for it
    - There is a need to illustrate and test the functioning of the system
- Prototype is just a specification, not the final implementation
- Prototype should be made only if it is really considered necessary

SUMMARY

Use cases are the most important things in requirements analysis

- Descriptions are informal, communicative, but consistent

Domain/business model does not describe the internal data structure of the system

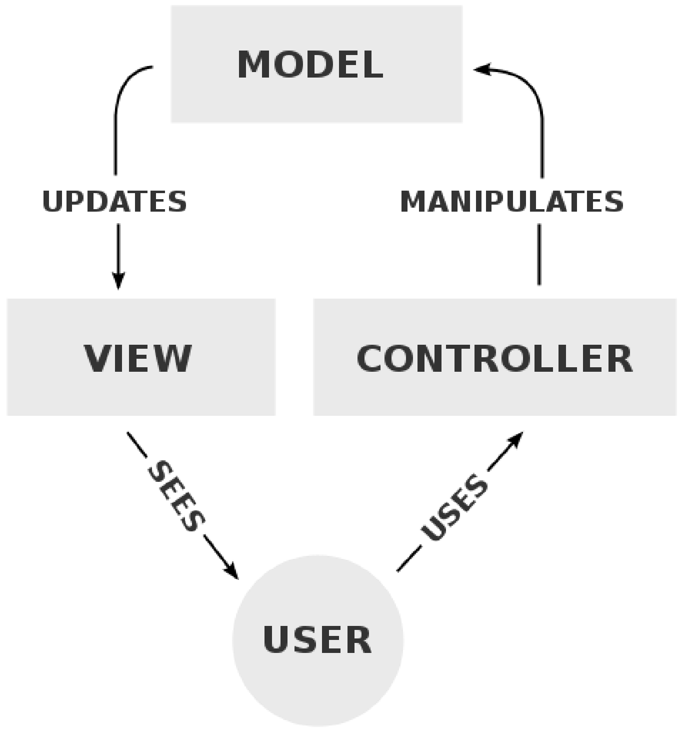
Also the non-functional requirements must be described

User interface description and a possible prototype makes requirements analysis more concrete

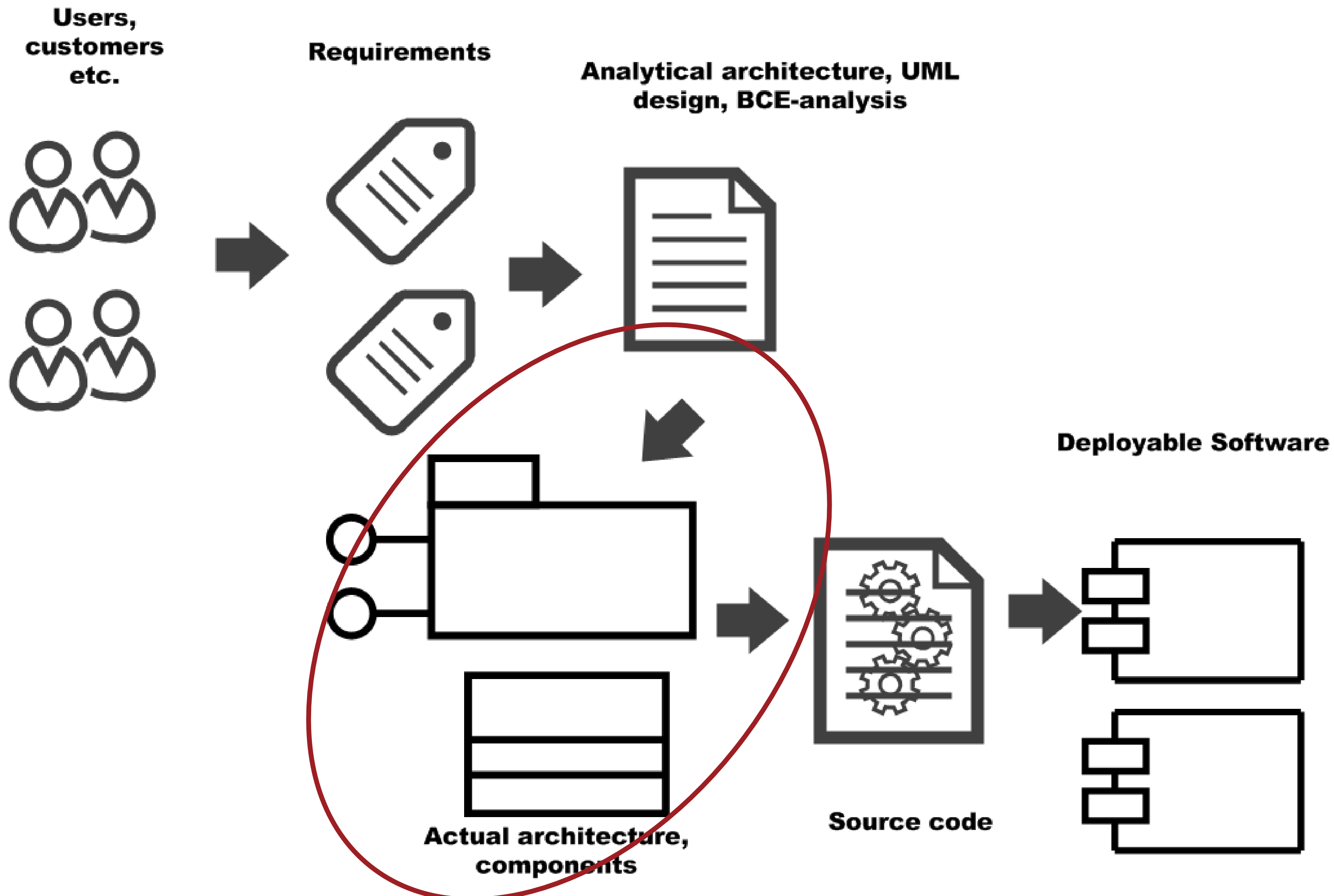
BCE/MVC MODELING

BCE, OR MVC?

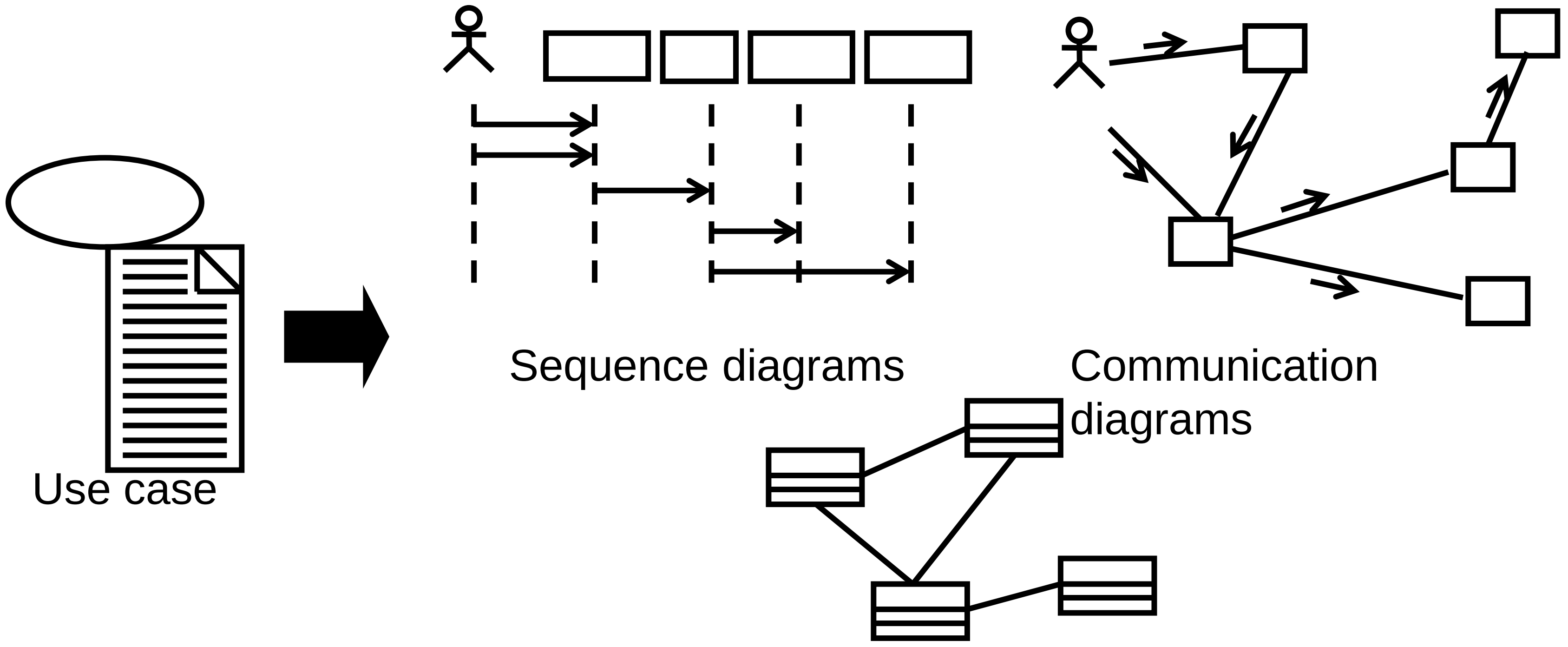
- **BCE (Entity-Boundary-Control) model is a simplification of the MVC (Model-View-Controller) architecture model.**
- **BCE used especially in UML-based approaches, but the concept is directly comparable.**
  - View == Boundary
  - Model == Entity
- **Model is the data structure.**
- **View is the representation of the data.**



FROM REQUIREMENTS TO IMPLEMENTATION


Users, customers etc.
Requirements
Analytical architecture, UML design, BCE-analysis
Deployable Software
Actual architecture, components
Source code

USE CASE REALIZATION



A use case is defined as a set of classes and the interaction between objects of these classes

- Interaction diagrams: sequence diagrams and communication diagrams

ALLOCATING RESPONSIBILITIES TO CLASSES

The starting point is the use case description

- A description of the flow of events that is as detailed as possible and proceeds step by step
- In this phase the description can be completed
    - With the features of the selected technology
    - by adding details
    - By describing in detail the operation of interfaces etc.

The purpose is to search and find the <u>analysis classes</u> that implement the use case

ANALYSIS CLASS

RUP uses a very simple and mechanical way to find classes for a use case

Each use case has

- At least one **boundary class** for each actor in use case
- At least one **control class** that takes care of the flow of events
- As many **entity class** as needed

BOUNDARY CLASSES (VIEW IN MVC)

Only an object of a boundary class can have interaction with external objects (i.e. Actors)

An object of a boundary class delivers messages between actors and other objects

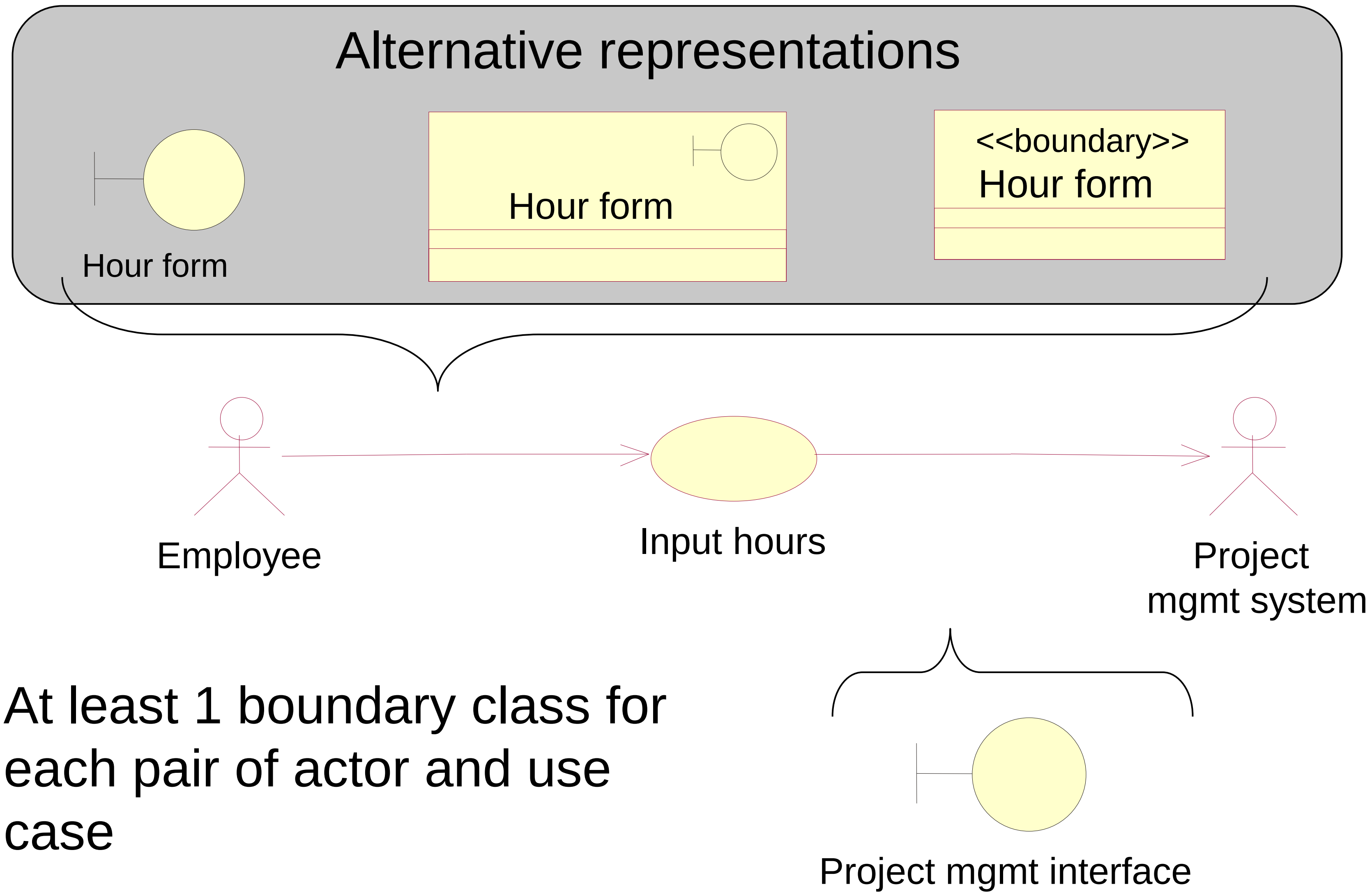
Examples of boundary classes

- User interface classes: forms, windows, …
- System interfaces: data communications, external interfaces
- Hardware interfaces: printers, barcode readers etc.

A boundary object gets signals from users, other systems and hardware and delivers them to internal objects

A boundary object sends messages to users, other systems, and hardware

REPRESENTING A BOUNDARY CLASS



At least 1 boundary class for each pair of actor and use case

CONTROL CLASS (CONTROLLER IN MCV)

Controls and coordinates the flow of events in a use case

- A ”program” that executes the use case

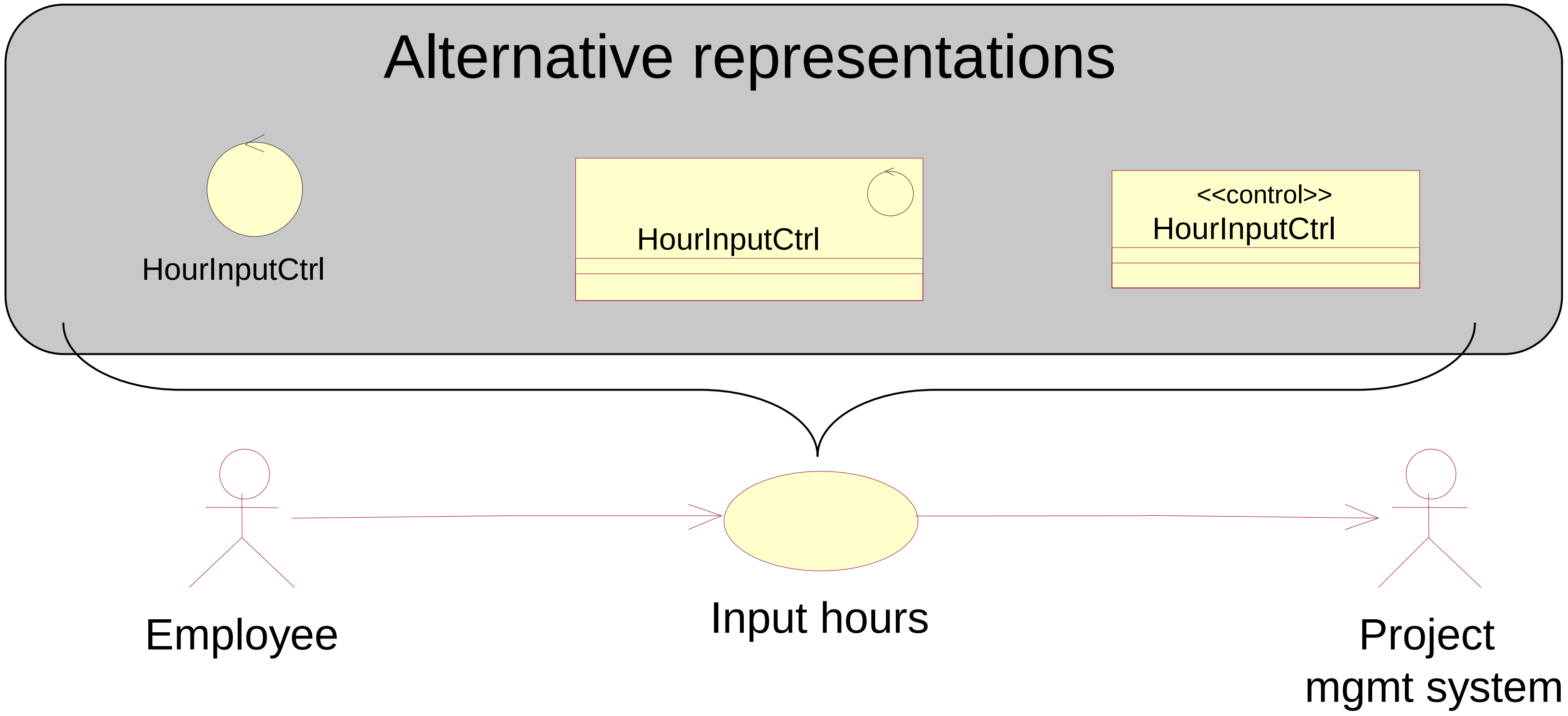
Usually a temporary object that is created only for the use case execution

Not necessary – boundary and entity classes can execute the use case without a control class

Defines the control logic, the order of execution and transaction management

May be involved in implementing distribution

REPRESENTING A CONTROL CLASS



Usually one control class per use case

ENTITY CLASS (MODEL IN MVC)

Models information and behaviour related to it

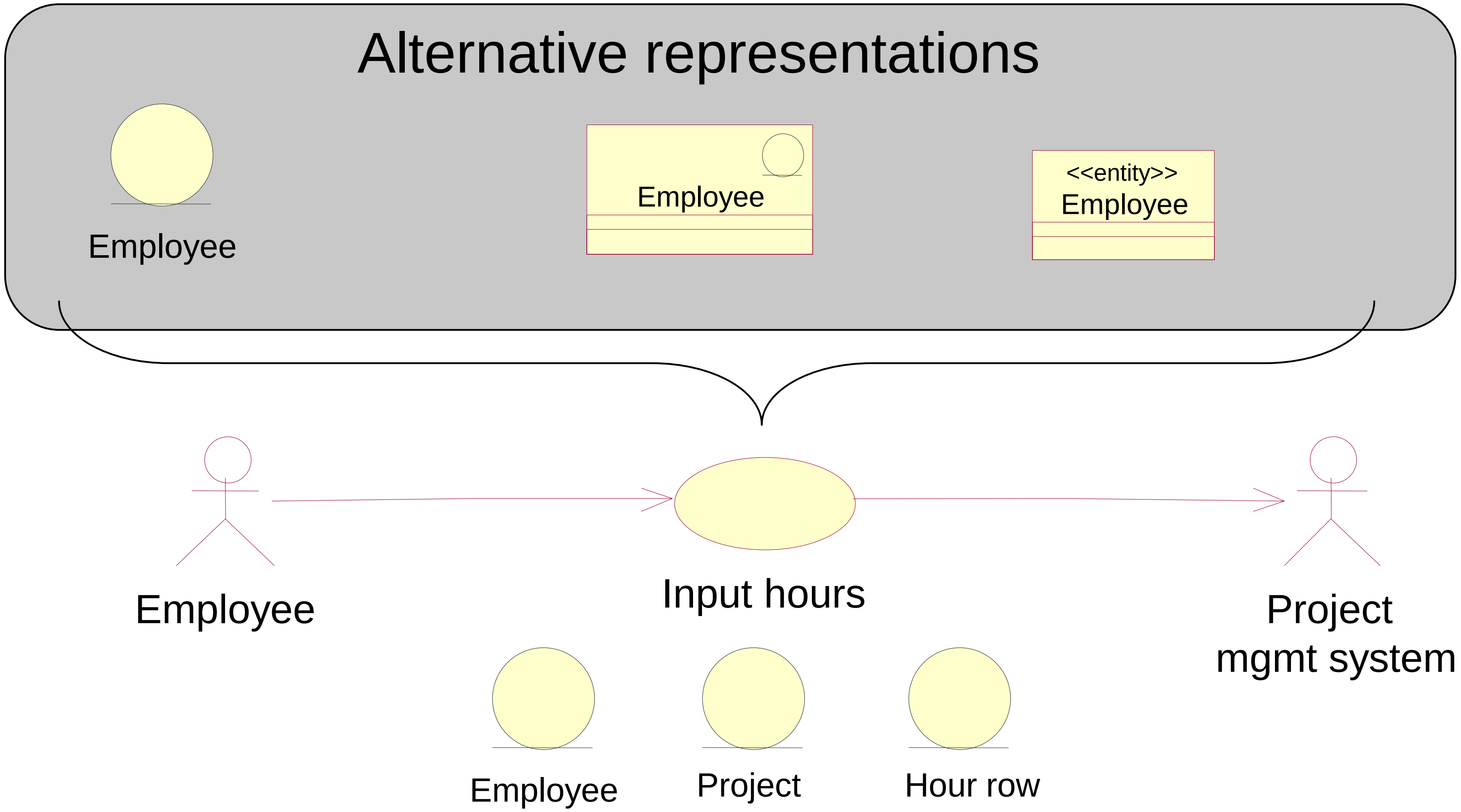
- E.g. Data storage, retrieval and modification

Describes some phenomenon, like an event, person, or real world object

An entity class is independent from its surrounding (user interface, actors, use case)

Many use cases utilize the same entity class

REPRESENTING AN ENTITY CLASS



As many entity classes as needed

DESIGN OF BOUNDARY CLASSES

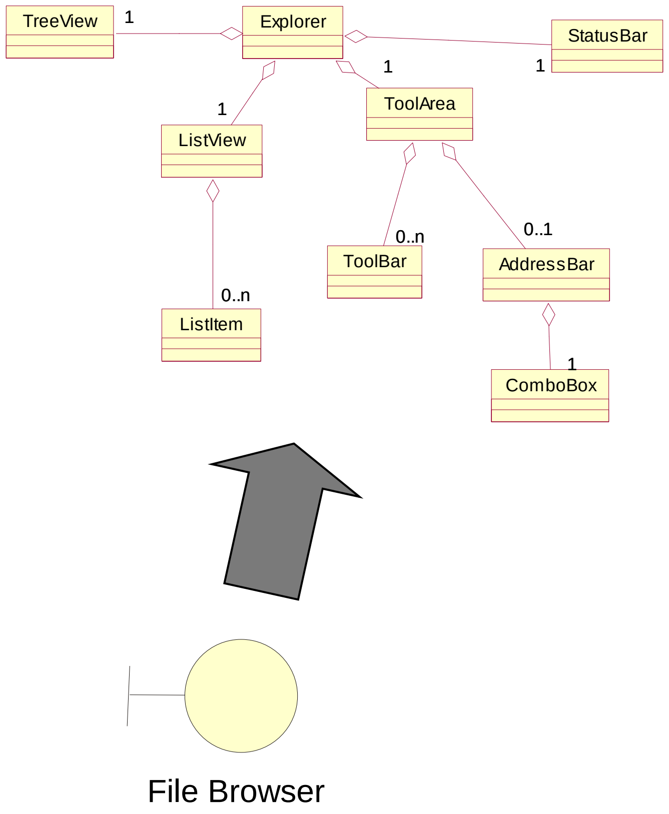
In analysis, boundary classes are usually heavily simplified

- Division to many components

The division is determined by the implementation environment

- How much of the functionality is generated and handled by the environment?
- How much must be implemented manually in separate classes?

This applies both to UI classes and external interface classes



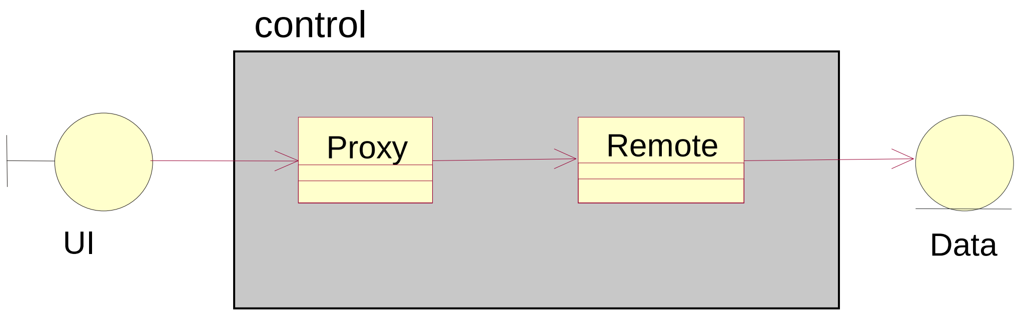
DESIGN OF CONTROL CLASSES

Is the control class needed?

- Are there any own functionality in the class or is it just a message deliverer from a boundary class to entity classes?

Justification for control class implementation

- Complexity of the functionality – move functional logic into the control class
- Distribution and its management
- Transaction management
- Isolation of changes in the control class

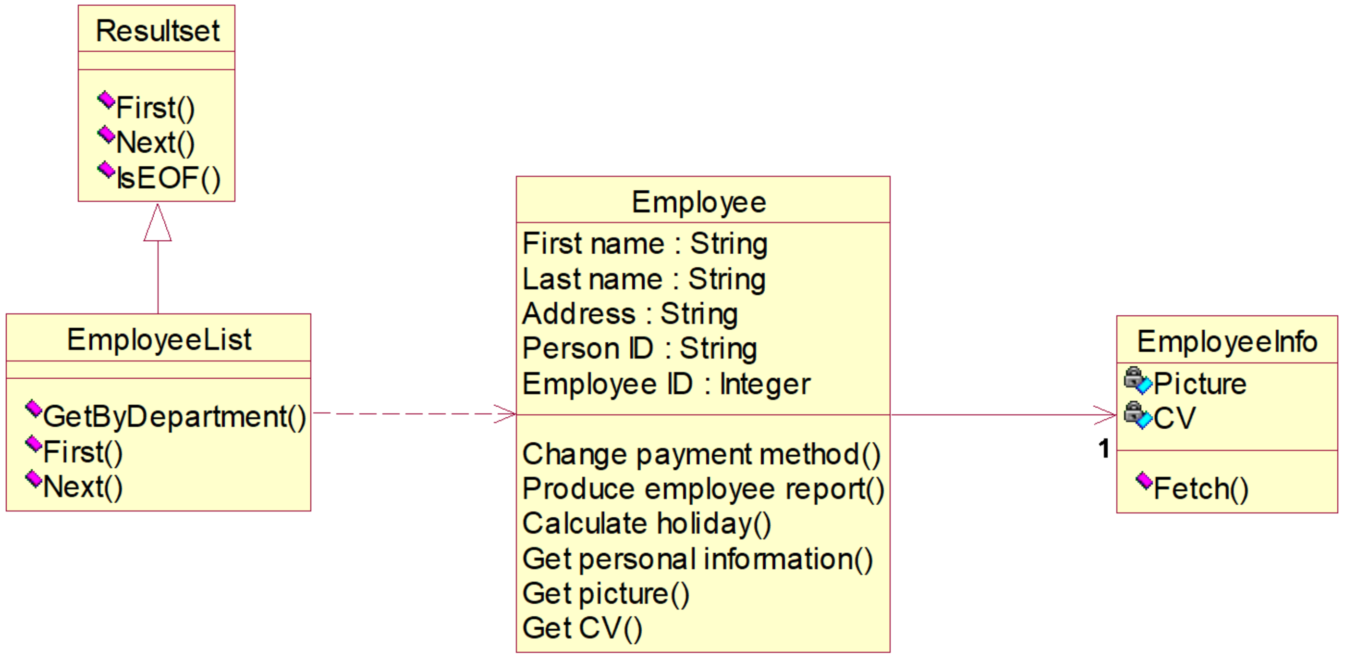


DESIGN OF ENTITY CLASSES

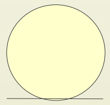
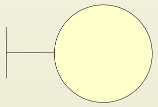
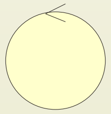
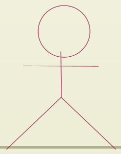
Entity classes are designed according to the database middleware

Performance may need optimization

One entity class may create many kinds of lists and collections



BCE MODEL CONNECTIONS

Can connect to	Entity 	Boundary 	Control 	Actor 
Entity			X	
Boundary			X	X
Control	X	X	X	
Actor		X		