Requirements Engineering by Use Case Analysis

UML based modeling and analysis

Dániel Varró

Requirements Analysis

Requirements analysis

- **Requirements engineering (RE)** is the process of *identifying*, *organizing*, and *documenting* the continuously changing requirements of a project
- **Requirement**: a condition or capability to which the system must conform
- An *early identification of requirements* is critical for the quality of the system under design
  - consistent?, complete? unambiguous?
- Gathering of requirements is a very complex engineering task
  - „Requirements do not come from the air”
  - an iterative refinement process with regular control

Problems of Requirements Analysis (Surveys)

- Failure of SW projects:
  - 1/3 never completed
  - an additional 1/2 completed with only partial success
- Causes of failure:
  - Problems with requirements specification >50%
    - 13%: lack of interaction with users
    - 12%: incomplete requirements
    - 11%: changing requirements
    - 11%: irreal or unclear requirements

Definition of requirements analysis

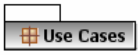
Identification of

- Goals: the objectives of the system
  - Why do we need the SW?
- Services („operationalization”)
  - What functionality do we need to design?
- Constraints
  - Restrictions of the design process (e.g. cost, deadlines)
- Responsibilities to each requirement (SW vs. human)

Categorization of RE

- **High-level (System-level) requirements**
  - Feature (FEAT): high-level product requirement from the customer’s point of view
  - Stakeholder needs (NEED):
  - The agreement between the customer and the system analyst documented in the vision document
- **Low-level (Software-level) requirements**
  - Software requirements
  - **Actor**: someone or something outside the system that interacts with the system
  - **Use case** (UC): a functional requirement
  - **Supplementary requirement** (SUPL): a non-functional requirement

Main documents of RE



- **Use Case model**
  - Actors, Use Cases, Subsystems
  - Scenarios as workflow

- Architectural description: Detailed textual description of
  - Use cases
  - Scenarios

- Glossary (Szójegyzék)
  - Precise definition of common terms

- GUI prototype
  - Communication with end users

Requirements of a Table Game Championship Manager System

Verbal Requirements

- Design a system for organizing championships of table games (chess, go, backgammon, etc.)
- Requirements:
  - A player should register and log in to the system before using it.
  - Each registered player may announce a championship.
  - Each player is allowed to organize a single championship at a time.
  - Players may join (enter) a championship on a web page
  - When the sufficient number of participants are present, the championship can be started by the organizer.
  - After starting a championship, the system must automatically create the pairings in a round-robin system.

Passive sentences should be avoided!

Verbal Requirements

- Design a system for organizing championships of table games (chess, go, backgammon, etc.)
- Requirements:
  - A player should register and log in to the system before using it.
  - Each registered player may announce a championship.
  - Each player is allowed to organize a single championship at a time.
  - Players may join (enter) a championship on a web page
  - When the sufficient number of participants are present, the organizer starts the championship.
  - After starting a championship, the system must automatically create the pairings in a round-robin system.

Verbal Requirements (cont.)

- Requirements (cont.):
  - If the championship is not started yet (e.g. the number of participants does not reach a minimum level), the organizer may cancel the championship
  - The actual game is played between existing clients, which is outside the scope of the modelled system.
  - Both players should report the result and the moves after each game using a web form. A win scores 1 point, a draw ½, and a loss 0.
  - If players report contradicting results, the organizer should judge who is the winner. The organizers penalizes the cheating player by a 1 point penalty.
  - When all games are finished, the organizer should close the championship by announcing the winner. Then he or she may start organizing a new championship.

Missing Requirements

- A game should be finished within a given deadline (time limit).
- If none of the two players have reported the result within this deadline, then both players are considered to be losers.
- If only one player has reported the result, then his (or her) version is considered to be the official result.

- NOTE: New requirements will emerge during UC analysis (especially when detailing UCs). An iterative requirements engineering process is highly recommended.

Best Practice: Requirements

- A requirement should contain
  - a short description
  - a stand-alone sentence / paragraph
- English:
  - Avoid passive sentences
  - Use the following auxiliaries:
    - Positive: shall/must, should, may,
    - Negative: must not, may not
- Detail them with parameters:
  - Priority, Status, Difficulty, Responsibility, Risk

Elements of Use Case Diagrams by Example

Definition of Use Cases

- **Use cases (használati eset)** capture the functional requirements of a system
- UCs describe
  - the typical interactions
  - between the users of a system and
  - the system itself,
  - by providing a narrative of how a system is used
- A set of scenarios tied together by a common user goal
- Verb + Noun (Unique)!

M. Fowler: UML Distilled. 3rd Edition. Addison-Wesley

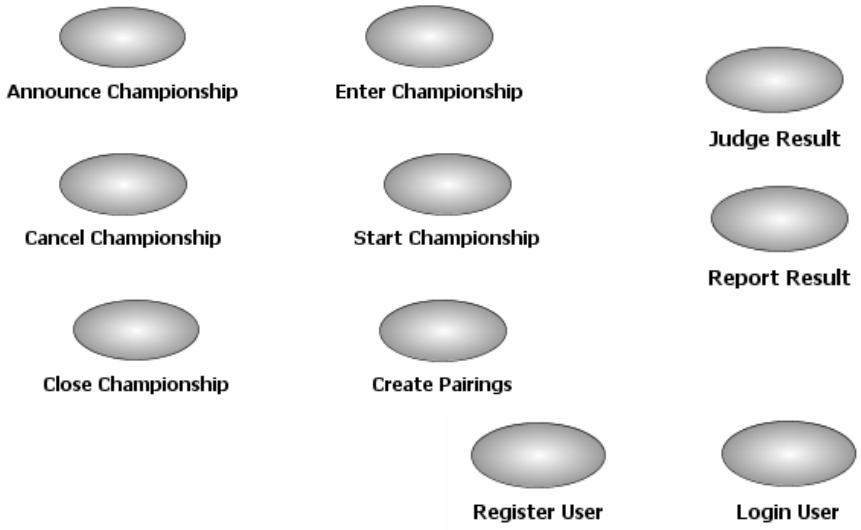
From Verbal Requirements to Use Cases

- Requirements:
  - Each registered player may announce a championship.
  - A player should register and log in to the system before using it.
  - Each player is allowed to organize a single championship at a time.
  - Players may join (enter) a championship on a web page
  - When the sufficient number of participants are present, the organizer starts the championship.
  - After starting a championship, the system must automatically create the pairings in a round-robin system.

Verbal Requirements (cont.)

- Requirements (cont.):
  - If the championship is not started yet (e.g. the number of participants does not reach a minimum level), the organizer may cancel the championship
  - The actual game is played between existing clients, which is outside the scope of the system system.
  - Both players should report the result and the moves after each game using a web form. A win scores 1 point, a draw ½, and a loss 0.
  - If players report contradicting results, the organizer should judge who is the winner. The organizers penalizes the cheating player by a 1 point penalty.
  - When all games are finished, the organizer should close the championship by announcing the winner. Then he or she may start organizing a new championship.

(Initial) Collection of Use Cases



Definition of Actors

- **Actor (aktor)** is a **role** that a user plays with respect to the system.
  - Primary actor: calls the system to deliver a service
  - Secondary actor: the system communicates with them while carrying out the service
- Relationship of UCs and Actors
  - A single actor may perform many use cases;
  - A use case may have several actors performing it.
- One person may act as more than one actor,
  - Example: A person may organize a championship and may participate in another
- An actor is outside the boundary of the system

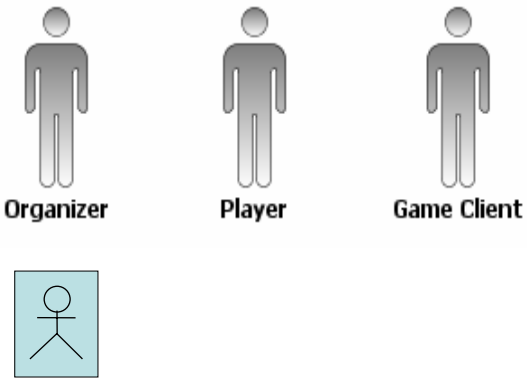
From Verbal Requirements to Use Cases

- Requirements:
  - Each registered player may announce a championship.
  - A player should register and log in to the system before using it.
  - Each player is allowed to organize a single championship at a time.
  - Players may join (enter) a championship on a web page
  - When the sufficient number of participants are present, the organizer starts the championship.
  - After starting a championship, the system must automatically create the pairings in a round-robin system.

Verbal Requirements (cont.)

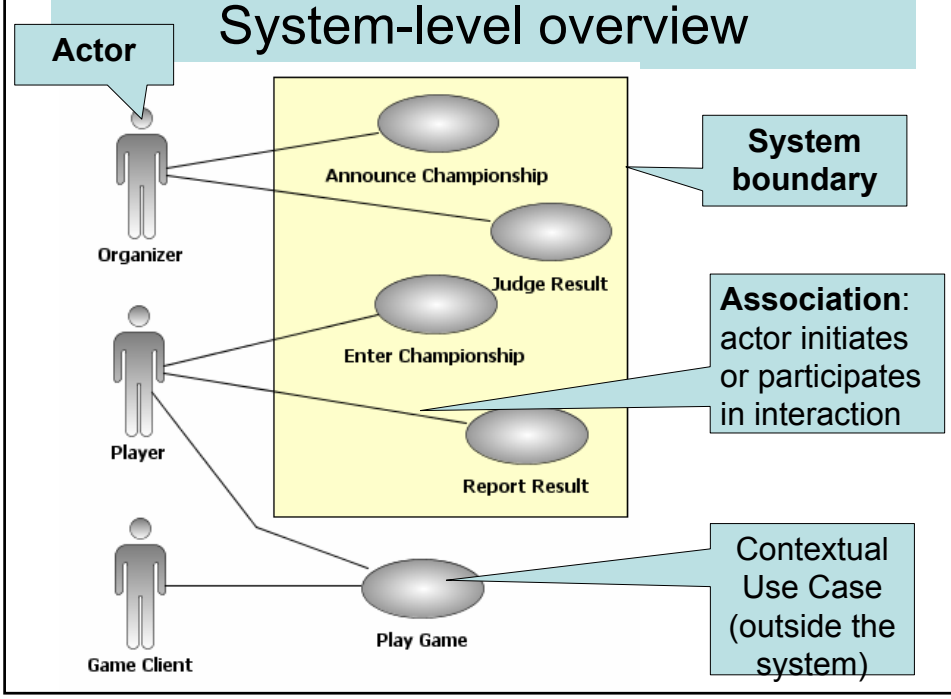
- Requirements (cont.):
  - If the championship is not started yet (e.g. the number of participants does not reach a minimum level), the organizer may cancel the championship
  - The actual game is played between existing clients, which is outside the scope of the system system.
  - Both players should report the result and the moves after each game using a web form. A win scores 1 point, a draw ½, and a loss 0.
  - If players report contradicting results, the organizer should judge who is the winner. The organizers penalizes the cheating player by a 1 point penalty.
  - When all games are finished, the organizer should close the championship by announcing the winner. Then he or she may start organizing a new championship.

(Initial) Collection of Actors

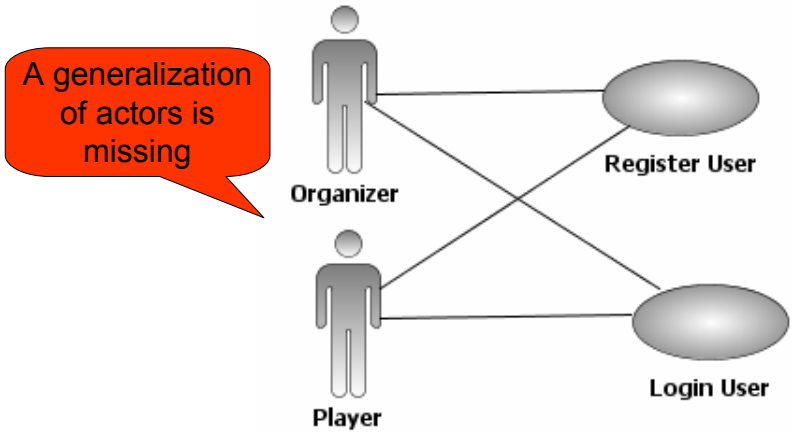


Relations between UCs and Actors

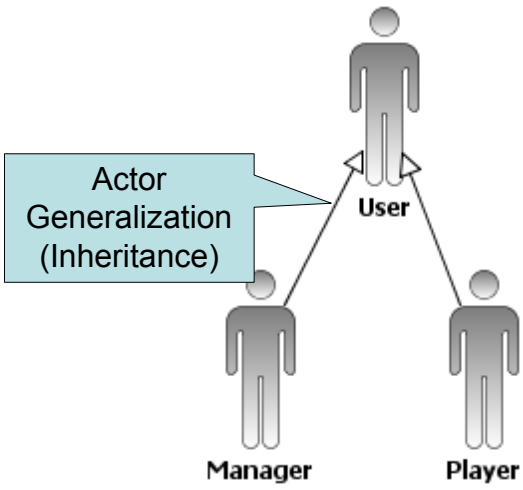
System-level overview



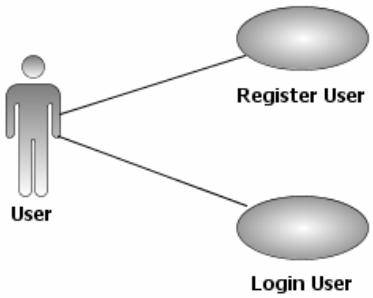
Anti-pattern: UC diagrams



Overview of Actors

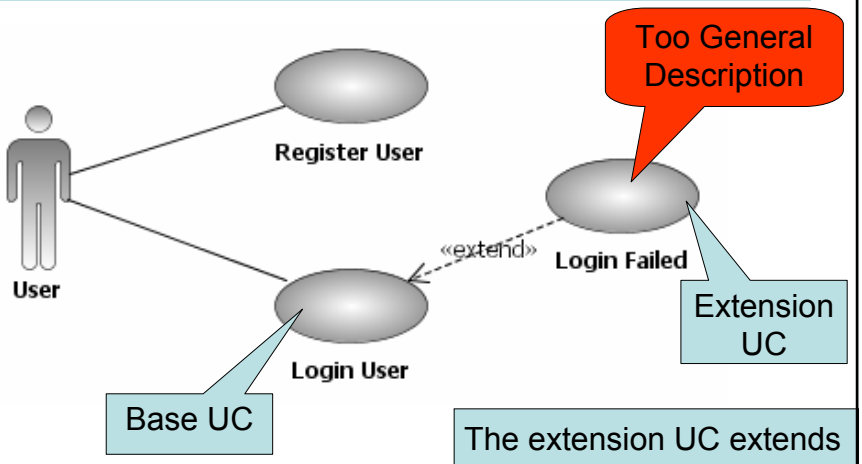


User Management



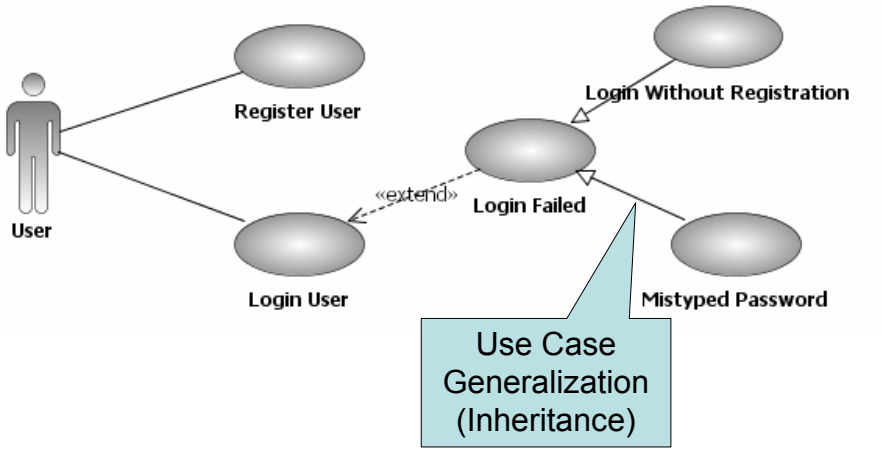
What happens if
- the user's password is incorrect?
- a user is not registered, but attempts to login?

Extend relationship



The extension UC extends core functionality by handling unusual (exceptional) situation

Refinement of Use Cases

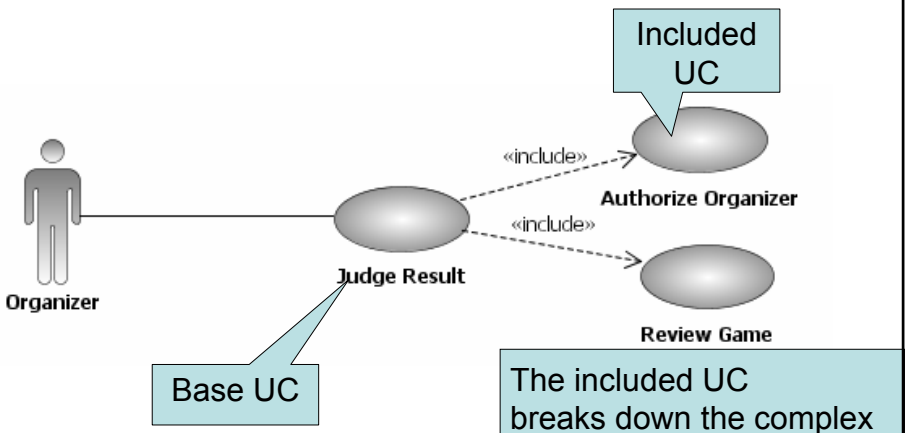


How to handle complex functionality?

Judge result =

- Check if the organizer is the judge
- Analyze the game
- Decide on result
- Report the result

Include relationship



Summary: UC Relations

- Association (Asszociáció)
  - actor – use case
  - the actor initiates (or participates) the use of the system
- Extend (Bővítés)
  - use case – use case
  - a UC may be extended by another UC (typically solutions for exceptional situations)

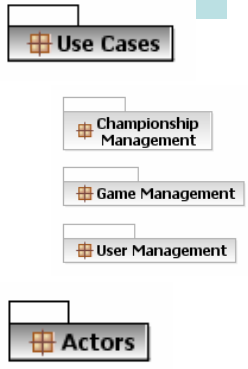
Summary: UC Relations

- Generalization (Általánosítás)
  - actor - actor
  - use case – use case
  - a UC or actor is more general / specific than another UC or actor
- Include (Beszúrás)
  - use case – use case
  - a complex step is divided into elementary steps
  - a functionality is used in multiple UCs

Best practices of UC analysis

Best practices: Grouping

- Grouping UCs
  - Identify functional building blocks
  - Group them into packages
  - NOTE: related by functionality, NOT by role
- Grouping actors:
  - Keep actors in a package within the subsystem they exclusively belong to
  - Global actors: in top-most package



Best practices: Naming and arrangement

- Actors
  - Name actors according to their roles and avoid using job titles
  - Divide complex roles into multiple actors
  - Start the diagram by placing the most important actor in the top left corner
- Use Cases
  - Use domain specific verbs for UCs
  - Avoid technical descriptions – UCs are frequently for non-technical reader
- Relationships
  - Avoid crossing or curved lines when drawing relations
  - Use <<extend>> and <<include>> relations „lightly”
  - Place them into the appropriate functional block

Main guideline: UC diagrams should be SIMPLE

What UC diagrams to create?

- **Actors' inheritance tree**: usually once
- **System-level overview**: once in a system
- **List of UCs**: once in a functional block (subsystem) with many UCs
- **„Regular" UC diagrams**: as many as necessary to have simple UC diagrams

Detailing Use Cases

Parameters of Use Case

- Responsibility: (Contact name)
- Priority:
  - Must, Should, Could
- Status:
  - Proposed, Approved, Incorporated, Validated
- (Technical) Difficulty:
  - Low, Medium, High
- Risk:
  - Schedule: Low, Medium, High
  - Technology: Low, Medium, High
- Iterations:
  - Planned
  - Actual
- Stability
  - Low, Medium, High

Detailing UCs in your homework

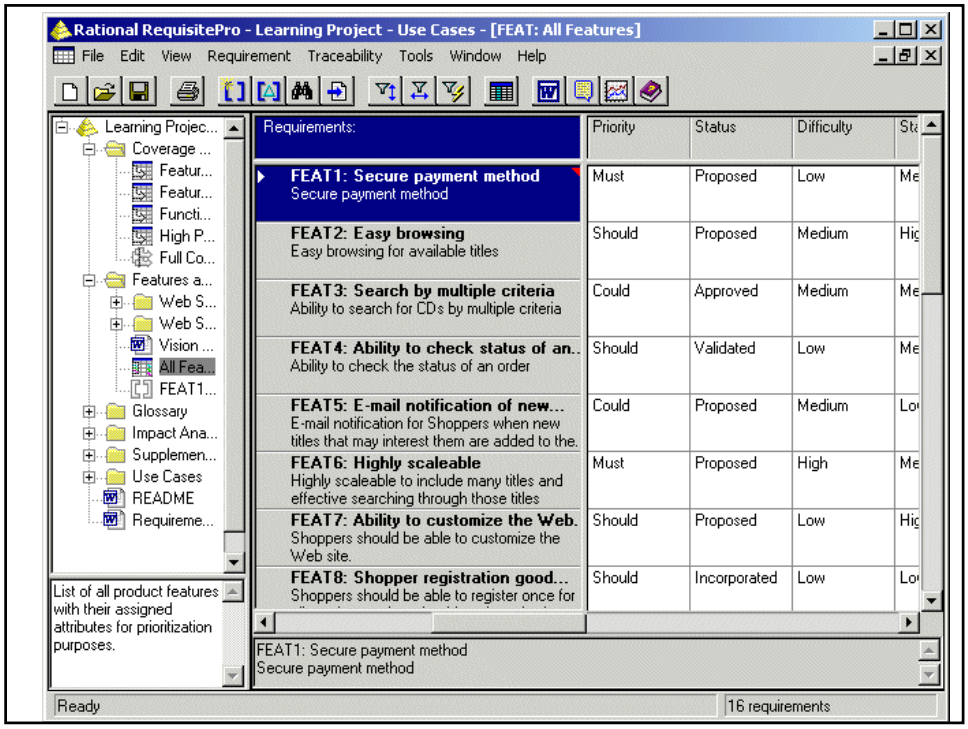
Attach a note to each UC containing at least
- Contact name
- Priority

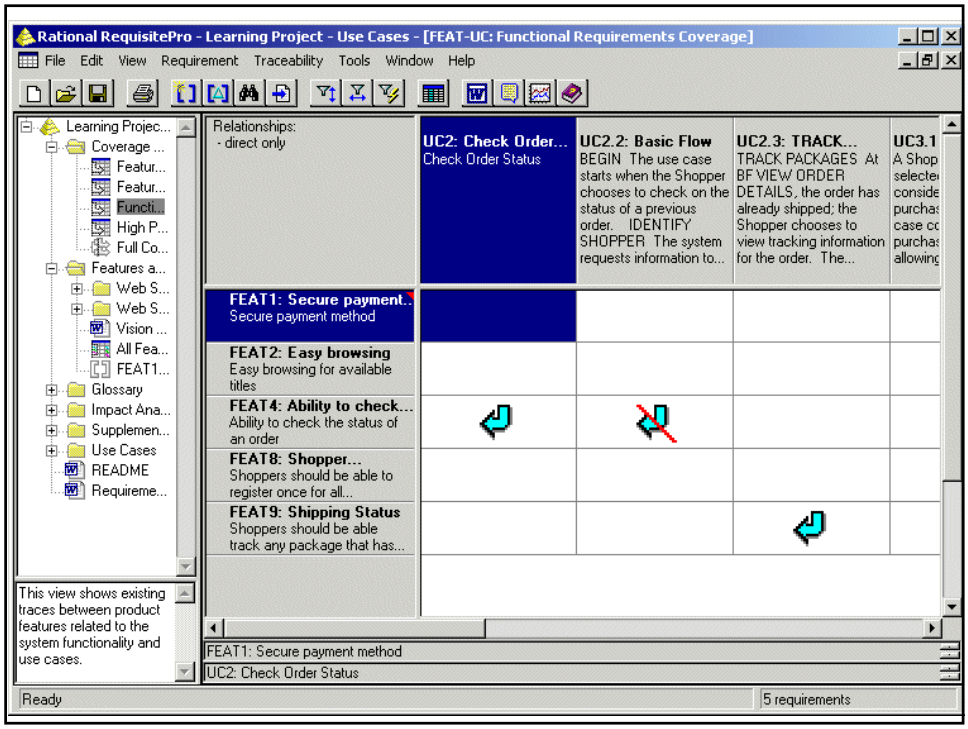
- Scenarios (Next lecture)
  - Workflow model
  - Textual description

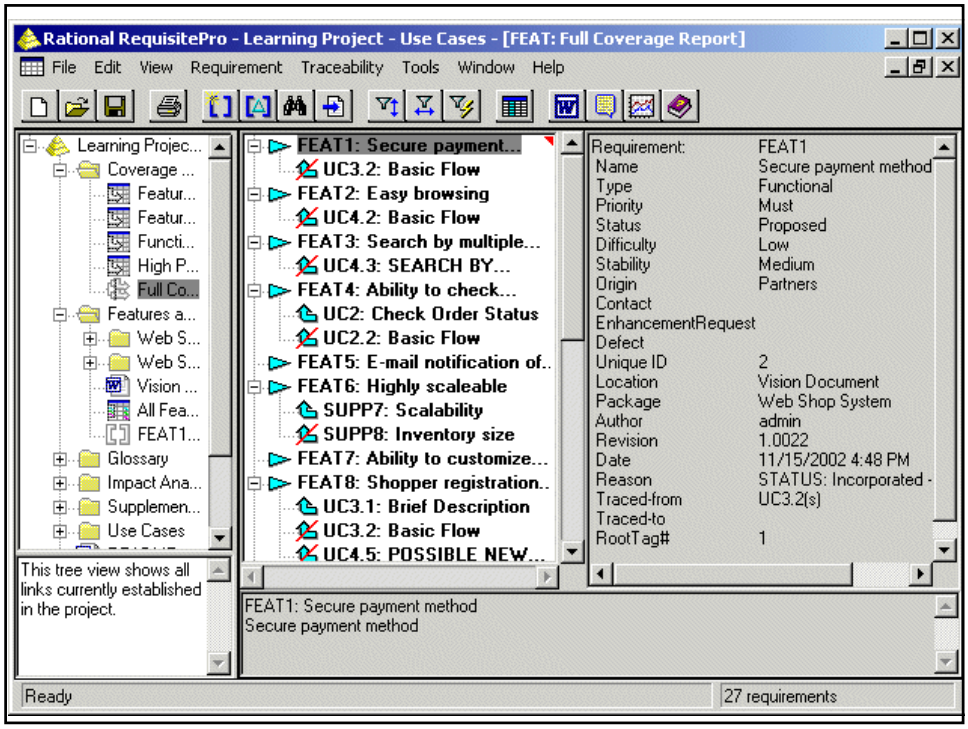
Organization of Requirements in Rational RequisitePro

RequisitePro

- An integrated tool of IBM Rational for managing
  - High-level requirements
  - Use cases
- Goals:
  - Facilitates communication and team work
  - Decreases project risks
- Tools:
  - Word documents
  - Requirements Database
  - Integrated into IDEs


Rational RequisitePro - Learning Project - Use Cases - [FEAT: All Features]
File Edit View Requirement Traceability Tools Window Help
Learning Projec...
Coverage ...
Featur...
Featur...
Functi...
High P...
Full Co...
Features a...
Web S...
Web S...
Vision ...
All Fea...
FEAT1...
Glossary
Impact Ana...
Supplemen...
Use Cases
README
Requireme...
List of all product features with their assigned attributes for prioritization purposes.
Requirements:
Priority
Status
Difficulty
FEAT1: Secure payment method
Secure payment method
Must
Proposed
Low
FEAT2: Easy browsing
Easy browsing for available titles
Should
Proposed
Medium
FEAT3: Search by multiple criteria
Ability to search for CDs by multiple criteria
Could
Approved
Medium
FEAT4: Ability to check status of an...
Ability to check the status of an order
Should
Validated
Low
FEAT5: E-mail notification of new...
E-mail notification for Shoppers when new titles that may interest them are added to the...
Could
Proposed
Medium
FEAT6: Highly scaleable
Highly scaleable to include many titles and effective searching through those titles
Must
Proposed
High
FEAT7: Ability to customize the Web...
Shoppers should be able to customize the Web site.
Should
Proposed
Low
FEAT8: Shopper registration good...
Shoppers should be able to register once for
Should
Incorporated
Low
FEAT1: Secure payment method
Secure payment method
Ready
16 requirements


Rational RequisitePro - Learning Project - Use Cases - [FEAT-UC: Functional Requirements Coverage]
File Edit View Requirement Traceability Tools Window Help
Learning Projec...
Coverage ...
Featur...
Featur...
Functi...
High P...
Full Co...
Features a...
Web S...
Web S...
Vision ...
All Fea...
FEAT1...
Glossary
Impact Ana...
Supplemen...
Use Cases
README
Requireme...
This view shows existing traces between product features related to the system functionality and use cases.
Relationships:
- direct only
UC2: Check Order...
Check Order Status
UC2.2: Basic Flow
BEGIN The use case starts when the Shopper chooses to check on the status of a previous order. IDENTIFY SHOPPER The system requests information to...
UC2.3: TRACK...
TRACK PACKAGES At BF VIEW ORDER DETAILS, the order has already shipped; the Shopper chooses to view tracking information for the order. The...
UC3.1
A Shop selecte consid purcha case c purcha allowin
FEAT1: Secure payment...
Secure payment method
FEAT2: Easy browsing
Easy browsing for available titles
FEAT4: Ability to check...
Ability to check the status of an order
FEAT8: Shopper...
Shoppers should be able to register once for all...
FEAT9: Shipping Status
Shoppers should be able track any package that has...
FEAT1: Secure payment method
UC2: Check Order Status
Ready
5 requirements

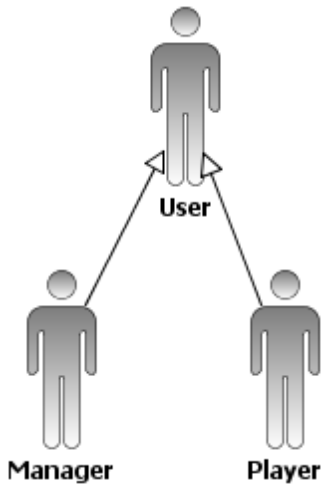


Next Lecture: Detailing Use Cases

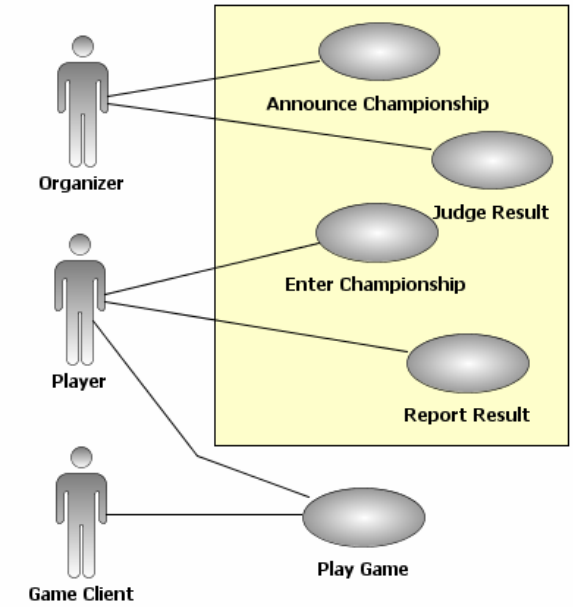
- How to textually capture scenarios?
- How to capture scenarios using UML Activity diagrams?

Milestone: UC Diagrams

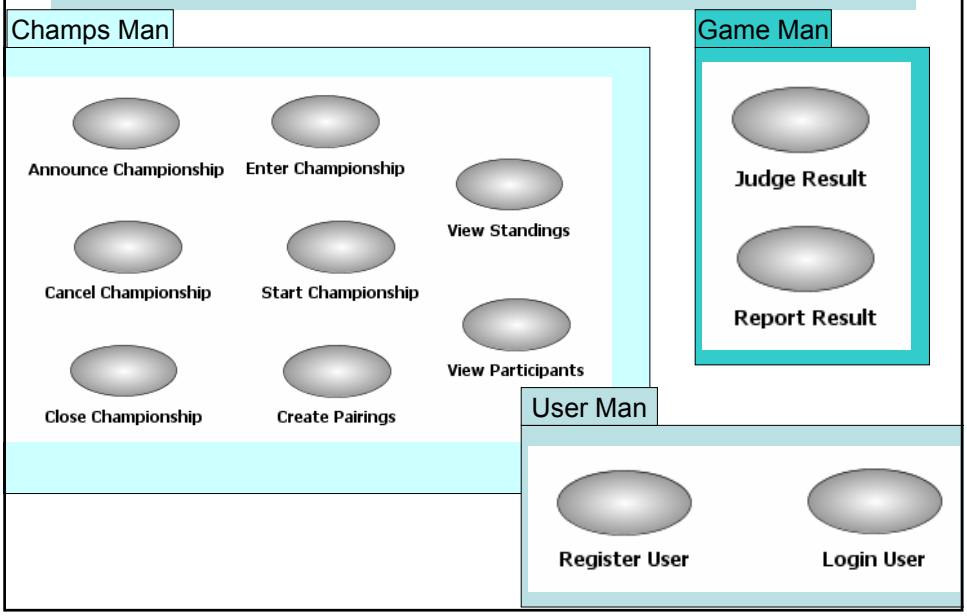
Actors' Inheritance Tree



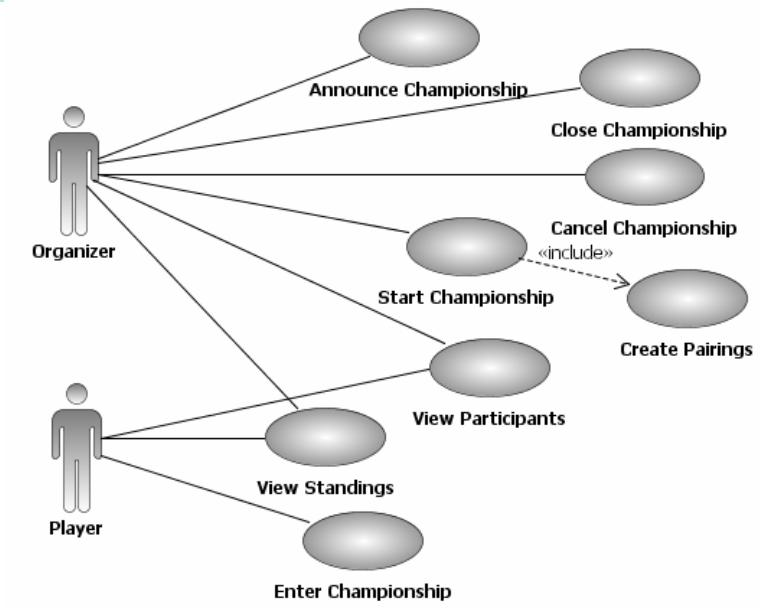
System-level overview



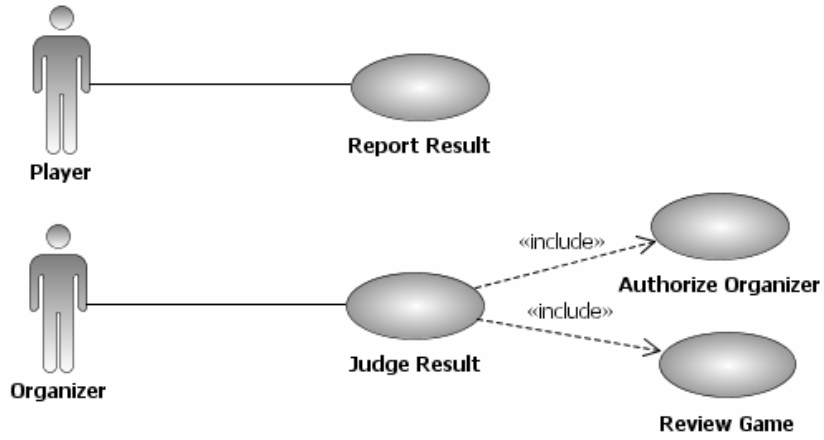
Collection of Use Cases (Incomplete!)



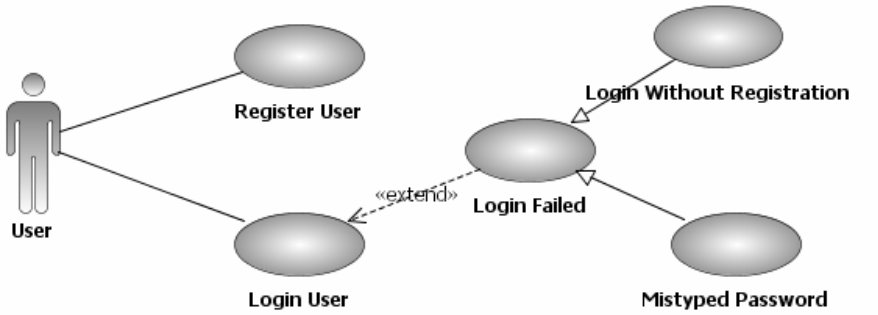
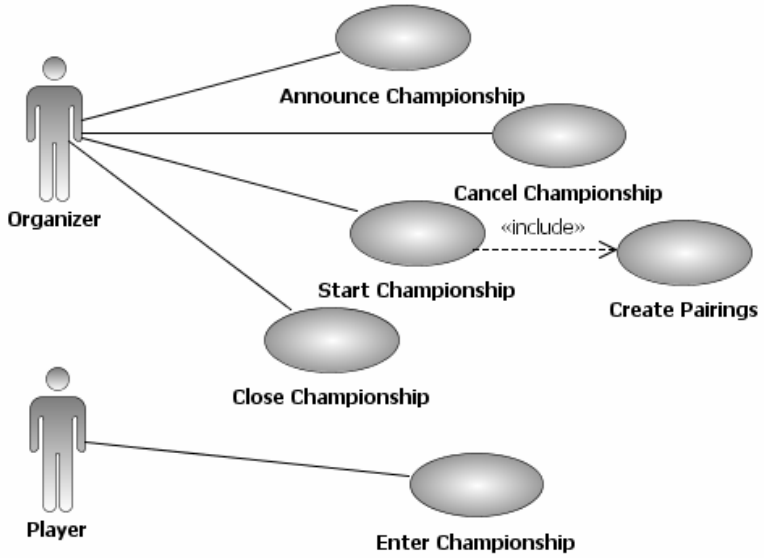
Championship Management



Game Management



User Management



Championship Management