Lecture 1: Processes, Requirements, and Use Cases
Development Processes

• Early Days: evolve a system
  – Build and fix
  – Leads to chaos
  – Need for intelligent design

• Waterfall Model
  – Requirements, Design, Code, Tests, Maintenance
Rational Unified Process

• Iterative, incremental development:
  – *Development cycle*: results in a release
  – *Iteration phases*: Inception, Elaboration, Construction, Transition
  – *Phase activities*: Business Modeling, Requirements, Design, Implementation, Test, Deployment, Configuration & Change, Project Management, Environment (tools, process design)
Factors for Success and the RUP

- Iterative development: develop part of the functionality, explore designs such as GUI
- Daily builds: incorporate new code into (partially) complete system; rapid feedback via testing
- Team experience in shipping multiple products
- Early focus on building and evaluating a cohesive system architecture
Lightweight vs Heavyweight Process

• Formal, heavyweight: strict phases with defined formal documentation, often not iterative
• Agile: adjust to application, activities in one phase differently weighted
• Extreme programming: smaller projects, all staff eye to eye, paired programming, tests as specifications -> little formal documentation other than code
Introduction to Requirements

• Basic but not sole step in inception
• Types of requirements: FURPS
  – Functionality
  – Usability
  – Reliability
  – Performance
  – Supportability
Some OO Terminology

• Responsibility – function that must be performed. E.g. hiring personal, searching a data base for a record
• Role – abstract entity that performs the role
  – E.g. project leader, DB class
• Actor – concrete entity that plays a role to fulfill some responsibility. E.g. Fred Bloggs, DB class instance
Actors and RUP

• External object that interacts with system
  - external = assumed, already given
• Causes input events, receives output
• Examples from POS: cashier, customer, product info database (assumed external)
Use Cases

• Story of system usage, basic RUP requirements method; Functionality aspect of requirements
• Scenario: Specific use of a system, interaction between actors and system
• Use case: collection of related scenarios
• Use case documentation
  – Individual use cases: tables, paragraphs
  – Actors / Use case interactions: use case diagrams
## Use Case Table - Example

<table>
<thead>
<tr>
<th>Dater</th>
<th>System responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Log in</td>
<td>2. Determine valid</td>
</tr>
<tr>
<td></td>
<td>3. Display dater menu</td>
</tr>
<tr>
<td>4. Choose get-date</td>
<td>5. Display criteria input form</td>
</tr>
<tr>
<td>6. Enter prefs</td>
<td>7. Find date</td>
</tr>
<tr>
<td>7. Logs out</td>
<td>8. Display date data</td>
</tr>
</tbody>
</table>
Paragraph Style - Sections

- Primary Actor(s)
- Stakeholders and interests
- Preconditions
- Postconditions
- Happy Case
- Alternative Cases
- Special Requirements
- Variations (technology, data)
- Frequency
- Open Issues
Stakeholders

- Developers, users, marketing, customers, regulators
- Use case describes part of contract between stakeholders and developers
- E.g. Dating system
  - daters, administrator, developers
Pre-conditions and Post-conditions

• Preconditions
  – What we will assume to be true before a system use

• Postconditions
  – What we guarantee to be true afterwards

• Contracts: pre and post conditions
Happy Cases

• Main Flow: normal or main flow of control in a use case

• Alternatives: minor success flows, error flows, exceptions
  – Branches or subcases of main flow
  – Refinements of abstract functionality
Finding Use Cases

• Not single steps, not whole complex processes.
  – E.g. Dating system: not log on, not DS usage, but Dater Asks for a Date

• Elementary Business Process

• Abstract from concrete to intentional
  – E.g.
    • concrete: enter card, enter PIN
    • intentional: user identifies himself

• Use cases satisfy user/actor goals
Finding Actors/Roles

• Types of Actors
  – Primary, supporting, and offstage actors

• MetaRoles: system initiation and termination, updates

• Events and who causes them
Use Case Diagrams

- Show actors/roles and use cases they interact with
- Shows relationships between use cases
  - Alternative subcases
    - e.g. administrator updates DS database
      - Delete a member, Add a member
  - Uses subcase
    - e.g. member/administrator uses system
      - Log on sub use case
      - Data-base component
Sample Use Case Diagram
Tips

• Do not get caught up constructing Use Case Diagrams
• Use *intentional* use cases during requirements, *concrete* during design
• Avoid tables if they are too restrictive
• Accept that inception use cases will be incomplete or some will lack detail
System Increments

• System increments
  × Horizontal: one layer or tier at at time
  √ Vertical: one or more “threads” or complete functional uses

• Use Cases and system increments
  – Choose a subset of the uses cases
Use Cases and System Increments

• Initial and subsequent increments
• Selection Factors
  – Risk: of remaining, are some complex, ambiguous, uncertain usability
  – Coverage: want to touch on all major functionality in early iterations, principal data flows, happy cases
  – Criticality: critical to business enterprise
  – Architectural: start up, stripped down vertical functionality
DS Use Case Groupings

• Use cases by class of user
  – Member
  – Administrator
  – Unauthorized user

• Use case by member user functionality
  – GetADate
  – SetMemberData
DS First Increment Choice

- Use Case: member logs on and asks to get a date, system returns a date from DB or reports no date present
- Rationale
  - Criticality
  - Coverage
- Missing supporting functionality
  - Administrator enters members
  - Members enter their properties
  - Will need to preload the DB with data