Object-Oriented Requirements

Topics

- What are “object-oriented” requirements?
- Unified Modeling Language (UML)
- Modeling the problem domain
  - UML Class diagrams
- Scenarios
  - UML Use Case diagrams

Object-oriented requirements

- Requirements are not usually object-oriented.
- Object-oriented notations are useful for defining and specifying requirements.
- Unified modeling language (UML) is one such notation.

Unified Modeling Language

- Unifies the notations of the three most popular OO methodologies:
  - OOA/OOD (Grady Booch)
  - OMT (Jim Rumbaugh)
  - OOSE (Ivar Jacobson)
- Adopted by OMG
- Is a (mainly graphical) notation, not a method.

Class diagrams

- Similar to entity-relationship diagrams, but more expressive in most regards.
- Used for modeling the problem domain during the requirements phase.
- Used to model the logical structure of the application domain during the analysis phase.

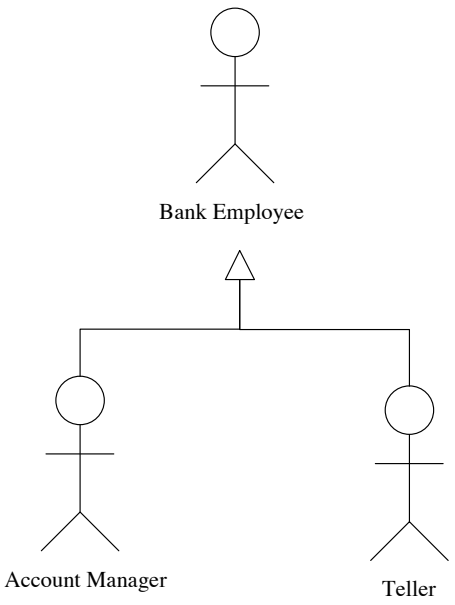
Use case diagrams

- Show actors and use cases (scenarios), and the relationships among them.
- Use cases represent the functionality of a system.
- Actors represent the users of a system (human or otherwise).

Actors

- Users are grouped into classes called *actors*.
- There may be many users instantiating a single actor.
- There may be a single user who instantiates more than one actor.
- An actor need not be human, i.e.could be another system.
- Actor classes may be grouped into an inheritance hierarchy.

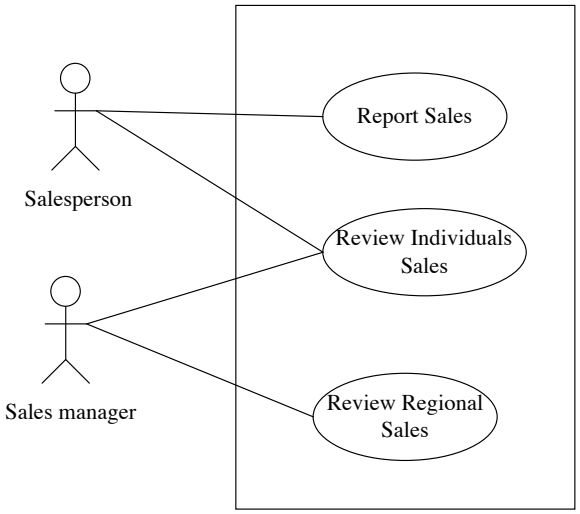
Actors (2)



Use cases

- A use case constitutes a complete course of events initiated by an actor.
- A use case provides something of value to the actor.
- It specifies the interaction that takes place between an actor and the system.
- Like actors, use cases are classes that can be instantiated each time a user performs a use case on the system.

Use cases (2)



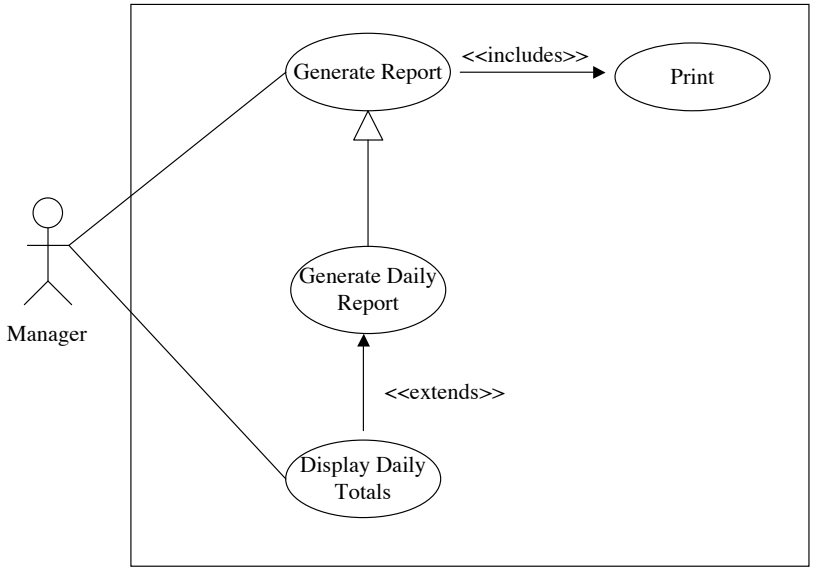
Relationships

- Association
  - The participation of an actor in a use case, i.e. instances of the actor and instances of the use case communicate with each other.
  - Multiplicity man be specified.
- Generalization
  - A generalization from use case A to use case B or from actor A to actor B indicates that A is a specialization of B.

Relationships (2)

- Extend
  - An extend relationship from use case A to use case B indicates that an instance of use case B may be extended by the behavior specified by A. The behavior is inserted at the location defined by the *extension point* in B which is referenced by the extend relationship.
- Include (formerly uses)
  - The inverse of extend, but the intent is that B is an *abstract* use case.

Relationships (3)



Generalization/specialization

- Generalization/specialization may be useful for decomposing use case diagrams into greater detail.
- An inheritance hierarchy of actors is often useful to model various levels of access permissions to a system.

Extend/Include

- Model optional parts of the system
  - May form the basis for incremental development
- Model complex and alternative courses which seldom occur
- Model separate sub-courses which are executed only in certain cases.
- Model that one of several sub-courses may be inserted (e.g. menu system with options)

Use case descriptions

- Describe the use cases in detail
- May include ordinary text, state transition diagrams, etc.
- Basic course
  - Most important course giving the best understanding of the system
- Alternative courses
  - Describe variations and exceptions

Example: Generate Sales Report

Basic Course

1. *Salesperson* chooses "Report Sales" from the "Report" menu.
2. The system displays the salesperson login screen (see figure 6.2).
3. *Salesperson* enters name, employee number, and ID.
4. The systems verifies the employee information.

…

15. The salesperson is logged off of the system.

Alternative Courses

Invalid ID number.

> If the system is unable to verify the employee information in step 4, a message is displayed indicating the error, the failed attempt is recorded in the system error log, and the user is asked to reenter the information (return to step 2).

Key points

- Requirements are not "object-oriented".
- Object-oriented concepts are useful for expressing requirements.
- Use case diagrams show actors, use cases, and the relationships between them.
- Extension and specialization are useful for decomposing requirements.
- Use case details may be expressed formally or informally using standard notations.