Tuesday, November 8

Announcements

- Homework 7: Coding Dojo
  - Work through a coding problem
  - Test-driven development, pair programming, IDE use
  - Conflicts with other classes will be accomodated
- Quick survey on re-designing ics.jobs
  - http://eee.uci.edu/survey/ICS_Jobs
- Undergraduate App Jam Competition
  - Start: Monday, November 14th at 7:00pm in ICS 432
  - End: Monday, November 21st at 11:59pm in a dropbox
  - Judging: Tuesday, November 22nd at 6:00pm to 8:00pm in DBH 6011
  - Email: carlina@uci.edu and icsscaaachair@gmail.com

UML - Behavioral Diagrams

Types of UML Diagrams

Structure

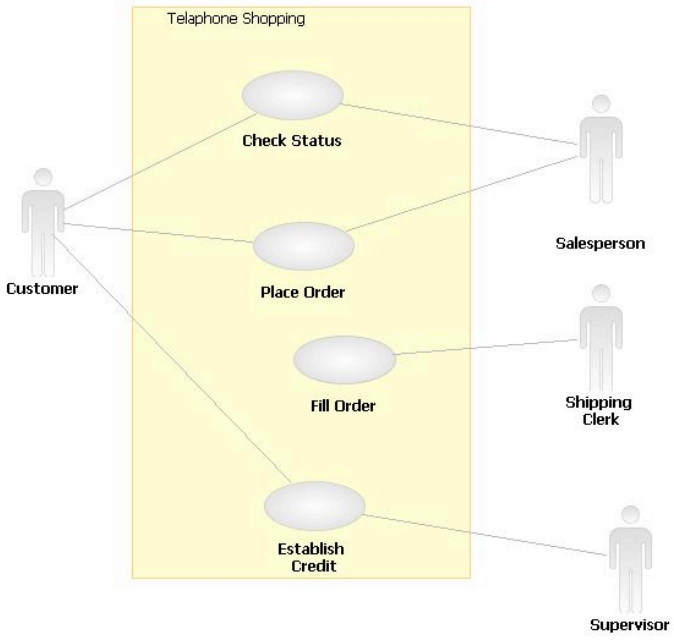
- Class diagram
- Object diagram
- Package diagram
- Composite structure diagram
- Component diagram
- Deployment diagram

Behavior

- Activity diagram
- Use case diagram
- State machine diagram
- Interaction diagrams
  - Sequence diagram
  - Communication diagram
  - Interaction overview diagram
  - Timing diagram

Use Cases

Use Case Diagram



Use Cases

- Use case diagrams are just visual representations of use cases
- Use cases are text
  - This is the important stuff
- Larman, p. 64
  - Use cases are text documents, not diagrams, and use-case modeling is primarily an act of writing text, no drawing diagrams.

Use Cases

- Simple
  - Written in natural language
  - Different stakeholders can participate
- Emphasize user goals and perspective
  - Helps keep the big picture in focus

Use Cases

- A *use case* is a typical sequence of actions that a user performs in order to complete a given task
  - The objective of *use case analysis* is to model the system
    … from the point of view of how users interact with this system
    … when trying to achieve their objectives.
  - A *use case model* consists of
    - a set of use cases
    - an optional description or diagram indicating how they are related
- A requirements gathering technique

Elements of Use Cases

- Actor
  - Something with behavior, e.g. person (really a role), computer system, or organization
- Scenario
  - Specific sequence of actions
  - Also use case instance
- Main Success Scenario
  - A typical, unconditional path to success

How to write use cases

- How are you going to use your use case? This will help with later decisions.
  - Audience
    - Customers, managers, developers, testers, etc.
  - Level of detail
    - Brief – A 1-paragraph summary
    - Casual- Multiple paragraphs, informal text
    - Fully dressed- All steps and variations in detail
      - A use case template is helpful
  - Level of granularity
- Start by identifying the system boundary

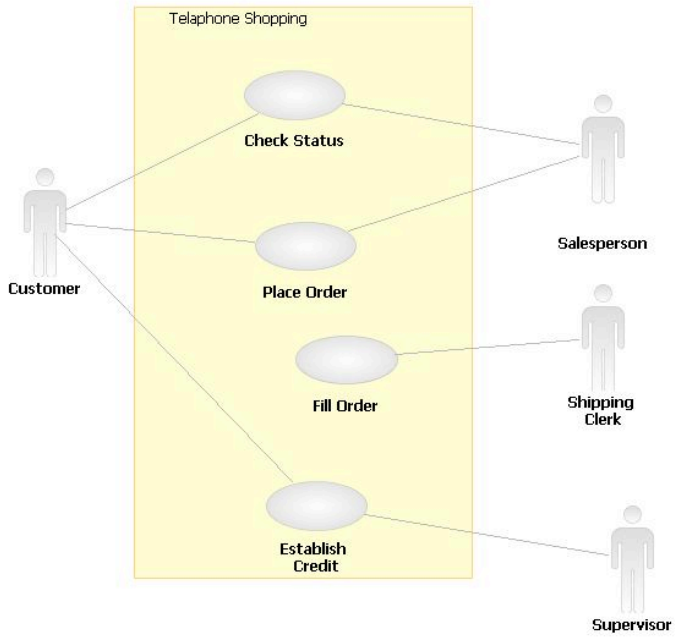
Three Tests for Level of Granularity

- The Boss Test
  - Ask “What have you been doing all day?”
  - Would the answer make the actor’s boss happy?
- The EBP (Enterprise Business Process) Test
  - Task performed by one person in one place at one time
  - Responds to a business event
  - Adds business value
- The Size Test
  - More than a single step
  - Fully-dressed version is usually 3-10 pages

Elements of Use Case Diagrams

- The main purposes of a use case diagram:
  - Show all the names of the use cases, like a table of contents
  - Show relationships between actors and use cases
  - Show relationships between use cases
- Stick figure: Actor
- Oval: Use case
- Extension
  - Optional interactions to cover exceptions
- Inclusion
  - For common substeps; can be re-used in diagram
- Generalization
  - Like superclasses; for representing several similar use cases

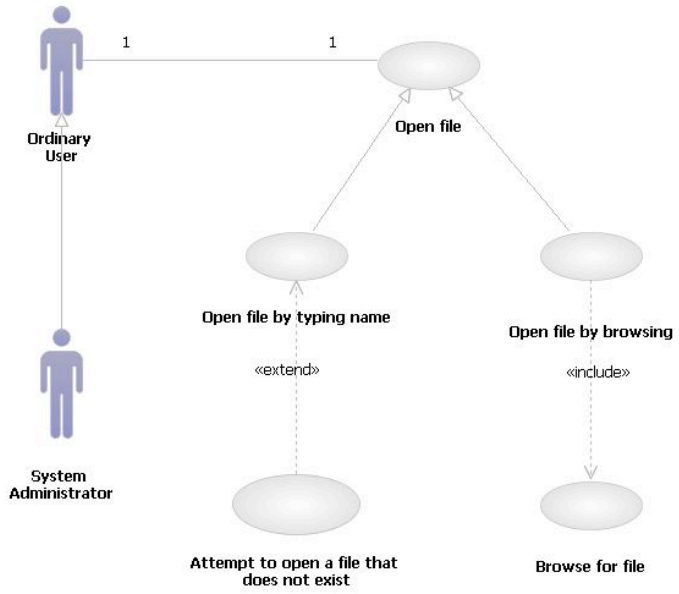
Use Case Diagram

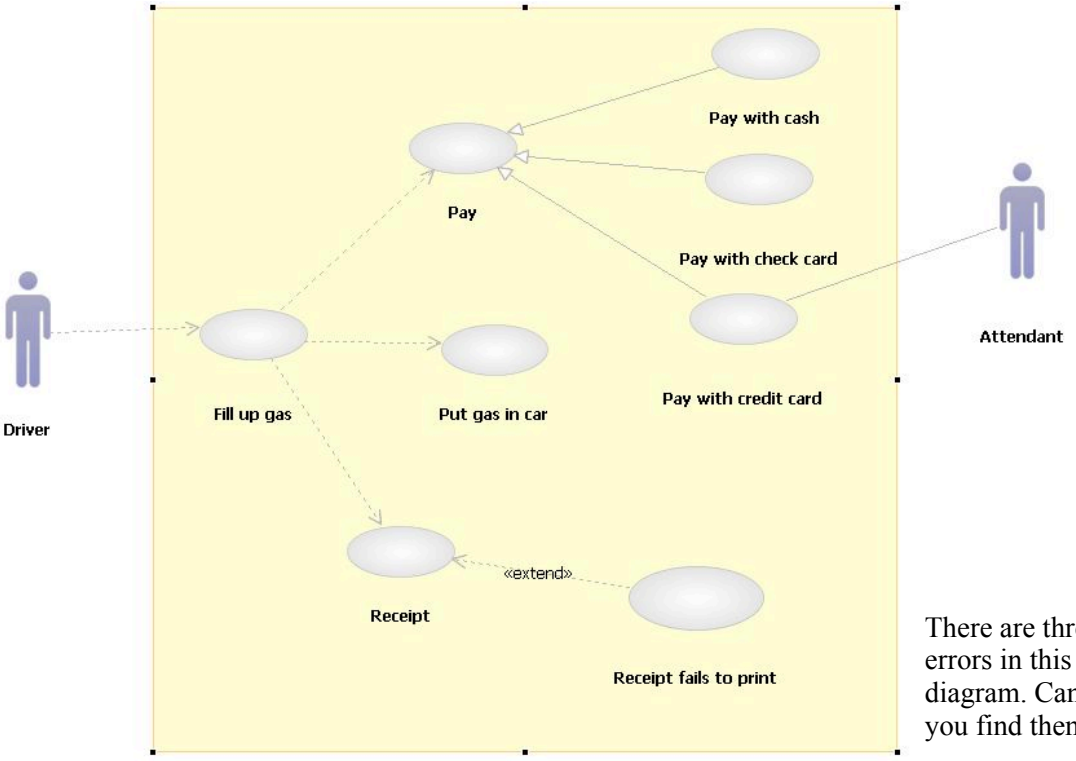


Elements of Use Case Diagrams

- Extension
  - Optional interactions to cover exceptions
- Inclusion
  - For common substeps; can be re-used in diagram
- Generalization
  - Like superclasses; for representing several similar use cases

Example of generalization, extension and inclusion





There are three errors in this diagram. Can you find them?

Sequence Diagrams

- System Sequence Diagrams
  - For showing the interaction actors and the software system
  - Next step in gathering requirements after writing use cases
- Object Sequence Diagrams
  - For showing the interaction between objects
  - Next step in designing classes and objects, after high-level design (architecture, packages, initial classes)

Thursday, November 10

System Sequence Diagram

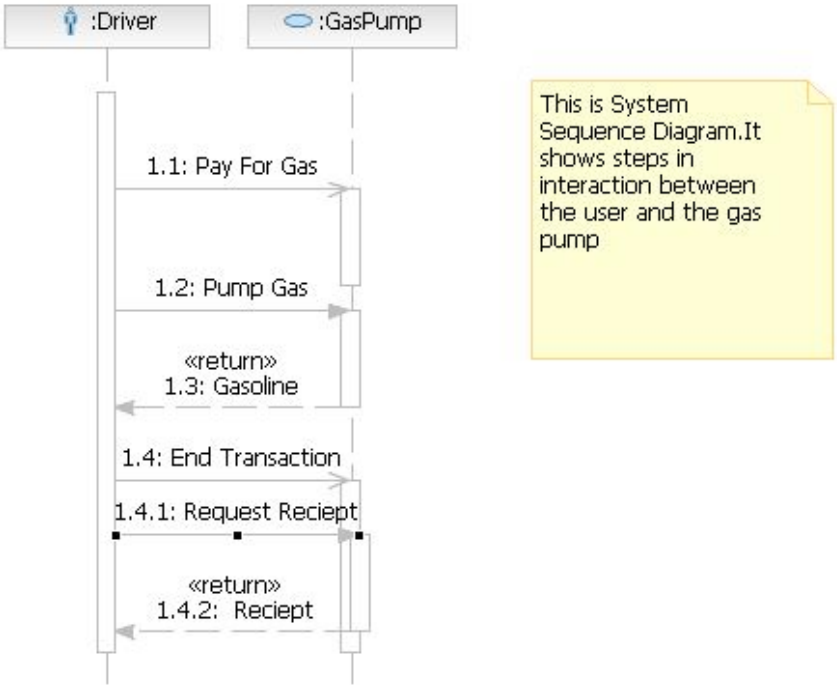
- Simpler syntax (fewer elements in the vocabulary)
- Simpler diagrams (says less)

- Shows system level events
- Focus on one scenario for one use case

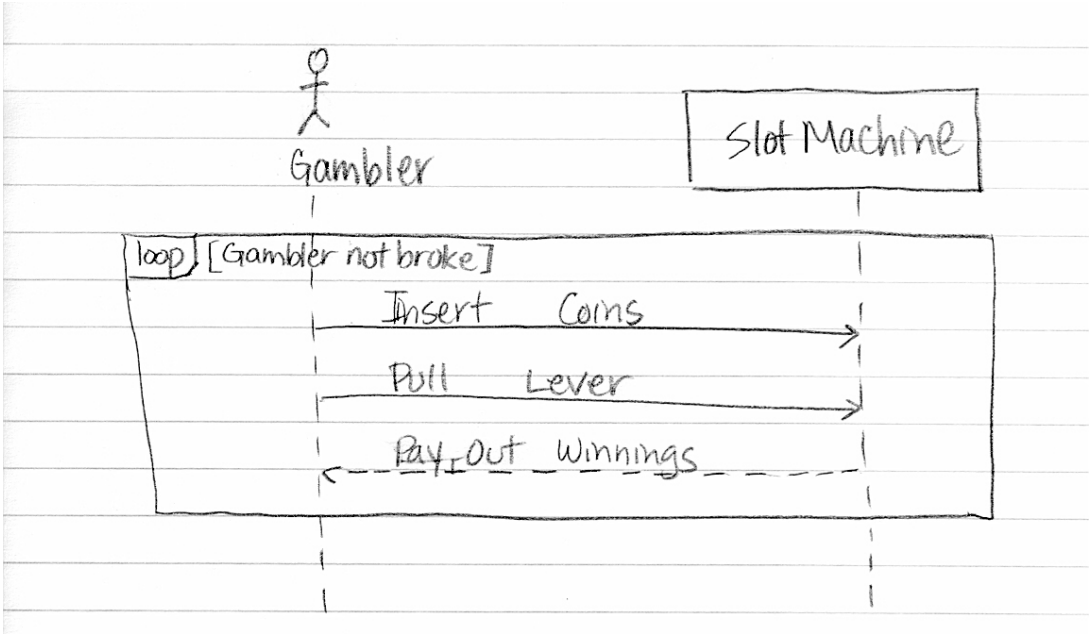
Elements of System Sequence Diagrams

- Objects
  - Life lines: time goes from top to bottom
- Messages
  - Analogous to method calls in a program
    - Can have parameters
  - Return messages
- Looping Constructs

Pumping Gasoline



Frame and Guard



Frame Operator

- Keyword specifies the type of frame
- Guard specifies condition
- Keywords
  - `loop` –Loop fragment while guard is true
  - `opt` –Optional fragment that executes if guard is true
  - `alt` –Alternative fragment for mutual exclusion conditional logic expressed in the guards
  - `par` –Fragments that execute in parallel
  - `region` –Critical region within which only one thread can run

Object Sequence Diagram

- More detail
- More syntax
- Can be transformed into source code

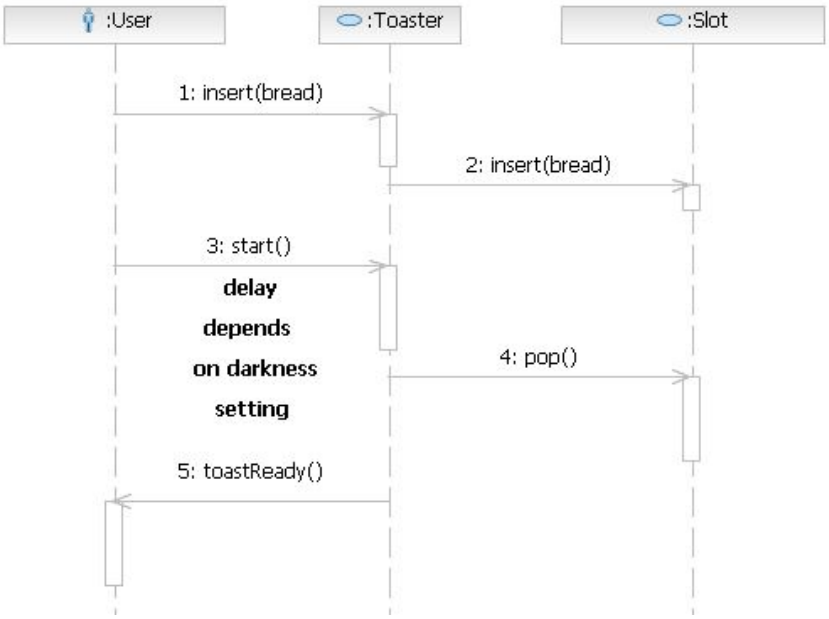
Objects

- Lined up along the top
  - May be several instances of one class
- Defines columns
- Life lines
  - Time goes from top to bottom
- Activation Bars
  - Boxes on life lines show if object is active

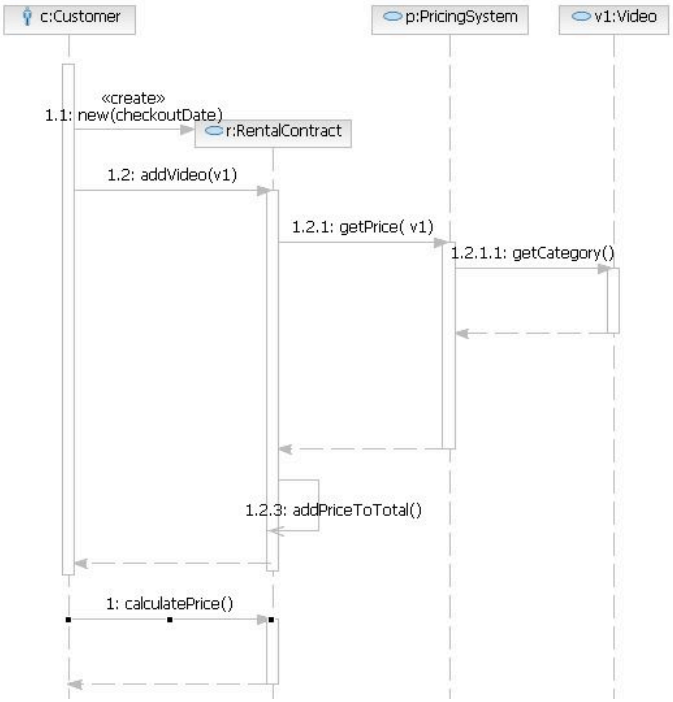
Messages

- Analogous to method calls in a program
  - Can have parameters
- Synchronous messages
  - Calling object waits for call to complete
- Asynchronous messages
  - Calling object does not wait for call to complete
- Special messages
  - new — shown by position of object
  - delete — shown through big X
- Return messages
- Self-calls

Making Toast

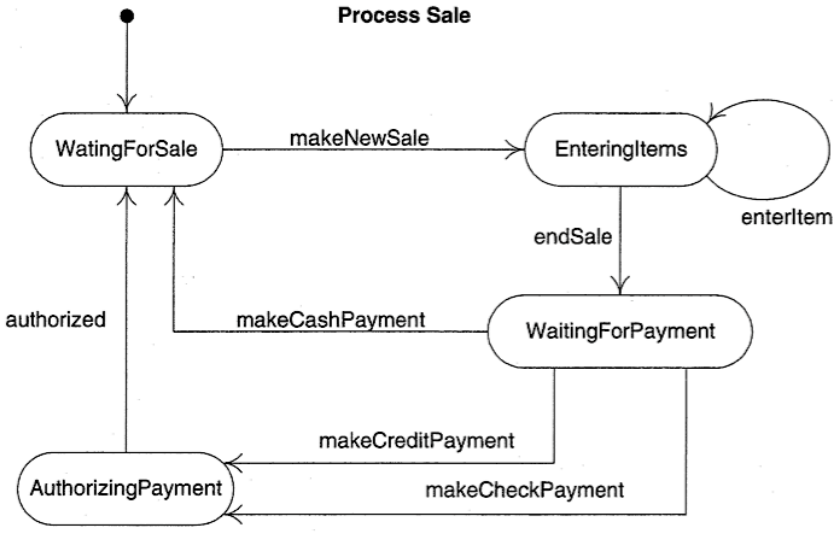


Video Rental



State Machine Diagram

- Some objects have states and some don't
  - State-independent objects respond the same way an event
  - State-dependent objects respond differently depending on their internal state
    - Need State Machine Diagrams to fully describe these kinds of objects
- Used to show the events and states in the life of an object
- Also called State Diagrams



Elements of State Machine Diagrams

- State
  - Rounded rectangle
  - Condition of an object at some time, delineated by events
  - May be nested in side each other
- Transition
  - Line with line arrowhead
  - Represents movement from one state to another
- Event
  - Occurrence that triggers a transition
  - Can have guard condition

State Machine Diagram

- Also model dynamic aspects
  - Show how an object reacts to events depending on its state or mode
  - Model the behavior of a complex object, e.g. physical devices, transactions, and UI Navigation
- Show events and states of an object in the system
  - State = condition of an object at a moment in time
  - Event = a significant occurrence
  - Transition = a relationship indicating an object move from a prior state to another state

Telephone

- Guard condition
- Nested States