Software Tools & Methods
Class 8

Lecturer: Greg Bolcer, greg@bolcer.org

Summer Session 2009

ELH 110 9am-11:50am

Overview

- Last Class
  - Scrum Review
  - Testing
- This Class
  - eXtreme Programming review
  - Use cases
- Next Class
  - Final

Software Methods & Tools

EXTREME PROGRAMMING

Extreme Programming (XP)

- Kent Beck invented XP by the seat of his pants in 1996 to fix a disastrous project at Chrysler
- "Extreme Programming Explained: Embrace Change" by Beck published in 1999
  - Version 2.0 published in 2004
  - Tries to associate with a software process with extreme sports
- Idea: Take a good programming practice and push it to the extreme.
  - Example: testing
  - Since testing is good, do it all the time, even before we have code.

Current Reasons for Popularity

- Effectiveness
- Results
- Compatibility with web applications
- Cool factor

Extreme Practices

- If code reviews are good, we'll review all the time (**pair programming**).
- If testing is good, everybody will test all the time (**unit testing**).
- If design is good, we'll make it part of everybody's daily business (**refactoring**).
- If simplicity is good, we'll always leave the system with the simplest design that supports the functionality (**the simplest thing that could possibly work**).
- If architecture is important, everybody will work defining and refining the architecture all the time (**metaphor**).
- If integration testing is important, then we'll integrate and test several times a day (continuous integration).
- If short iterations are good, we'll make the iterations really, really short– seconds and minutes and hours, not weeks and months and years (the Planning Game).

Twelve Key Practices of XP

Programmer Practices	Simple Design Test-driven development Refactoring Pair programming Collective code ownership Continuous integration Coding standards
Management Practices	Planning Game Small releases 40-hour week
Customer Practices	On-site customer Metaphor

XP Metaphor

- Metaphor is the architecture of the system described in a way that facilitates communication
- Practice of Agile most ignored by practioners, though communication is cited as key value
  - Giving examples, cleaner way to describe
  - Check against metaphor after first design
  - Check against metaphor after completed system
  - Drawings and words that capture aspects of the system
  - Create unique language and meaning that is used as a label

Software Methods & Tools

PLANNING GAME

Planning Game

- Planning and project management are tricky to get right, burdensome, and can be the most emotionally charged part of developing software
- The Planning Game creates some distance between planning and participants by treating it like a game

Planning Game

- Like most games, has Goal, playing pieces, players rules
- Pieces: A User Story, each story written on an index card
- Goal: to put the greatest possible value of stories into production over the life of the project
- Players: business and development

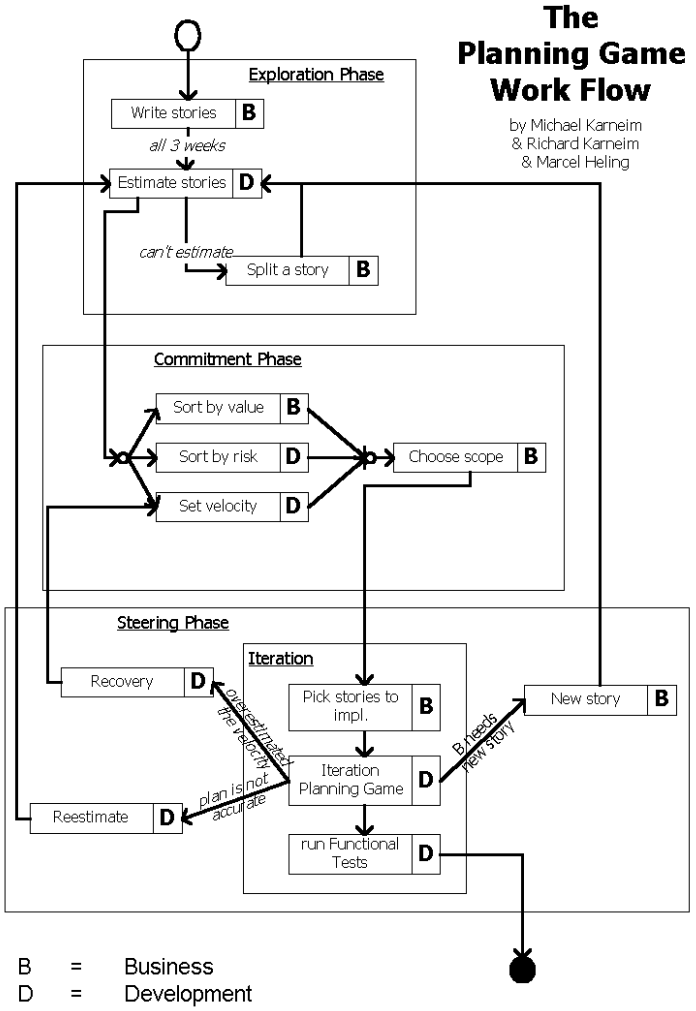
Planning Game Moves

- ***Write story:*** business writes a story at any time and assign a value (with enough info that it can be assigned a dev cost),
- ***Estimate story:*** development takes story and assignes and estimate of 1, 2, 3 weeks of Ideal programming time; 3 real weeks = 1 Ideal week
- ***Make commitment:*** business and development pick which stories make the cut for next release
- ***Story driven commitment:*** Development agrees to take on story for next release until time commitment filled up
- ***Date driven commitment:*** Business picks a release date, development calculates the cost of stories they can accomplish by that date, business picks stories that add up to that number
- ***Value and risk first:*** Development orders stories in commitment so
  - Fully working but sketchy system is done immediately, first two weeks
  - More valuable stories are moved earlier in schedule
  - Riskier stories are moved earlier in the schedule

Planning Game Moves

- ***Overcommitment Recovery:*** Development provides new estimates, e.g. 100 units of value vs 150, business repicks stories to implement (or defers deadline)
- ***Change Value:*** Business changes the value of a story, development changes order of stories in current commitment
- ***Introduce New Story:*** Business writes new story, development assigns costs, re-evaluates Value and Risk
- ***Split Story:*** Business splits a story into two or more because resources do not allow it to be done all at once
- ***Spike:*** Business writes a story to divert resources to do a throwaway spike to fight a fire, reduce risk, or prove a concept
- ***Re-Estimate:*** Development is asked at any given point in time to re-estimate the remaining stories

Planning Game Workflow



Software Methods & Tools

AGILE METHODS REVIEW

Rational Unified Process

RUP is a software development process that is:

- Use case-driven
- Architecture-centric
- Iterative and incremental

Use Case Driven

- *Use cases* are the primary artifact for establishing the behavior of the system, verifying and validating the system's architecture, testing, and communicating with stakeholders.

- Start with use cases.
- Talk to users to find out what the system is supposed to do.

Architecture-centric

- *Architecture-centric* means that a system's architecture is used as the primary artifact for conceptualizing, constructing, managing, and evolving the system under development.

- Create a big picture of the system first.
- Then add details.

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

Software Methods & Tools

USE CASES

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


TURN OFF ENGINE
NO SMOKING
CARD READER
PAYMENT KEYS
CASH
CREDIT
ATM
NO
YES
TO START PUMP
SELECT PAYMENT KEY
THEN FOLLOW INSTRUCTIONS
ON DISPLAY SCREEN
PLEASE
PAY FIRST
THIS SALE $
GALLONS
PRICE PER GALLON $
ALL TAXES INCLUDED
TURN OFF ENGINE
NO SMOKING
1 Remove Nozzle
2 Select Grade by pressing red button
3 Operate Handle
Turn Off Cellular Phones, Pagers and Personal Electronic Devices While Fueling.
8
PREMIUM UNLEADED
PRICE PER GALLON $
91
ALL TAXES INCLUDED
PUSH
UNLEADED PLUS
PRICE PER GALLON $
89
ALL TAXES INCLUDED
PUSH
UNLEADED
PRICE PER GALLON $
87
ALL TAXES INCLUDED
PUSH

Use Cases

- In general, a use case should cover the full sequence of steps from the beginning of a task until the end.
- A use case should describe the user's interaction with the system ...
  - not the computations the system performs
  - not a state machine
- A use case should independent of any particular user interface design.

Advantages of Use Cases

- Use cases are fast and easy to write (after some practice). They require no training to read.
- Use cases are fairly concrete examples of how the system could be used.
- Use cases are examples of usage. Starting with just a few examples is easy. More examples can be added incrementally as needed.

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

How to write use case

- Choose the System Boundary
  - Who or what are on the outside? On the inside?
- Find Primary Actors and Goals
- Define Use Case
  - Start the name of use cases with a verb
- Write Main Success Scenario
  - The use case should focus on one success.
  - Possible error conditions or other failures are simply noted as extensions to the main success scenario.
- Detail the steps to be take to achieve the desired goal
  - Include enough detail to check for correctness

How to write use cases

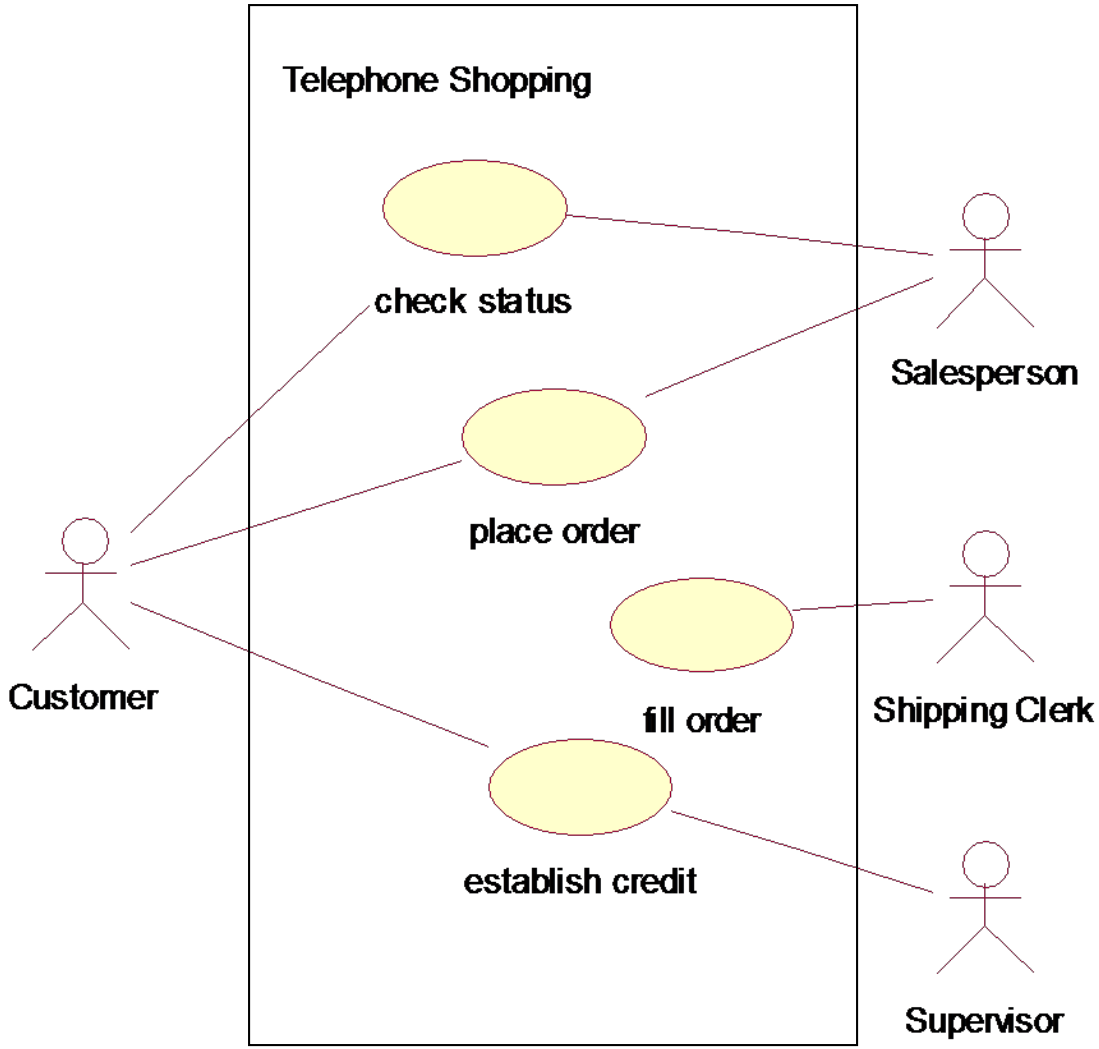
- As you go, add notes to yourself and questions to come back to.
- Are there any preconditions that you assumed?
- Are there alternative ways to accomplish the same thing? E.g., cash withdrawal vs. fast cash. You can note them as extensions, or write whole new use cases.
- Are there ways that the user can fail? Note some as extensions.

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


TURN OFF ENGINE
NO SMOKING
CARD READER
PAYMENT KEYS
CASH
CREDIT
ATM
NO
YES
TO START PUMP
SELECT PAYMENT KEY
THEN FOLLOW INSTRUCTIONS
ON DISPLAY SCREEN
PLEASE
PAY FIRST
THIS SALE $
GALLONS
PRICE PER GALLON $
ALL TAXES INCLUDED
TURN OFF ENGINE
NO SMOKING
1 Remove Nozzle
2 Select Grade by pressing red button
3 Operate Handle
Turn Off Cellular Phones, Pagers and Personal Electronic Devices While Fueling.
8
PREMIUM UNLEADED
PRICE PER GALLON $
91
ALL TAXES INCLUDED
PUSH
UNLEADED PLUS
PRICE PER GALLON $
89
ALL TAXES INCLUDED
PUSH
UNLEADED
PRICE PER GALLON $
87
ALL TAXES INCLUDED
PUSH

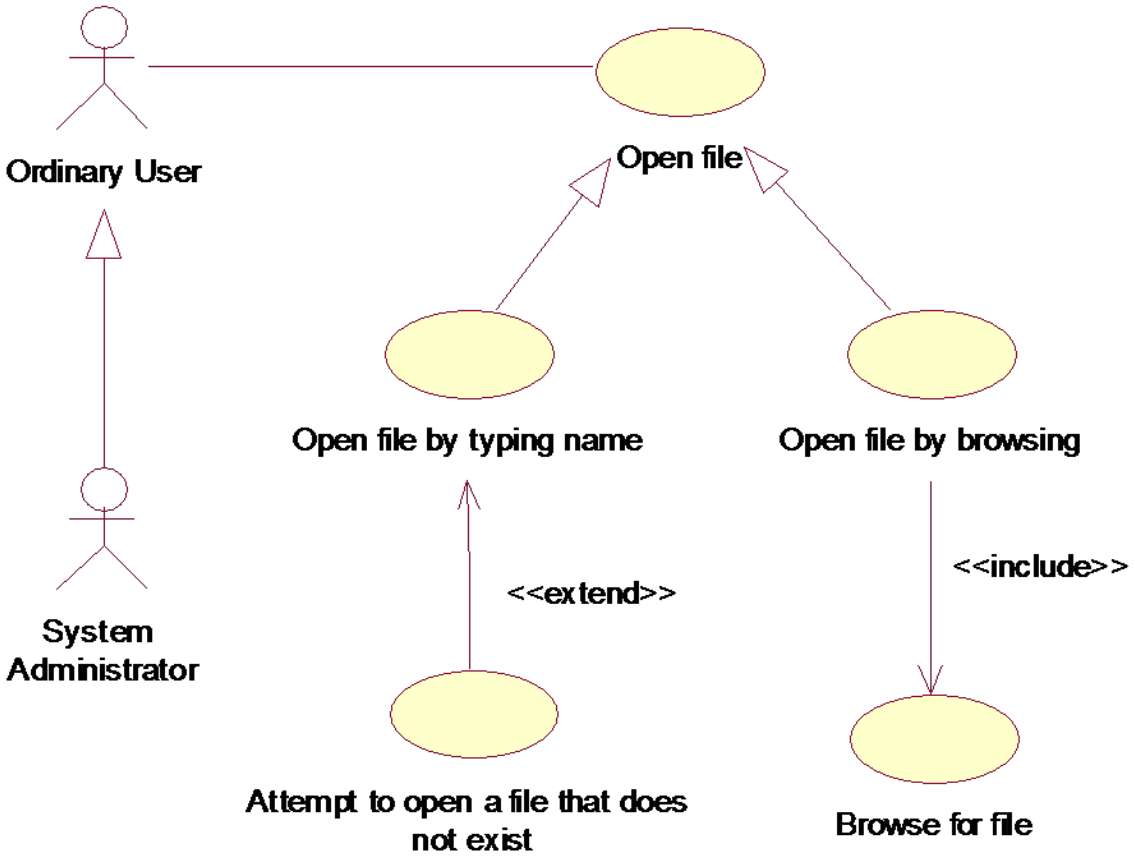
Use Case Diagram



Elements of Use Case Diagrams

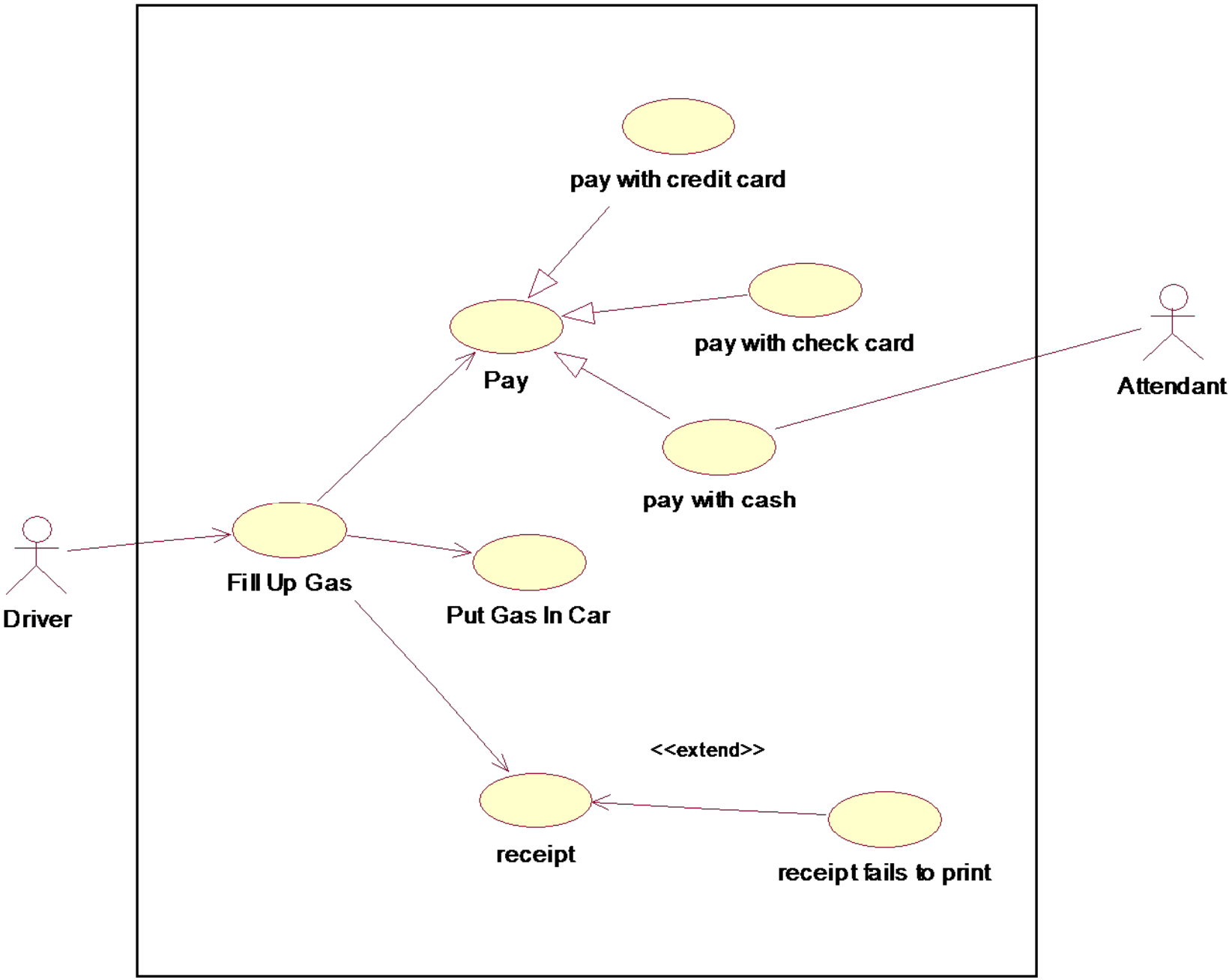
- Extension
  - Optional interactions to cover exceptions
- Inclusion
  - For common substeps; can be re-used in diagram
- Generalization
  - Like superclasses; for representing several similar use cases

Example of generalization, extension and inclusion



Whiteboard Exercise (Use Case)

A driver needs to fill up their car with gas. The station requires the driver to pay first. Payments methods include paying with a credit card or check card at the pump, or paying with cash which is collected by the attendant inside the station. Once a payment method is established, the attendant approves the distribution of gas at the particular pump and the driver fills up the car. After the car is full, the pump will print a receipt. If no receipt is available, the driver may request a receipt from the attendant.


pay with credit card
pay with check card
Pay
Attendant
pay with cash
Driver
Fill Up Gas
Put Gas In Car
<<extend>>
receipt
receipt fails to print

Software Methods & Tools

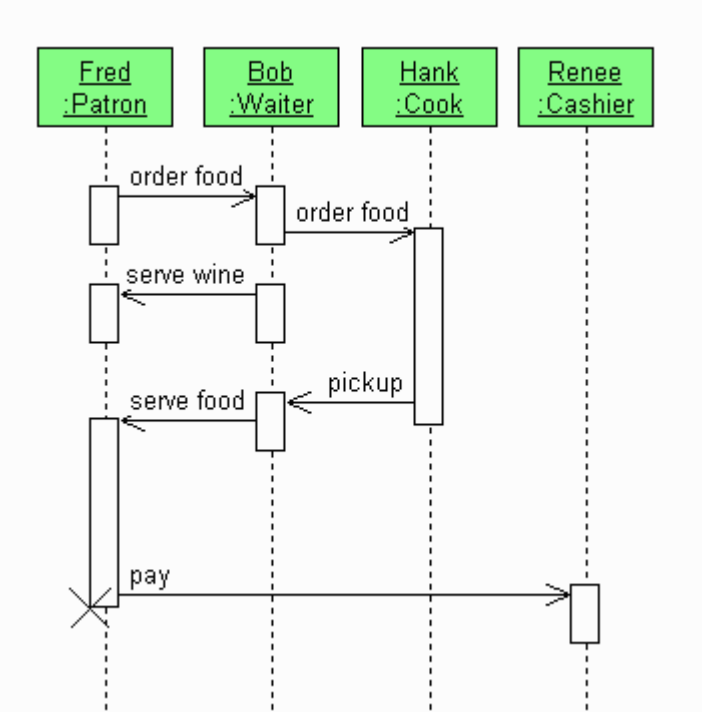
SEQUENCE DIAGRAMS

Sequence Diagrams

- Are a method of visually coding sequences of events
- Like all models, used as an abstraction
- Includes time based durations and precedence ordering
- Actors are listed on top of diagram in boxes with dashed lines vertically below
- Entry state and exit states similar to UML diagrams, can have a closed ball with an arrow and/or an “X” to exit.
- Partial arrows represent asynchronous events where the actor doesn’t wait for the completion of the action before continuing

Use Case

Fred is a patron at a local restaurant. He goes in, sits down, and orders food and wine from the Waiter, Bob. Bob passes the food order on to Hank the cook and comes back to serve Fred the wine. When the food is ready, Bob picks up the food order from Hank and serves the food to Fred. When Fred is done with his meal, he pays Renee, the cashier and leaves.



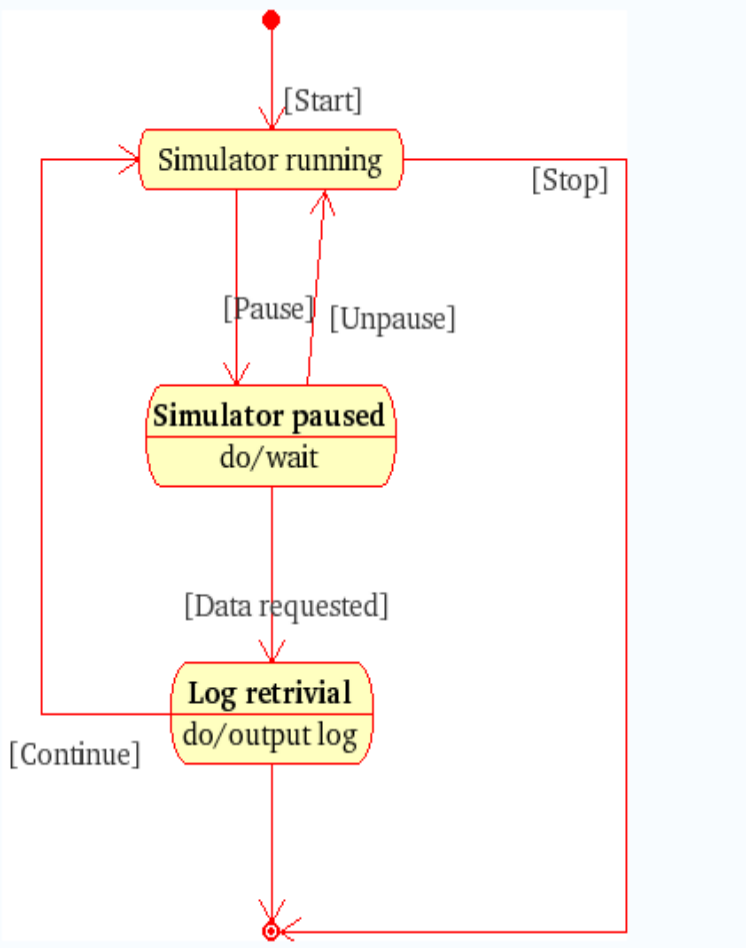
Software Methods & Tools

STATE DIAGRAMS

State Diagrams

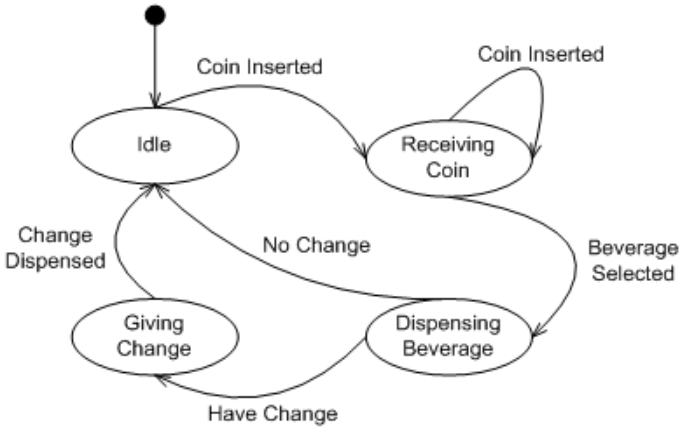
- A visual diagram that models a software system using a discrete number of states and transitions
  - Represents a reasonable abstraction
  - Used to describe behavior of system
  - Taylor Booth, 1967, Finite State Machines
- UML State Diagram standard format

UML State Diagrams



- **Filled circle**—pointing to initial state
- **Hollow circle containing a smaller filled circle**, indicating the final state (if any)
- **Rounded rectangle**, denoting a state. Top of the rectangle contains a name of the state. Can contain a horizontal line in the middle, below which the activities that are done in that state are indicated
- **Arrow**, denoting transition. The name of the event (if any) causing this transition labels the arrow body. A guard expression may be added before a "/" and enclosed in square-brackets (***eventName*[guardExpression]**), denoting that this expression must be true for the transition to take place. If an action is performed during this transition, it is added to the label following a "/" (***eventName*[guardExpression]/action**).
- **Thick horizontal line** with either x>1 lines entering and 1 line leaving or 1 line entering and x>1 lines leaving. These denote join/fork, respectively.

State Transition Example



- Start at initial state
- Outgoing arrows represent state changes
- When in a given state, only the transitions listed are possible

Final Announcements

- Follow the final study guide
- Make sure you understand today's lecture and can read use cases, state diagrams, sequence diagrams
- If you are turning in HW8 for extra credit, it's only replacing lost points on other homework
- 1 person submitted multiple choice question, will be used on final
- Any and all issues: email me or coordinate with ugrad counseling office
- Hope to have final grades posted on EEE by 29th