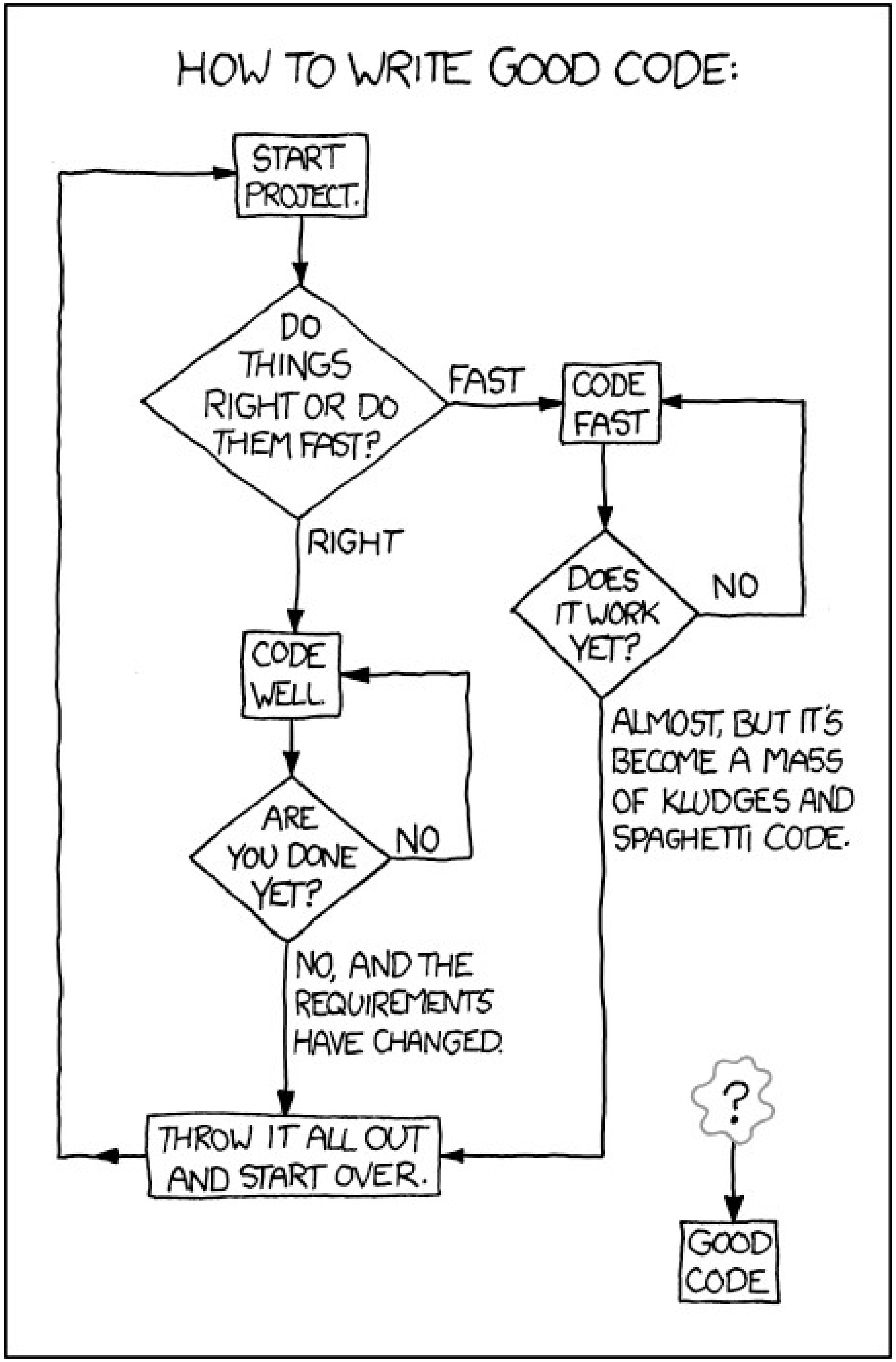
Software Design

Requirements

Slides are adapted from various software engineering classes offered by Prof. Tao Xie at UIUC

This week’s goals

- □How can we document requirements?
- □What are different ways of writing Use Cases?
- □When do Use Cases (not) fit for capturing requirements?


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
DOES IT WORK YET?
NO
CODE WELL.
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
?
THROW IT ALL OUT AND START OVER.
GOOD CODE

http://xkcd.com/844/

Primary project killers

- Requirements mistakes
  - Wrong requirements
  - Ever-changing requirements
  - Ever-increasing requirements (“creep”)
- Improper schedule

Other people for requirements

- Business analyst
  - analyze an organization or business domain and document its business, processes, or systems, assessing the business model or its integration with tech
- Systems engineer
  - also develop software components, but also specify, build, maintain and support technical infrastructure (e.g., build, test and production environments used to deliver SaaS, and systems for performance monitoring, and as on-call support engineer)
- Project lead
- Program manager
  - PM at Microsoft (by Steven Sinofsky): ”do not program nor do they manage”, “working in partnership with expert designers, clever developers, super smart testers, etc. you all work together to define the product AND the responsibilities each of you have.” http://blogs.msdn.com/techtalk/archive/2005/12/16/504872.aspx

What’s in requirements

- Example items
  - Reports to be generated
  - UI for a game
  - Behavior of system
  - Data formats
  - …

Requirements

- Functional requirements
  - Inputs, outputs, and the relations between them
- Non-functional requirements (-ilities)
  - Security
  - Reliability
  - Efficiency
  - Usability
  - Scalability
  - Maintainability
  - Portability
  - …

Functional requirements I

□ Inputs, outputs, and the relationship between them

□ Use Cases are one example we'll look at

□ Formats, standard interfaces

□ Business rules and complex formula

Functional requirements II

- Command language
  - The “get” command will transfer ...
- Web pages
  - Input forms, dynamic pages
- Connections to other systems
  - Files, sockets, XML, …
- Reports, displays

Non-functional requirements I

- Performance
  - Must answer a query in 3 seconds
- Usability
  - New user must be able to finish buying a book in 15 minutes
  - 90% of users must say they like interface
- Maintainability
  - New programmers should be able to fix first bug in two weeks on the job

Non-functional requirements II

- Technology
  - Must use Java
- Business
  - Must get it finished in 1 year spending less than $1,000,000

Accuracy

- Functional requirements - in theory, can specify completely
- Non-functional requirements - hard to specify, can't specify completely

- Requirements should be specific, so you can tell whether you met them
- Requirements should be as precise as necessary, but no more

System description

- Typical description has two parts
  - Data
  - Operations on that data

- Three possible descriptions
  - Requirements
  - Specification
  - Design

Many notations

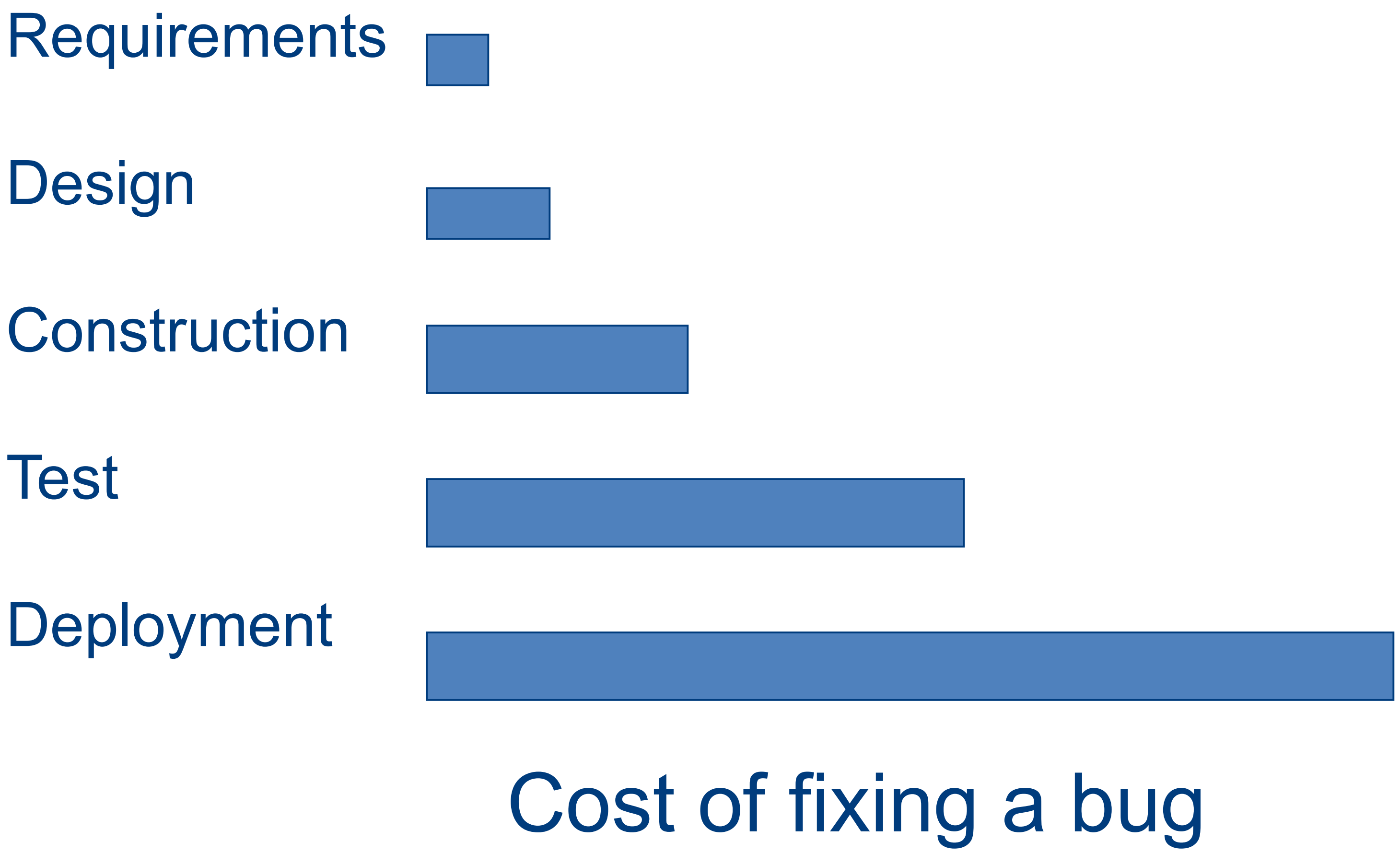
- UML
  - Use cases
  - Class diagram
  - State diagram
- Finite state machine
- Data flow diagram
- Pseudocode
- Programming language

Many purposes

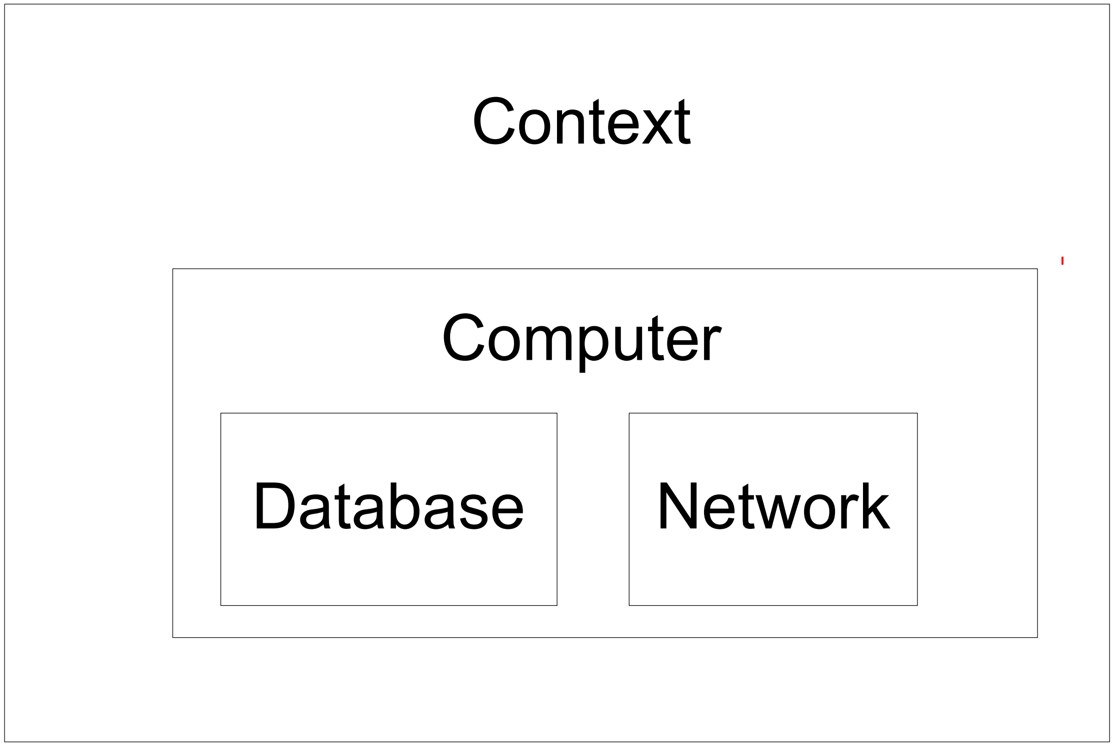
- Communicate to
  - User
  - Developers
  - Boss

- Complete - lots of detail
- Easy to read - less detail

Cost of change curve



Requirements analysis



Context

Computer

Database

Network

Questions to ask

- How do you learn
  - What the problem is?
  - What you can change?
  - What the computer should do?
  - Whether you were correct?

Discovery techniques

□Reading

□Interviews

□Teams

□Creating requirements document

□Building models

□Building prototypes

Requirement document

- List requirements
- Name requirements
- Categorize requirements by
  - Source
  - Feature
  - Subsystem
  - Type

Summary of requirements

- Getting the right requirements is crucial and hard. Key part of software engineering
- Must communicate with users
- Models are helpful in analyzing and communicating requirements

Use Cases

- One way to express requirements
- Invented by Ivar Jacobsen
- Popularized by Alistair Cockburn
  - consult his book’s draft version http://alistair.cockburn.us/get/2465

Use case diagram

□Shows context - what is in and out of the system

□Shows scope and boundaries

□Shows actors and use cases

□Shows relationships between actors and use cases

□Does not show a description of use cases

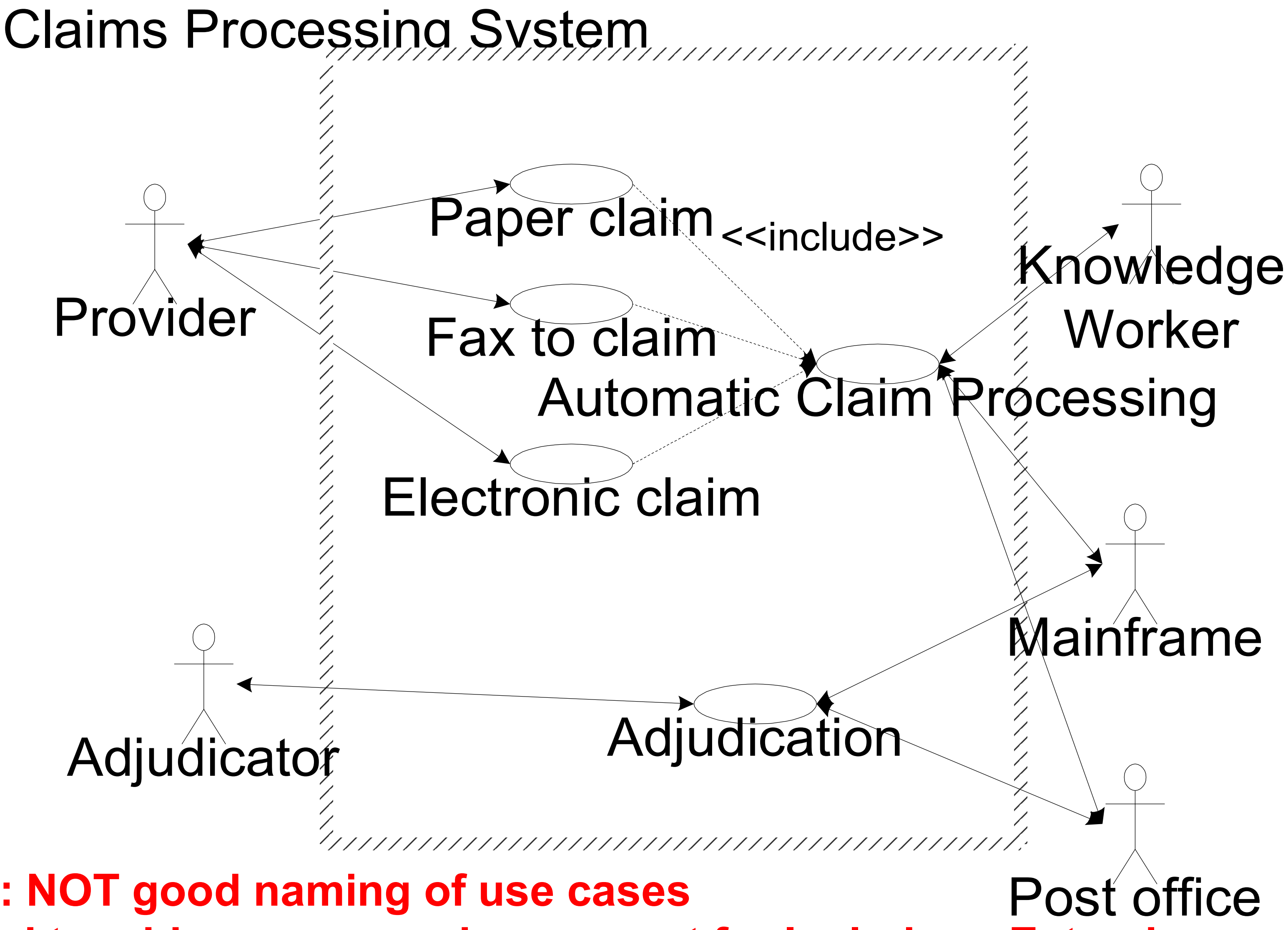
Main entities

□Actor: Role of something or someone that interacts with System

□Use case: Something that the System does, or that happens to the System. Always involves an actor

□System: The thing you are studying

Example Use Case Diagram



Caveat: NOT good naming of use cases
NO need to add arrows on edges except for Include or Extend relationships, see later slides

Use cases

- Text - a form of writing
- Describes the system’s behavior as it responds to request from a primary actor
  - Actor wants to achieve some goal
- Many kinds of use cases
  - Brief / detailed
  - Requirement / specification / design

Use cases are text

- Use cases should be easy to read
  - Keep them short
  - Tell a story
  - Write full sentences with active verb phrases
  - Make the actors visible in each sentence

Use cases are event sequences

- Describe the sequence of events that happen when the system responds to a request
  - Can describe alternatives (note that a scenario doesn’t have alternatives)
  - Can describe errors

Parts of a use case

- □Primary actor – who starts it?
- □Why the use case? Primary actor has goal.
- □Normal steps
- □Alternative steps

Four kinds of use cases

- Actor-goal list
- Use case brief
- Casual use cases
- “Fully dressed” use cases

- Names from “Writing Effective Use Cases” by Alistair Cockburn
  - How are XP user stories related?

Goals

- Primary actor always has a goal
- Must pick right level for goal
  - Use case for Amazon - buy a book
  - Higher-level goal - learn something, make someone happy
  - Lower-level goal - provide credit card info

Actor-goal list

- Goals should be on the same level
- Goal should be from point of view of primary actor
- Sort goals by primary actor
- Priority to designer of system, not to actors

Actor-goal list

Actor	Task-level goal	Priority
Provider	Submit paper claim	1
Provider	Submit fax claim	2
Provider	Submit electronic claim	3
Adjudicator	Adjudicate problem	2

Including lower-level goals

□Lower-level goals can improve readability

□To design components

□When user has only one goal

Actor-goal list for games

- Games have one use case - play game
- For use cases to be good requirements document, more detail is needed
  - Use lower-level goals

Example: Tower game

Actor	Task-level goal	Priority
User	Create towers	1
Tower	Shoot monsters	2
Monster	Move toward user	2
Horde	Create monsters	3

Use case brief

- 1-6 sentence description of behavior
- Mention only most significant behavior and failures
- Short enough to put many on a page
- Used to
  - Estimate complexity
  - Find components to reuse

Use case briefs

Actor	Goal	Brief
Provider	Submit paper claim	Submit claim on paper, which is converted into electronic form, and either paid or sent for adjudication
Provider	Submit fax claim	Submit claim by fax, which is processed by OCR and either paid or sent for adjudication.
Adjudicator	Adjudicate failed claim	Decide whether a questionable claim should be paid by the mainframe payment system or rejected

Identify Use Cases from Video Scenarios

- □ Watch a video clip (you are allowed to write notes when watching!): https://www.youtube.com/watch?v=udr9-CN5mXU
- □ Get into groups of 2-3 students
- □ Competition: enumerate as many use cases as possible in the form of Actor-goal list such as (you can leave priority empty)

Actor	Task-level goal	Priority
XXX	YYY	Z
Tower	Shoot monsters	2
Monster	Move toward user	2
Horde	Create monsters	3

Casual & Fully Dressed

- Used to tell developers what to build
- “Casual” consists of normal paragraphs
- “Fully Dressed” has labeled sections
- Both should emphasize “main success scenario”
- Both should include ways of failing
- One per page

Design scope

- Actor-Goal list and use case briefs help decide design scope
  - What is in, what is out
  - What is a use case, what is not a use case
- Casual/fully dressed use case is given to developers so they know what to build

Casual (short) version of Submit Fax Claim

The Provider submits a claim by fax. The claim processing system will log the image to optical disk, apply form dropout, deskewing, despeckling, and then process it using OCR. Knowledge workers will repair any fields that seem to be in error. The claim will then either be paid (using existing mainframe processing system) or sent to adjudication.

Fully dressed

- □Primary actor
- □Goal in context - what is the primary actor’s bigger goal?
- □Scope - system we are designing
- □Level - user goal, higher-level (summary), lower-level (subfunction)
- □Stakeholders

Scope or Level options, see Page 3 of http://alistair.cockburn.us/get/2465

Fully dressed

- Preconditions: things that must be true before this use case can happen
- Guarantees: things that the use case will ensure are always true
- Triggers: things that cause the use case to start

Main success scenario

- Describe trigger
- Give numbered list of actions leading to success
- Alternatives left to “extensions”

Extensions

- Alternatives to main success scenario
- Special cases, failures
- Steps have same numbers as in main success scenario, but modified to show they are alternatives
  - 3a instead of 3

Example fully dressed (detailed) version of Submit Fax Claim

Primary Actor: Provider

Goal in context: Pay legitimate claims while rejecting bad ones.

Scope: Business - the overall purchasing mechanism, electronic and non-electronic, as seen by the people in the company.

Level: Summary

Stakeholders and interests:

Provider: Wants to be paid for services rendered.

Company: Wants to cut costs and avoid fraud.

Precondition: none

Minimal guarantees: Pay only certified providers, pay only for services that are covered by plan, do not pay if there are obvious mistakes.

Scope or Level options, see Page 3 of
http://alistair.cockburn.us/get/2465

Main success scenario:

Trigger: Claim submitted by fax

1. Provider: submit claim by fax
2. Claim system: drop forms, deskew, despeckle, use OCR to convert to electronic form
3. Claim system: check claim to make sure it is legal
4. Mainframe payment system: pay claim

Extensions:

2a. Some fields have low confidence: Knowledge worker corrects

3a. Claim is not valid: send to adjudication

Writing

- □Use present tense
- □Subject should be primary actor, system under design, and secondary actors
- □Verb should be what actor does to successfully move the use case forward
- □Avoid GUI: write in terms of goals, not details of the GUI

Team Writing in Practice

□Team better for brainstorming, reviewing

□Individuals better for writing

□Make list in a team

□Write use cases in ones or twos

□Review as a team

Use cases and requirements

- An important part of requirements
  - You do need more than just use cases
- Help manage requirements
- Help requirement traceability

Traceability

- Traceability - the ability to trace the effect of a requirement and determine who caused it
  - Why do we have this requirement? Who wrote it?
  - How is this requirement met?
  - What requirement caused this design?
  - Backward (code to requirement) and forward (requirement to code) traceability

Manage requirements

- Must agree to change in requirements
  - Usually increases price
  - Must be reviewed
- Make sure each part of design is due to a requirement
- Analyze problems: what was the root cause of this fault/defect?
  - E.g., Orthogonal Defect Classification (ODC)

http://en.wikipedia.org/wiki/Orthogonal_Defect_Classification

Scenarios and use cases

- Scenario is concrete and detailed
  - Names of people
  - $ values, particular dates, particular amounts
- Scenario is a test case
- Use case is a contract, and collects all the scenarios

Goals and use cases

□Actor has a goal for the use case

□System forms subgoals to carry out its responsibility

□Goals can fail

□Use case describes a set of ways for carrying out the goal, and several ways of failing

When use cases don’t work

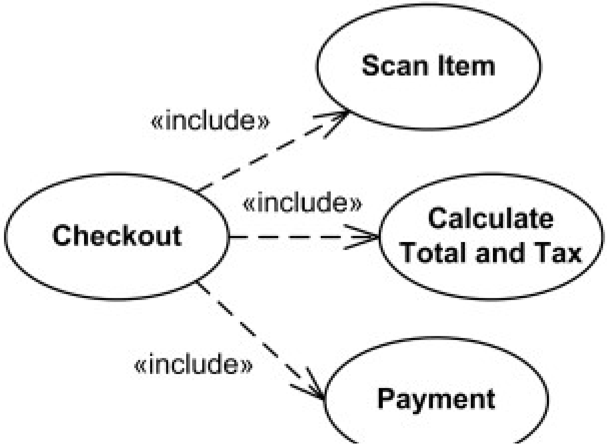
- Compilers
  - One use case - compile a program
- Despeckler
  - One use case - remove speckles
- No interaction
- …

Use Case Diagram: Stereotypes

- Use Case X <u>includes</u> Use Case Y:
  - X has a multi-step subtask Y. In the course of doing X or a subtask of X, Y will always be completed.
- Use Case X <u>extends</u> Use Case Y:
  - Y performs a sub-task and X is a similar but more specialized way of accomplishing that subtask. X only happens in an exception situation. Y can complete without X ever happening.

Use Case Diagram: Include

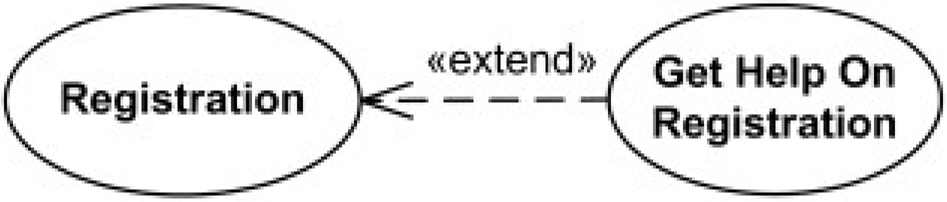
- □ Use Case X includes Use Case Y:
  - ◊ X has a multi-step subtask Y. In the course of doing X or a subtask of X, Y will always be completed.
- □ Arrow direction
  - ◊ "**Include** relationship between use cases is shown by a dashed arrow with an open arrowhead from the including (base) use case to the included (common part) use case."



http://www.uml-diagrams.org/use-case-include.html

Use Case Diagram: Extend

- □ Use Case X extends Use Case Y:
  - ◊ Y performs a sub-task and X is a similar but more specialized way of accomplishing that subtask. X only happens in an exception situation. Y can complete without X ever happening.
- □ Arrow direction
  - ◊ “**Extend** relationship is shown as a dashed line with an open arrowhead directed from the **extending use case** to the **extended (base) use case**.”



http://www.uml-diagrams.org/use-case-extend.html

“Executable use cases”
Behaviour Driven Development

□Cucumber (http://cukes.info/) can execute tests written in structured natural language (as scenarios in use cases)

□Example: test for a calculator program

```
Feature: Division
  Scenario: Regular numbers
    Given I have entered 3 into the calculator
    And I have entered 2 into the calculator
    When I press divide
    Then the result should be 1.5 on the screen
```

Summary of use cases

- Use cases are useful, but not perfect
- Many ways to write use cases
- Big projects need big use cases
- Use the simplest way you can!