

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

7 More Than Just Coding Skills

8 The Big Questions

- What are we solving?
- Who are we solving it for?
- How are we solving it?
- What are the different cultures involved?
- How to choose one methodology over another?

9 More Than Just Coding Skills

10 Methodologies

- Waterfall
- Iterative
- Agile
- Model Driven

11 custom

12 Software development life cycle (SDLC)

The SDLC is the sequence of activities covering

- *requirements*
- *analysis*
- *design*
- *implementation*
- *testing*

over which a software system is developed.

13

Slide 12 of 65

There are four kinds of methodologies:
 Code a bit at a time
 Waterfall
 Rapid prototyping
 Iterative and incremental
 Agile methods

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

11 CANTIN

12 Software development life cycle (SDLC)

The SDLC is the sequence of activities covering

- Requirements
- Analysis
- Design
- Implementation
- Testing
- Deployment
- Over which a software system is developed

13 Requirements

14 Requirements

15 Requirements

16 Requirements

17 Requirements

Requirements

```

graph LR
    User[User needs/wants] --> Requirements[Requirements]
    Requirements --> BDM[Business and domain model]
    Requirements --> FNR[Functional and non-functional Requirements]
    
```

A functional requirement is one on the **functionality** of the system, e.g., "the system must book movie ticket if the user has paid for it".

A non-functional requirement might be on the **performance** of the system, e.g., "the system must book the ticket within 30 seconds".

15

Slide 15 of 65

AutoSave File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

11 Requirements

12 Software development life cycle (SDLC)

13 Requirements

14 Requirements

15 Requirements

16 Requirements

17 Requirements

Requirements

- Requirements should be captured in the language of the user.
 - Use cases help distil the essence of requirements as sets of action-response transactions between the user and the system (e.g., as user cases).
- Example:
 - Action <Put \$1 -> Press Button>, Response <Dispense Coffee>
 - Action <Press Button>, Response <Nothing>

Who is the user here? The customer.

Slide 16 of 65

AutoSave File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

13 Requirements

14 Requirements

15 Requirements

16 Requirements

17 Analysis

Analysis

- A key theme of the analysis workflow is to understand how and where requirements interact and what it means for the system.
- In an ideal world:
 - Action <Press "a">, Response <Do "this">
 - Action <Press "b">, Response <Do "that">
- In real world:
 - Action <Put \$1 -> Press "tea">, Response <Dispense "tea">
 - Action <Press [any]>, Response <Display "No stock" if tea is finished>

Talk about the interaction where you put coin first in one case and in the other case, you just want to know whether things are available. Because when you talk with the customer, he himself does not know everything that she wants. A key job of the developer is to help customer find out what she wants.

Slide 17 of 65

File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

13 Requirements

14 Requirements

15 Requirements

16 Requirements

17 Analysis

18 Analysis

19 Analysis

20 Analysis

Analysis

- Analysis also involves
 - Detecting and removing ambiguities and inconsistencies amongst requirements
 - Requirement 1: "the system must book movie ticket if the user has paid for it"
 - Requirement 2: "the system must book tickets within 30 seconds"
 - Developing an internal view of the system
 - Identifying the analysis classes and their collaborations
 - Analysis classes are preliminary placeholders of functionality

Say it is both incomplete and inconsistent. Incompleteness: show by what if user book 100000 tickets at the same time, what happens to the money? Inconsistency: show by it takes more time than 30 seconds to print say 50 tickets.

Slide 18 of 65

File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

15 Requirements

16 Requirements

17 Analysis

18 Analysis

19 Analysis

20 Design

Design

- Deciding on the collaboration between components lies at the heart of software design.
 - A component fulfils its own responsibility through the code it contains.
 - A component exchanges information by calling methods on other components, or when other components call its own methods.

Give an example in the context of programming. Two functions exchanging information.

Slide 20 of 65

AutoSave File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do wktslides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

17 ANALYSIS

- Analyse the needs of the system
- Identify requirements and constraints
- Define the system boundaries
- Identify stakeholders
- Define the system's behavior
- Identify system requirements
- Identify system constraints
- Define the system's interface

18 ANALYSIS

- Define architecture
- Define system components and their interactions
- Define system requirements and constraints
- Define system interfaces
- Define system behavior

19 ANALYSIS

- Define architecture
- Define system components and their interactions
- Define system requirements and constraints
- Define system interfaces
- Define system behavior

20 DESIGN

- Designing the collaboration between system components
- Designing the system's interface through which it interacts with other systems
- Designing the system's internal components to implement its functional requirements

21 DESIGN

- The design activities include:
 - Designing the system's interface
 - Designing the system's internal components
 - Designing the system's behavior

22 Implementation

- Change of implementation is often required to meet requirements
- Implementing the system's components
- Implementing system requirements
- Implementing system constraints
- Implementing system interfaces
- Implementing system behavior

23 Implementation

- Change of implementation is often required to meet requirements
- Implementing the system's components
- Implementing system requirements
- Implementing system constraints
- Implementing system interfaces
- Implementing system behavior

24 Explain unit testing and integration testing briefly. Say that we will learn more about it during the software testing lectures.

Slide 22 of 65 84%

What are all the interaction scenarios. Implementation is programming on your project.

Testing will be taught different levels. Give a lot emphasis in this course.

AutoSave File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do wktslides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

19 ANALYSIS

- Define architecture
- Define system components and their interactions
- Define system requirements and constraints
- Define system interfaces
- Define system behavior

20 DESIGN

- Designing the collaboration between system components
- Designing the system's interface through which it interacts with other systems
- Designing the system's internal components to implement its functional requirements

21 DESIGN

- The design activities include:
 - Designing the system's interface
 - Designing the system's internal components
 - Designing the system's behavior

22 Implementation

- Change of implementation is often required to meet requirements
- Implementing the system's components
- Implementing system requirements
- Implementing system constraints
- Implementing system interfaces
- Implementing system behavior

23 Implementation

- Change of implementation is often required to meet requirements
- Implementing the system's components
- Implementing system requirements
- Implementing system constraints
- Implementing system interfaces
- Implementing system behavior

24 Testing

- The primary activities of the test workflow include
 - Creating test cases (manual or automatic),
 - Running test procedures, and analysing test results.
- Due to its very nature, testing is never complete.

Say that we will learn something about automatic test generation strategies later in the course. Highlight the completeness issue of testing, emphasize well.

Slide 24 of 65 84%

File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

23 Implementation

24 Testing

25 Test

26 Waterfall model

27 Waterfall model

28 Waterfall Model Problems

29 Waterfall Model

30 Cultural Exercise (10 min)

31 Rapid Prototyping

32 Rapid Prototyping

33

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

Waterfall model

Requirements → Analysis → Design → Implementation → Test → Roll-out

Nature of deliverable

Working software

Time

Software Engineering: Concepts and Applications by Subhajit Datta (Oxford University Press, 2010)

26

Explain that we have discussed different stages of software development in isolation, but we have not discussed how these stages interact among them. Now we will describe a few standard models that capture the relationship between different models.

When there are not many risks involved. Waterfall model does not work for cyberdata.

Meant for limited risks and requirements. You do all the features, and once you have the prototype that you are happy with, then you do the final prototype that fulfills the requirements.

File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

27 Waterfall model

28 Waterfall Model Problems

29 Waterfall Model

30 Cultural Exercise (10 min)

31 Rapid Prototyping

32 Rapid Prototyping

33

wk1slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

Rapid Prototyping

Requirements → Proceed to development

Clarify

Prototype

Throw away

User

Developers

Software Engineering: Concepts and Applications by Subhajit Datta (Oxford University Press, 2010)

31

Prototype gives an initial idea to the user how the product will look like. It has to be built fast in order to fix any ambiguities in the requirement. This way rapid prototype can avoid errors and ambiguities that appear much later in the software development process and allow opportunities to fix them quickly. Prototype should only include core functionalities, other extra functionality or non-functional requirement such as security and performance can be ignored.

File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

Enable Editing

27 Waterfall Model

28 Waterfall Model Problems

29 Waterfall Model

30 Current Exercise (100 min)

31 Rapid Prototyping

32 Rapid Prototyping

33 Software Development Life Cycle

Slide 32 of 65

Rapid Prototyping

- Rapid prototyping recommends the building of prototypes to clarify requirements and system scope.
- The prototypes, however, should never become the final system.
- Once the requirements have been sorted out, the system is built formally
- Timing is crucial for the prototype*

32

File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

Enable Editing

29 Waterfall Model

30 Current Exercise (100 min)

31 Rapid Prototyping

32 Rapid Prototyping

33 Software Development Life Cycle

34 Requirements and System Specification

Slide 33 of 65

Software Development Life Cycle versus Software Life Cycle

Activities

SDLC

SLC

Planning | Introduction | Useful life | Phase out | Obsolescence

Time

Analysis → Design → Implementation → Test → Roll-out → Analysis

Software Engineering: Concepts and Applications by Subhajit Datta (Oxford University Press, 2010)

Requirement changes over time, and subsequently design and analysis. For instance, customer wants new features. This will continue and propagate through all phases of the software development life cycle. Planning and introduction of the software, the time software is used, the time it is gradually stopped in phase out and finally, when the software is no longer used.

33

Cohort 2

Tuesday, January 23, 2018 11:34 AM

We talked about the different software development methodologies. Waterfall is a sequential order, and we saw that there were many problems there, because there could not be one-way dependency, need to revise as you go through the software development. One way is to use rapid prototyping, there is no sequential order of design implementation and testing, but make some prototypes to talk to the client. You go through with a few cycles and then you try with the software.

Iterative and Incremental Development. As you see this RADIT, these different states of software development. Let's say your project goes for 12 months, and timebox your project like two months, two months. You have this time box manner, and then you promise something and complete this set of features, and then you implement and put more features in the software. Similarly, you do some more features, and come up with an even greater software. It is very much like rapid prototyping, here you are not really making prototypes.

Rapid prototyping, you make something quick, you throw out the software and then make another. Here you are making the new software, then you put some more features, and then you have another release. So this is one way you follow your course project, because you can promise something.

You make different implements of the software, different versions, and you have different features being implemented.

Agile Manifesto: Agile is something which is not a software development methodology. A belief that people follow when they develop something. There are some diagrams, tools that people follow. Some tools that people follow when they use this software. Something which is more important and encouraged to follow. You use individuals and interactions. Your peers and your instructors and even TA. Working software over Comprehensive documentation: you show the clients what you have already done. This is a way to better communicate. It's the belief in the agile manifesto. You show that you communicate with the client. When you have something working, client feels better. Customer collaboration over Contract negotiation: you do not really collaborate with the customer with the project. You need to talk to clarify what you want in the project.

Responding to change over following a plan. Some people are not acceptable to making changes. You are trying to find out, that you have this additional feature as well. It's sometimes very common to be defensive about it. I'm not going to implement if he did not talk about it. Now that he or she has some working software, you might see that some additional features may be useful for us. When you design software, collaborate with the client, maybe he wants some features differently, the agile manifesto says that you should collaborate with it. Within the process, you could put a different software methodology.

Customer satisfaction by rapid delivery of useful software: We ask you to show something from the project. To convince you are making progress: instead of showing documentation, or some report, it's good to provide some software. Welcome changing requirements. Let's say somebody tell you it want to be different; change very late, you should be welcoming it.

Working software is delivered frequently: usually working software is delivered infrequently, in one year. Agile has to be done in a few weeks. Working software is the principal measure of progress. When you have the project meeting, working software would make your grades better. Try to make this process so that it is better for us. Is this agile thing clear? It's not a development methodology. We can follow agile things simple.

Agile is mostly about timeboxing. It is possible it might be this time but different. Say your software has 10 different features. Let's say you have 2 features first, then 4 then 4 last features. Agile says that you should do in timeboxes. Instead of completing certain sets, you will try to implement two features. Possible to implement something more. We have provided some requirement for every project meeting. But it doesn't stop you from going beyond. Meeting every fixed set of time rather than fixed set of principles.

There are things on which your project depends. Some empirical data.

$(B^{1/3} * Size) / Productivity = Effort^{1/3} * time^{4/3}$

B = special skill factor

Productivity = a productivity parameter (depends on team)

Effort = Human effort in years or months

Time = time to complete project in years or months.

Given that three months time, how much effort you intend to put.

Effort is the number of willing hours to put into your project. If you reduce the time, the time needed to reduce the project reduced by 1 but increased by a factor of 4.

What is the implication? Any schedule slip, say if your group does not work for a few amount of time, it increases the effort a lot more than these few impact hours. Say you did not work for two days, It does not mean 8 days effort would be required. B is some constant. Now what I want to show you it's that it is a power of four.

B is related to the size of the product. More code you develop, more skill you need.
 $5-15K = 0.16$

Productivity is the type of software being developed.

Introduction to Software Design

Monday, January 29, 2018 11:30 AM

week2slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

9 10 11 12 * 13 14 Software Design is Hard (cont'd)

The Rational Model

Input: Goal, Desiderata, Utility function, Constraints, Design tree of decisions

Until ("good enough") or (time runs out) {
Do another design (to improve utility function)
Until design is complete
Backtrack up design tree
Explore a path not searched before
While design remains feasible,
make another design decision
EndWhile
EndUntil
EndDo
Take best design
EndUntil

This does not work for a complex software project.

Slide 11 of 63 English (United States)

11

week2slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

9 10 11 12 * 13 14 Software Design is Hard (cont'd)

Criticizing the Rational Model

Software Engineering: Concepts and Applications by Subhajit Datta
(Oxford University Press, 2010)

13

Number of design options may be too big. Basically, the level of understanding of your project increases with time. For any internal project, any options become better. Even if we know all the design choices, the combinations of design choices are exponential and therefore it may not be easy to find a good design. Enumeration does not work.

Strategies for addressing complexity:

Decomposition, Abstraction, and Hierarchy.

Decomposition: Divide and conquer. To solve harder problem, simplify by going to the smaller problem.

If I have a big software project, how can I decompose the software project into simpler software systems.

Break down a system into smaller and more manageable parts, so that the channel capacity of our comprehension is not breached.

week2slides - Protected View - Saved to this PC

Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

15 ADDRESSING COMPLEXITY IN DESIGN

16 Strategies for addressing complexity

- Break up complex system and reduce its size
- Abstraction
- Modularity

17 Decomposition

- Break up large system into smaller, more manageable pieces
- Top-down
- Bottom-up
- Mixed approach
- Coupling and Cohesion

18 Decomposition

19 **Is it a Good Design?**

Human mind can handle complexity only up to a certain threshold. Hence, decomposition helps in the design process. For your app searching for restaurants, let us say someone focus on the search algorithm whereas someone focus on the GUI development.

20 Decomposition

Slide 19 of 63 English (United States)

week2slides - Protected View - Saved to this PC

Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

17 Decomposition

- Break up complex system and reduce its size
- Abstraction
- Modularity

18 Decomposition

19 **Is it a Good Design?**

Human mind can handle complexity only up to a certain threshold. Hence, decomposition helps in the design process. For your app searching for restaurants, let us say someone focus on the search algorithm whereas someone focus on the GUI development.

20 **Is it a Good Design?**

Human mind can handle complexity only up to a certain threshold. Hence, decomposition helps in the design process. For your app searching for restaurants, let us say someone focus on the search algorithm whereas someone focus on the GUI development.

21 Cohort Exercise 1.1

They can develop independently, when merge together during integration then they can come together.

Dating Application

- : Find filters (for dating partner)
- : Take the addresses from the filtered results and cross examine with the results
- : Takes the user's input
- : Use the input to re-filter results
- : UI
- : Database
- : Algorithm (For recognition)
- : Calculation

Coupling is between input to find filters, addresses from filters to find relevant partners. Within the Cohesion edges, find the inputs as to determine the likeliness/probability of getting the thing.

AutoSave OK

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week2slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

21 Cohort Exercise 1.1

* Assume you are designing a dating application which displays relevant matches for a user. Can you configure the disentanglement process so that the user can see the distance and chapter to the nearest resource? If so, how will the user matches sorted by distance and chapter? If not, why not? What would you do instead or which the app will comprise what type of information about the user and their potential matches? Describe the application and let me know your thoughts.

22 Abstraction

The power of abstraction

Abstraction is at the center of much work in Computer Science. It encompasses finding the right interface for a system as well as finding an effective design for a system implementation.

23 Abstraction

* One of the most frequently used words in the context of software design

• Key idea: As we strive to repress the intricacy of a complex object, we choose to ignore its material details, thereby trading off the generalization provided above at the cost (Beck et al., 2012).

24 Abstraction

25 Abstraction

26 Abstraction

22

Abstraction

The power of abstraction

Abstraction is at the center of much work in Computer Science. It encompasses finding the right interface for a system as well as finding an effective design for a system implementation.

Slide 22 of 63 Notes Share Print Search Help Feedback 90%

week2slides - Protected View - Saved to this PC

Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

27 Abstraction

28 Abstraction

29 Abstraction

30 Abstraction

31 Abstraction

32 What's that?

Abstraction

What is a good abstraction?

Level of Abstraction

GUI

Includes everything down

Libraries, Packages, ...

System Calls

Disk /Hardware /Driver

30

week2slides - Protected View - Saved to this PC

Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

27 Abstraction

28 Abstraction

29 Abstraction

30 Abstraction

31 Abstraction

32 What's that?

Abstraction

- High-level programming language is another abstraction we use daily
- It abstracts away the low-level implementation details
 - Which processor?
 - What type of Memory?
 - What type of system (PC, supercomputer, embedded systems)?

31

week2slides - Protected View - Saved to this PC

Student - Laura Ong Jin Hua

Abstraction

The diagram shows a vertical stack of abstraction levels. At the top is a computer screen labeled "DSL (domain specific)" with the text "Includes everything down". Below it is a box labeled "JAVA" containing code: "import java.io.*"; "mov x, r2"; "add r1, r2"; "mov r1, y". Further down is a box labeled "Assembly code" with the same assembly language code. At the bottom is a computer processor labeled "ISA (Turing complete)". A stick figure on the left points to the "JAVA" box. A dashed red arrow points from the stick figure to the "JAVA" box. To the right of the stack is a vertical arrow labeled "Level of Abstraction" pointing upwards.

31 Abstraction

- high-level programming language to abstract abstraction away at code
- it abstracts away the low-level implementation details
- what type of abstraction?
- what type of source? (PC, application-specific, embedded systems?)

32 What's that?

33 Abstraction

34 Abstraction

35 Abstraction

36 Cohort Exercise 1.2

Slide 34 of 63 English (United States)

A Language is **turing complete** if it can be used to simulate any turing machine.

I can simulate any program if written in machine code.

week2slides - Protected View - Saved to this PC

Student - Laura Ong Jin Hua

Abstraction

The diagram shows a vertical stack of abstraction levels. At the top is a computer screen labeled "DSL (domain specific)" with the text "Not Turing Complete". Below it is a box labeled "Lambda Calculus" with the same assembly language code: "mov x, r2"; "add r1, r2"; "mov r1, y". Further down is a box labeled "Simply typed lambda calculus" with the same assembly language code. At the bottom is a computer processor labeled "ISA (Turing complete)". A stick figure on the left points to the "Lambda Calculus" box. A dashed red arrow points from the stick figure to the "Lambda Calculus" box. To the right of the stack is a vertical arrow labeled "Level of Abstraction" pointing upwards.

33 Abstraction

34 Abstraction

35 Abstraction

36 Cohort Exercise 1.2

37 Cohort Exercise 1.2

38 Hierarchy

Slide 35 of 63 English (United States)

Something that can be done in machine level, will not be done in the lambda calculus. The concept of

turing complete does not comply: i.e. HTML

Hierarchy: Whenever we examine something new or something that seems to be complex, we try to see

what common characteristic it shares with some other things we know, or if it is part of something we have seen earlier.

Key idea: "is a" relation: a car is a locomotion device with wheels. Class Car extends locomotion

"Part of" relation: A part of a car is the engine. Engine e is part of Class c documentation.

Use Case:

Monday

Monday, February 5, 2018 11:44 AM

week3slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

3 Use Case Diagrams

- A diagram that shows a set of use cases and actors and their relationships.
- Use case diagrams represent system functionality, the requirements of the system from the user's perspective.

4 Actors

Actors are the people or systems that interact with the system. They are entities that are outside the boundaries of the system that interact with the system. Primary actors are those that initiate events. Secondary actors are those that receive events.

5 Exercise 1

```
graph LR; User((User)) --- Call([Call]);
```

6 Exercise 2

```
graph LR; User((User)) --- Call([Call]);
```

7 Who are the Actors?

Actors

- Who/what will be interested in the system?
- Who/what will change the data in the system?
- Who/what will want to interact with the system?
- The customer. The user.
- The company managing the system?
- The telephone network operator

8 Use Case in Use Case Diagrams

Slide 5 of 67 English (United States)

5

Exercise 1

```
graph LR; User((User)) --- Call([Call]);
```

Who are the primary and secondary actors?

Primary Actor: making a call
Secondary Actor: switch operator

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

3 Use Case Diagrams
A diagram that shows a set of use cases and actors and their relationships.
Use case diagrams represent system functionality, the requirements of the system from the user's perspective.

4 Actors
Actors are the people or systems that provide or receive information from the system. They are the external entities of a system, which could be human beings, other systems, or hardware devices.
Actors that interact with the system are called primary actors. Actors that receive information from the system are called secondary actors.

5 Exercise 1

6 Exercise 2

7 Who are the Actors?
Actors
Who/what will be interested in the system?
Who/what will want to change the data in the system?
Who/what will want to interface with the system?
Who/what will want information from the system?
The switch
The telephone network operators

8 Use Case in Use Case Diagrams

Slide 6 of 67 English (United States)

Actor: Switch (subsystem of telephone network). Secondary Actor is the subscriber, whoever is trying to make the call.

Exercise 2: Use Case Diagram

```

graph LR
    User((User)) --- SendDialtone([Send Dialtone])
  
```

Who are the primary and secondary actors?

6

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

3 Use Case Diagrams
A diagram that shows a set of use cases and actors and their relationships.
Use case diagrams represent system functionality, the requirements of the system from the user's perspective.

4 Actors
Actors are the people or systems that provide or receive information from the system. They are the external entities of a system, which could be human beings, other systems, or hardware devices.
Actors that interact with the system are called primary actors. Actors that receive information from the system are called secondary actors.

5 Exercise 1

6 Exercise 2

7 Who are the Actors?
Actors
Who/what will be interested in the system?
Who/what will want to change the data in the system?
Who/what will want to interface with the system?
Who/what will want information from the system?
The switch
The telephone, The switch
The telephone network operators

8 Use Case in Use Case Diagrams

Slide 7 of 67 English (United States)

Who are the Actors?

Actors

- Who/what will be interested in the system?
 - Any human interested in making telephone calls
- Who/what will want to change the data in the system?
 - The switch
- Who/what will want to interface with the system?
 - The telephones, The switch
- Who/what will want information from the system?
 - The telephone network operators

7

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

7 Who are the Actors?
Actors
• Which actor will be interested in the system?
• Who does the system interact with?
• Who does the system want to interact with?
• The teacher
• Students want to interact with the system?
• The students
• Who interacts with whom from the system?
• The teacher interacts with students

8 Use Case in Use Case Diagrams
Use Case
• Is a description of a set of sequences of actions, including variants, that system performs that yields an observable value to an actor.
• Should ideally begin with a verb.

9 Notations: between Use Cases
include
• Specifies that the source use case explicitly incorporates the behaviors of another use case at a location specified by the target use case.
• Specifies that the target use case extends the behaviors of the source.

10 Use Case Diagram: Example

11 Use Case Diagram: Example

12 Granularity of Use Cases

Slide 8 of 67 English (United States)

8

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

7 Who are the Actors?
Actors
• Which actor will be interested in the system?
• Who does the system interact with?
• Who does the system want to interact with?
• The teacher
• Students want to interact with the system?
• The students
• Who interacts with whom from the system?
• The teacher interacts with students

8 Use Case in Use Case Diagrams
Use Case
• Is a description of a set of sequences of actions, including variants, that system performs that yields an observable value to an actor.
• Should ideally begin with a verb.

9 Notations: between Use Cases
include
• Specifies that the source use case explicitly incorporates the behaviors of another use case at a location specified by the target use case.
• Specifies that the target use case extends the behaviors of the source.

10 Use Case Diagram: Example

11 Use Case Diagram: Example

12 Granularity of Use Cases

Slide 9 of 67 English (United States)

9

Extend: same functionality generally, with some additional functions. I.E. Have a enrollment course use case, and extends the email exchange student enrollment
Includes: library .i.e. error inclusion

Tuesday

Tuesday, February 6, 2018 11:40 AM

Misuse cases: Not a good idea when you design a system to test only for the use cases. Must consider ways system can be abused. Quite related to the security or any kind of non functional entity in general. If you are talking about the telephone network, only consider what you are supposed to do. There are ways the system can be abused. There might be some loop hole which someone can abuse. Hacking and network is one. Non functional requirements count as part of the misuse cases.

Client will not consider misuse cases. Just because the client doesn't give it to you does not mean it doesn't want to give to you. Give an example how complicated it can become even for a simple use case diagram. Actors can be the customers. Hackers could also be actors. Or the system; actor does not need to be a human.

AutoSave week3slides - Protected View < Saved to this PC Student - Laura Ong Jin Hua

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

16 Misuse cases: Example
• Consider an e-commerce website where an attacker can purchase different items.
• The actions can be:
- a Customer
- a System

17 Use Case Diagram: Online Purchase

18 Misuse Case : Online Purchase

19 Misuse Case : Online Purchase

20 Misuse Case : Online Purchase

21 Misuse Case : Online Purchase

Misuse Case : Online Purchase

The diagram illustrates a misuse case relationship between a User or System actor and a Customer actor. The User or System actor is connected to a Purchase Item use case, which in turn connects to a Deny Purchase component. The User or System actor is also connected to a Security Control use case, which connects to a Digital Signature component. The Deny Purchase component is labeled 'Threatens' and 'Prevent', while the Security Control component is labeled 'Includes'.

19

The <<precedes>> paradigm captures which use-case is a pre-condition of another use-case.

Slide 19 of 67 English (United States)

If you have a digital signature, can prevent abuse of the system. Here it includes (security control includes functionality of digital signature) New thing here is prevent. Prevent is some component in your system. Designation is for misuse cases.

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

20 Misuse Case : Online Purchase

21 Misuse Case : Online Purchase

22 Misuse Case : Online Purchase

23 Misuse Case : Online Purchase

24 Misuse Case Diagram: Example

25 Misuse Case Diagram: Example

Misuse Case : Online Purchase

```

graph TD
    UC[User or System] -- Threatens --> PI[Purchase Item]
    UC -- Threatens --> DS[Digital Signature]
    UC -- Threatens --> SU[Steal User Info]
    UC -- Threatens --> AU[Authentication]
    UC -- Threatens --> CU[Change User Info]
    UC -- Includes --> SC[Security Control]
    SC -- Includes --> DS
    SC -- Includes --> ER[Encrypt Request]
    SC -- Includes --> AU
    SC -- Prevent --> DP[Deny Purchase]
    SC -- Prevent --> SU
    SC -- Prevent --> CU
    DS -- Prevent --> DP
    DS -- Prevent --> SU
    ER -- Prevent --> DP
    ER -- Prevent --> SU
    AU -- Prevent --> DP
    AU -- Prevent --> SU
    AU -- Threatens --> CU
    CU -- Threatens --> SU
    
```

The diagram illustrates a misuse case for an online purchase system. It features three main actors: 'Customer' (represented by a stick figure), 'User or System' (represented by a stick figure with a tail), and 'Hackers' (represented by a stick figure with a red border). The 'User or System' actor is associated with several misuse cases: 'Purchase Item', 'Digital Signature', 'Steal User Info', 'Encrypt Request', 'Authentication', and 'Change User Info'. Relationships between these misuse cases include 'Threatens' (from User or System to Purchase Item, Digital Signature, Steal User Info, and Change User Info), 'Includes' (from User or System to Security Control, which then includes Digital Signature, Encrypt Request, and Authentication), and 'Prevent' (from Security Control to Deny Purchase, which then prevents Purchase Item, Digital Signature, and Steal User Info; and from Security Control to Change User Info, which then threatens Steal User Info).

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

20 Misuse Case : Online Purchase

21 Misuse Case : Online Purchase

22 Misuse Case : Online Purchase

23 Misuse Case : Online Purchase

24 Misuse Case Diagram: Example

25 Misuse Case Diagram: Example

Misuse Case Diagram: Example

```

graph TD
    UC[User or System] -- Threatens --> PB[Purchase Bitcoin]
    UC -- Threatens --> DSA[Denial of Service Attack]
    UC -- Threatens --> BPPA[Brute force Password Attack]
    UC -- Threatens --> ESSV[Exploit System vulnerability]
    UC -- Threatens --> SMDU[Send malicious update]
    
```

The diagram illustrates a misuse case for purchasing Bitcoin. It features two main actors: 'User or System' (represented by a stick figure) and 'Hackers' (represented by a stick figure with a red border). The 'User or System' actor is associated with four misuse cases: 'Purchase Bitcoin', 'Denial of Service Attack', 'Brute force Password Attack', 'Exploit System vulnerability', and 'Send malicious update'. All relationships are labeled 'Threatens', indicating that the User or System actor is threatening the other misuse cases.

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

22 Misuse Case : Online Purchase

23 Misuse Case : Online Purchase

24 Misuse Case Diagram: Example

25 Misuse Case Diagram: example

26 Cohort Exercise 5 (10 minutes)

27 Cohort Exercise 6

Misuse Case Diagram: Example

```

graph LR
    Hackers[Hackers] --> PurchaseBitcoin[Purchase Bitcoin]
    Hackers --> MonitorNetwork[Monitor network]
    Hackers --> SystemPatch[System Patch]
    Hackers --> ExploitVulnerability[Exploit System vulnerability]
    Hackers --> MonitorLoginAttempts[Monitor login attempts]
    Hackers --> SendMaliciousUpdate[Send malicious update]
    Hackers --> DenialOfService[Denial of Service Attack]
    Hackers --> BruteForce[Brute force Password Attack]
    PurchaseBitcoin --> MonitorNetwork
    PurchaseBitcoin --> SystemPatch
    PurchaseBitcoin --> ExploitVulnerability
    PurchaseBitcoin --> MonitorLoginAttempts
    PurchaseBitcoin --> DenialOfService
    PurchaseBitcoin --> BruteForce
    MonitorNetwork --> SystemPatch
    MonitorNetwork --> ExploitVulnerability
    MonitorNetwork --> MonitorLoginAttempts
    MonitorNetwork --> DenialOfService
    MonitorNetwork --> BruteForce
    SystemPatch --> ExploitVulnerability
    SystemPatch --> MonitorLoginAttempts
    SystemPatch --> DenialOfService
    SystemPatch --> BruteForce
    ExploitVulnerability --> MonitorLoginAttempts
    ExploitVulnerability --> DenialOfService
    ExploitVulnerability --> BruteForce
    MonitorLoginAttempts --> DenialOfService
    MonitorLoginAttempts --> BruteForce
    DenialOfService
    BruteForce
  
```

The <<precedes>> paradigm captures which use-case is a pre-condition of another use-case.

You do need to model the cases where the system can be misused. Basically, you model in the same diagram. So it becomes more coherent.

newProj - EA

File Edit View Project Diagram Element Tools Add-Ins Settings Window Help

Tool... More tools... Use Case Actor Use Case Collabora Boundar Package Use Case ... Common

Project Browser Model Static Static Static Static Back up Switch Local Switch Maintenance Malfunctioning Switch Remote Switch Subscriber Suscriber Ask for new courses for Call Connect back to itself Determine Remote Swi Dial Invalid Number Dial Number Dial Tone Message Error Message Hanging Up Hear a Ringtone Lifts Phone cradle Make outgoing calls No Error Message

Properties General Settings Name Scope Type Stereotype Alias Complexity Version Phase Language

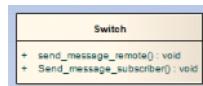
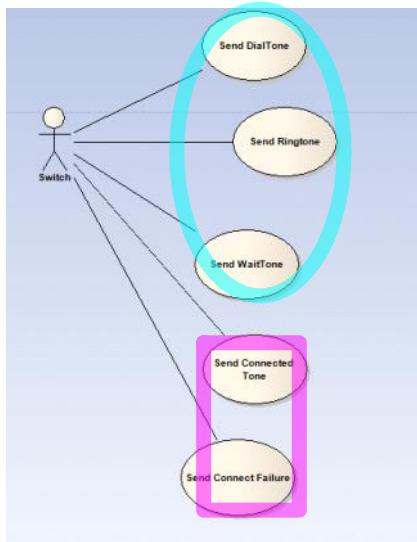
Diagram showing a Use Case Diagram with various actors (Subscriber, Maintenance, Malfunctioning Switch) and use cases (Reassign new switch, Pickup, Connect back to itself, Repair switch, Ask for new courses for repair, Wired breakdown, No Error Message, Hear a Ringtone, Call, Dial Number, Dial Tone Message, Error Message, Hanging Up, Hear a Ringtone, Lifts Phone cradle, Make outgoing calls, No Error Message). Relationships between use cases include <<precedes>>, <<invokes>>, and <<includes>>.

Class diagrams: What are the components of your system. You may not have come across a Class Diagram. It is when you try to construct something object oriented. It shows you the different classes of your software and the interactions between them.

Interfaces: dotted line. There should be a class that implements that interface.

Every use case for the switch has to be called by some operation. Every operation in the class has to correspond to some use case. Which means you have something redundant in your class. If you have some use case which is nothing in the class; you are not implementing system correctly. But it doesn't have to be one to one. At the same time, every function in the class has to correspond to some use case. So everything becomes connected. Let's assume I am abstracting it away.

You see that these three are actually sending messages to subscriber. And these two are sending message to remote



For every use case, I have a function in the class which is implementing the use case. I do not have any function in the class which is not in the use case.

Associations: When you model a system, certain classes will be related to each other.

Bi-Directional(standard) association: both classes are aware of each other and their relationship

I.E. plane and flight needs to know the existence of each other. Both classes are aware of each other. Flight knows which plane it has been assigned to. Plane knows which flight has to be known.

Then comes the attributes: number of the plane e.g.

1. Whether uni-directional or bi-directional
2. What is the number of objects assigned to

I.E. It's more important to understand how many planes to one flight path.

0..1: Zero or 1

Constant: One only

If I assign a plane to three different flights. So you could have 3 on the plane end of the association line.

Association could be both uni and bi. Uni: two classes are related, but only one class knows that the relationship exist.

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

Multiplicity

Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
*	Zero or more
1..*	One or more
3	Three only
0..5	Zero to Five
5..15	Five to Fifteen

52

53

54

Draw the association here again on the board and put multiple options. Ask students about which option mean what?

Slide 52 of 67 English (United States)

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

49 Inheritance

50 Interfaces

51 Associations

- When you model a system, certain classes will be related to each other.
- Bi-directional (standard) association: both ends have arrows on either side of the relationship.

52 Multiplicity

Multiplicity	Description
0..1	One or zero instances.
1..1	One instance.
1..n	One or more instances.
n..n	Any number of instances.
n..1	Any number of instances, but one at the other end.
0..*	Zero or more instances.
1..*	One or more instances.
0..n	Zero or more instances.

53 Uni-Directional Association

- In a uni-directional association, two classes are related, but only one class knows that the relationship exists.

54 Cohort Exercise 11 (10 minutes)

Why do you think it is unidirectional? Can overdrawnbankaccount report be part of an account?

53

```

classDiagram
    class OverdrawnAccountsReport {
        generatedOn : Date
        refresh()
    }
    class BankAccount {
        owner : String
        balance : Dollars
        deposit(amount:Dollars)
        withdrawal(amount:Dollars)
    }
    OverdrawnAccountsReport "0..*" --> "1" BankAccount : overdrawnAccounts
  
```

Slide 53 of 67 English (United States) 97%

Overdrawn bank account knows what he knows, but bank account knows how many there are.

newProj - EA

File Edit View Project Diagram Element Tools Add-Ins Settings Window Help

Tool... More tools... Class Package Class Interface Enumera Table Signal Associati Class Rel... Common

Logical Diagram: "Static" created: 2/5/2018 12:29:16 PM modified: 2/6/2018 12:47:59 PM 100% 795x1138

Project Browser Model Static

Properties

Switch

- + send_message_remote(): void
- + Send_message_subscriber(): void

Subscriber

- # name: char
- + number: int = 12345678
- + accessible(boolean): boolean
- + make_call(boolean, long): boolean

Call

- + Dial(): void
- + make_call_accessible(boolean): void

(know both numbers, make_call = true)

Project Browser Resources

Properties Notes

Left: 499 x Top: 400 - Width: 183 x Height: 70 CAP NUM SCR

Class: Call

Start Page *Static *Static Static

week3slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua

Lifelines

lifeline notation elements are placed across the top of the diagram. Lifelines represent either roles or object instances that participate in the sequence being modeled.

S1: Subscriber

An object named S1 of type Subscriber.

In practice: if it is synchronous you cannot do anything until the return has been acknowledged.
However, for dial numbers, it is asynchronous and you can do stuff to it.

29

AutoSave File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

28 Sequence Diagram
A sequence diagram is an interaction diagram that highlights the order in which the messages are exchanged.
Sequence diagram complements a use case with details on the workflow of events.

29 Lifelines
Lifeline notation elements are defined by the lifelines of the diagram. Lifelines represent either roles or objects that interact and participate in the sequence being modeled.

30 Messages
To show an object (i.e., lifeline) sending a message to another object, you draw an arrow to the receiving object.

31 Arrows and Orders
• A solid arrowhead is a synchronous call
• A stick arrowhead is an asynchronous signal
• The order of the messages is defined two rules:
– On the same lifeline, a higher message precedes a lower message
– Message sending precedes message receiving

32 Synch and Async Messages
Sync and Async Messages

33 Cohort Exercise 7 (10 minutes)

Messages

To show an object (i.e., lifeline) sending a message to another object, you draw an arrow to the receiving object.

```

sequenceDiagram
    participant S1 as S1: Subscriber
    participant T1 as T1: Switch
    participant S2 as S2: Subscriber
    S1->>T1: Lift_Receiver()
    activate T1
    T1-->>S1: Send_Dialtone()
    deactivate T1
    S1->>T1: Dial_Number(N1)
    activate T1
    T1-->>S2: Check_Number(N1)
    deactivate T1
    T1-->>S2: Valid_number()
    deactivate T1
    
```

Optional Messages are put dotted.

30

AutoSave File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week3slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

28 Sequence Diagram
A sequence diagram is an interaction diagram that highlights the order in which the messages are exchanged.
Sequence diagram complements a use case with details on the workflow of events.

29 Lifelines
Lifeline notation elements are defined by the lifelines of the diagram. Lifelines represent either roles or objects that interact and participate in the sequence being modeled.

30 Messages
To show an object (i.e., lifeline) sending a message to another object, you draw an arrow to the receiving object.

31 Arrows and Orders
• A solid arrowhead is a synchronous call
• A stick arrowhead is an asynchronous signal
• The order of the messages is defined two rules:
– On the same lifeline, a higher message precedes a lower message
– Message sending precedes message receiving

32 Synch and Async Messages
Sync and Async Messages

33 Cohort Exercise 7 (10 minutes)

Arrows and Orders

- A solid arrowhead if a synchronous call operation
- A stick arrowhead if an asynchronous signal
- The order of the messages is defined two rules:
 - On the same lifeline, a higher message precedes a lower message
 - Message sending precedes message receiving

```

sequenceDiagram
    participant instance1 as instance1 : Object1
    participant instance2 as instance2 : Object2
    instance1->>instance2: synchronousMessage()
    activate instance2
    instance2-->>instance1: asynchronousMessage()
    deactivate instance2
    
```

31

This diagram shows that the subscriber is connected to the switch. So you should be aware of the switch that you are connected to. If you have drawn the class diagram where it does not show any association between subscriber to switch, then you cannot have this kind of message. But actually, if you have bidirectional, then it is fine.

This association shows that your subscriber can only talk to one switch at a time. One subscriber is

sending message to only one switch. It is not sending to multiple switches. If this subscriber was also sending messages to another switch, this diagram becomes inconsistent. You need to be careful about association. Subscriber can only send to at least one switch.

When you draw for big project, this is something to be careful about.

You draw state diagrams for each of your class. You draw state diagrams for each of these classes.

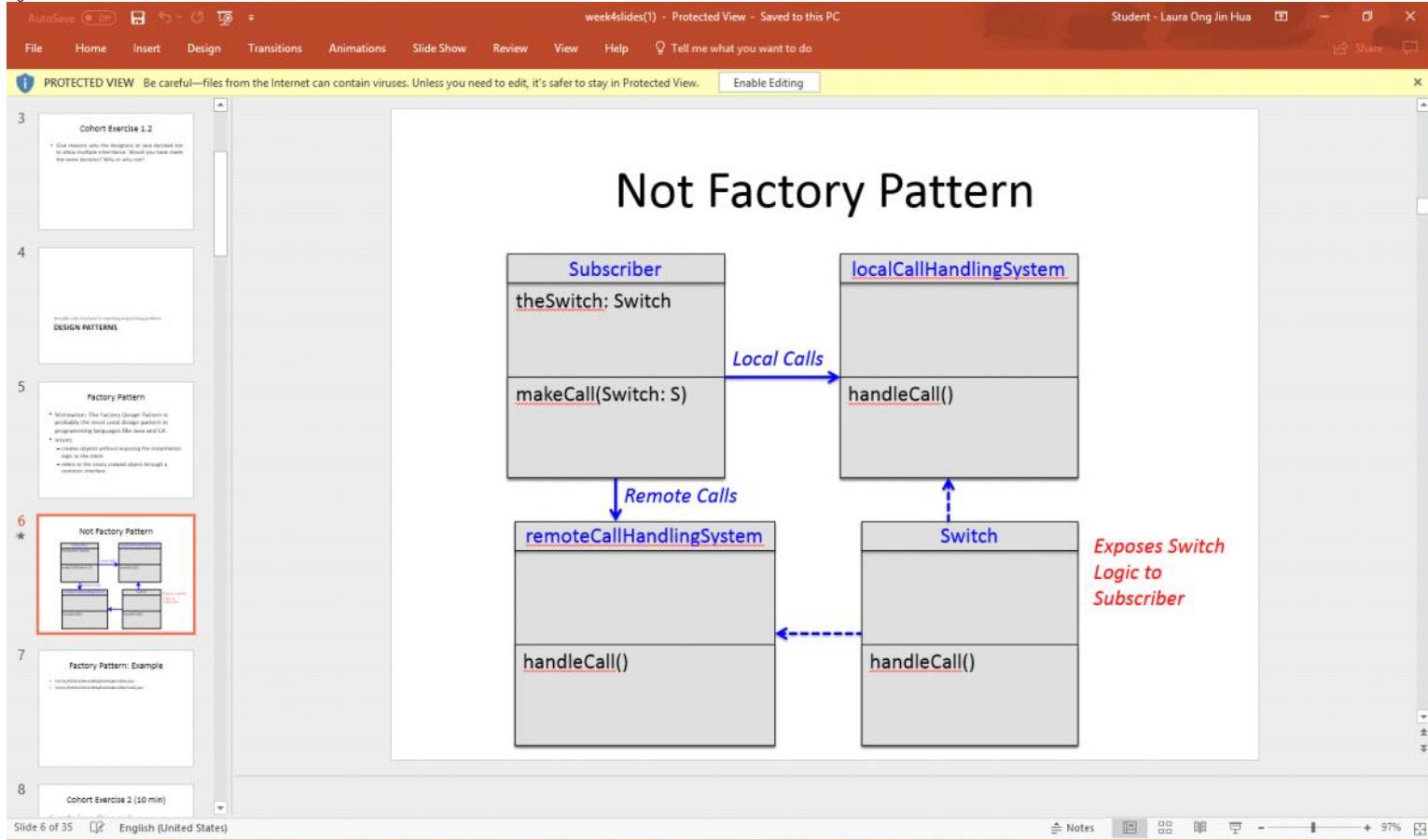
In state diagrams you have states and transitions.

You have initial states and final states like this.

Monday

Monday, February 12, 2018 11:35 AM

Not a Factory Pattern: Why is this Switch not? The subscriber should not be aware of implementation logic of a switch. All it should want to do is make contact with a call.



In Factory Pattern, I just create a switch factory, which will have a function called make switch for this call, for this call option (either remoteCall or local call). You don't instantiate the switch directly. Subscriber should have no idea of the different logic for local or remote. In its own logic, it will create the instantiation for the switch.

```

class SwitchFactory {
    // sudiptac: Encapsulation of all different subsystems in the switch
    public Switch makeSwitchForThisCall(String callOption){
        Switch newSwitch = null;

        if (callOption.equals("L")){
            return new localCallHandlingSystem();
        } else

        if (callOption.equals("R")){
            return new remoteCallHandlingSystem();
        } else
            return null;
    }
}
  
```

You have a switch factory (takes input from subscriber) This code was directly in the subscriber. Put it into a switch factory and subscriber only aware of factory class. Can handle both remote and local calls. Make the implementation transerent. There is an abstract class for the switch, and two different class which will be deriving from the switch class. One for local and one for remote.

The screenshot shows a Microsoft Word document with a presentation slide titled "Writing Tests for JUnit". The slide content includes:

Writing Tests for JUnit

- Need to use the methods of the [junit.framework.assert](#) class
 - javadoc gives a complete description of its capabilities
- Each test method checks a condition ([assertion](#)) and reports to the test runner whether the test failed or succeeded
 - [assertTrue \(boolean\)](#)
 - [assertTrue \(String, boolean\)](#)
 - [assertEquals \(Object, Object\)](#)
 - [assertNull \(Object\)](#)
 - [Fail \(String\) //fail a test with \(possibly\) a given message](#)
- All of the methods return void

On the left side of the Word window, there are several slide thumbnails labeled 17 through 22. Slides 17, 18, 19, 20, and 21 contain text and diagrams related to JUnit and test writing. Slide 22 contains the question "Why do we need fail() method?". The status bar at the bottom right indicates "Slide 20 of 85" and "English (United States)".

Monday

Monday, February 26, 2018 11:41 AM

$d \geq e$

true	false
Throws	SOPL
Error	$e-d > f$

True	False
SOPL("once") Middle = $(e+d)/2$	SOPL("return") Return $(e+d)/2$
Middle < true	

True	False
System... D = middle	System... E = middle

Loop Tail back to $e-d > f$

This is called loop baggage

Control Flow Graph: Captures how your program behaves at run time. After this block of code is executed, when you enter the control program for your code. Writing tests for coverage becomes much easier. So the point is that you must understand how the control graph is drawn from an arbitrary piece of code.

For each atomic condition; do not confuse with conditional statement. But where $x == 0$, for every test to be equilibrated to true and false. Must have $x == 0$ or $x != 0$ and $y < 0$ and $y > 0$.

The screenshot shows a Microsoft PowerPoint slide titled "Every Condition". The main content area contains the following bullet points:

- For each condition, there must be one test case which satisfies it and one which dissatisfies it.
- Question: how many test cases we need?
 - if (A && B) ...
 - if ((j>=0) && salary[j] > 10000) ...

The sidebar on the left lists several slides with titles such as "Every Path", "Exercise 10", "Exercise 11", and "White Box tests". The status bar at the bottom indicates "Slide 70 of 82" and "English (United States)".

For $a = t$ and $b = f$, $a = f$ and $b = t$. You have condition coverage and no branch coverage. You only execute the false portion of the branch.

Most of the time you cannot achieve it : Path coverage.

A path is defined as a sequence of executed nodes in the control flow graph between the entry node of the graph and the exit node. It is a path which is representing the sequence of executed nodes in run time. Where the program actually enters and the exit node. So you can have multiple exit nodes. It is defined as a sequence of nodes between entry or exit. Since program has loops and branches, there are many possible paths from entry to exit node.

week5slides - Saved to this PC Student - Laura Ong Jin Hua

Exercise 9

- * write a set of tests to cover each branch of the manipulate function. How many tests do you need? Is it the minimum number of tests to cover all the branches?

Every Path

A path is defined as a sequence of nodes in the control flow graph between the entry node of the graph and the exit node.

cond1->cond2->stmt1->loop tail->stmt3 is a path

cond1->cond2->stmt1->loop tail->cond1->cond2->stmt2->loop tail->stmt3 is also a path

cond1->cond2->stmt1->stmt2->loop tail->stmt3 is **not** a path

How many paths in total?

```

graph TD
    cond1[cond1] --> cond2[cond2]
    cond2 --> stmt1[stmt1]
    cond2 --> stmt2[stmt2]
    stmt1 --> loopTail[Loop Tail]
    stmt2 --> loopTail
    loopTail --> cond1
    loopTail --> stmt3[stmt3]
  
```

Click to add notes

Slide 75 of 85 English (United States)

Let's look at what I mean by path.

There are two different paths. You execute the loop twice.

Assume the loop is run through hundred times. You don't need to consider if one time or two times. So it is 2^100 .

AutoSave

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week5slides - Saved to this PC Student - Laura Ong Jin Hua

Cut Copy Paste Format Painter New Slide Section Layout Reset Font Paragraph Drawing Editing

Slides

74 Exercise 9

- Write a set of tests to cover each branch of the manipulate() function. How many tests did you write? What is the minimum number of tests to cover all the branches?

75 Every Path

- Again, a different sequence of tests in the control flow path between the entry node of the graph and the exit node.

76 Every Path

77 Exercise 10

- How many paths are there in the (manipulate) function? Explain your answer.

78 Every Condition

- For each condition, there must be one test case which validates it and one which does not.
- (Question: how many test cases we need?
 - $\approx 0.85N$
 - $\approx 0.85(1.44 \times 10^{10}) \approx 12,000,000$

Every Path

Remember some paths may be infeasible

How many paths in total?

Infeasible

Click to add notes

Not every path may be executed. Even for a realistic program you may find many paths that may not be executed.

Thursday

Thursday, March 1, 2018 10:41 AM

These are not valid input, but you might generate input that are invalid and there is no point in testing them because that is a parser and they are never going to reach calculator program. The idea is how we can mask this one and generate input to the calculator and go into the tick mark and go into the calculator to make it crash.

As long as you have the valid input for arbitrary program, it becomes easier to generate the valid input. All possible input are valid for the calculator. You can satisfy by grammar.

Grammar: can specify more languages like regex. Grammar : set of arithmetic expression is like the regular language. You can express any arithmetic expression. Whatever it generates it creates arithmetic expression. This part of the grammar is capturing addition and subtraction operation inside the grammar.

The second one is capturing multiplication and division operation.

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. [Enable Editing](#)

Student - Laura Ong Jin Hua

Idea

S := Expr

Expr := Expr + Term | Expr – Term | Term

Term := Term * Factor | Term / Factor | Factor

Factor := -Factor | (Expr) | Integer | Factor.Integer

Integer := Digit | IntegerDigit

Digit := 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Rules

Capture valid inputs by a grammar

Or and Or and Or.

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week6slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

31

32

33

34

35

36

Derivation of inputs from grammar

Generates $5+2*3$

Capture valid inputs by a grammar

AutoSave File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

week6slides - Protected View - Saved to this PC Student - Laura Ong Jin Hua Share

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.

38

39

40

41

42

43

Web System Testing

Test Code

Converts to browser specific actions

The driver translates program to firefox specific action, and execute whatever you wrote as a program in

firefox and done as a java program and actions as a driver.

What I mean by different element in a web page? You have text boxes and buttons.
How do you locate links in an arbitrary web object. We can write a program to do this for us.

All the things we learn about Junit and Jmock they are test drivers. They are not solving their work of test design. Test design is harder. These are tools for automating test design. There are three types of framework, how we can solve the framework. Selenium has many languages. With some modules not implemented. Junit is for unit testing. It's very important and much harder problem than using these two. When you have the executable problem and don't have source code. We have two ideas: How you partition the input. Every test in one partition behaves differently. This requires some specification.

Jmock is useful because it does not require the class to be in place while testing. What it needs is an expectations() type returned whether it is true or false.

Monday

Monday, March 12, 2018 11:44 AM

AutoSave OK

File Home Insert Draw Design Transitions Animations Slide Show Review View Help Tell me what you want to do

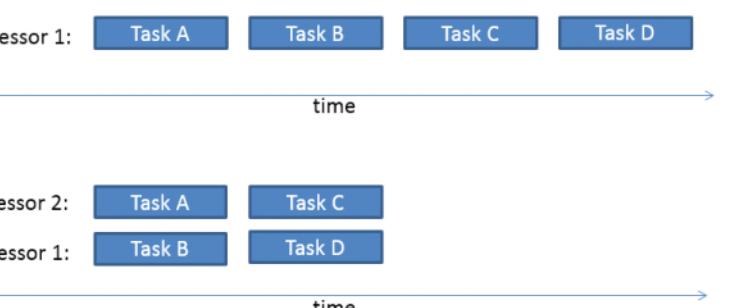
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

3 Example
MSA encountered a URL-encoded website message which contained a malicious exploit. We believe that the attack is going to happen next week from now, we need your help in deciphering the message.

4 Requirement/Analysis
Given a user enters, your program outputs its prime factors within 6 days.
green: pre-condition
red: post-condition
purple: non-functional requirement
[View the original message in Microsoft Edge](#)

5 Design/Implementation
Design
• Use the trial division method
[See C++ code for trial division](#)
Implementation
• Factorizes.exe

6 Testing
Correctness Testing
• 4308602797 (Prime Format Number)
• 1127914895703010279
Performance Testing
[Run tests](#)

7 Concurrency: Benefit
Better resource utilization: with K processors,
ideally we can be K times faster*

Processor 1: Task A, Task B, Task C, Task D
Processor 2: Task A, Task C
Processor 1: Task B, Task D

8 Concurrency: Benefit
Can we get better performance with 2 processors instead?

Processor 1: Task A, Task B, Task C, Task D
Processor 2: Task A, Task C
Processor 1: Task B, Task D

97%