Software Development

CSE 1020

`moodle.yorku.ca`

Software development

As we have already seen in Chapter 3, the process of software development consists of several phases including

- analysis
- design
- implementation
- testing
- maintenance

Analyst

An analyst is responsible for translating the requirements of the customer into a specification.

Software Engineering Requirements (CSE4312)

Designer

A designer/architect is responsible for developing a plan/algorithm to fulfill the specification.

Fundamentals of Data Structures (CSE2011) and Design and Analysis of Algorithms (CSE3101)

Developer

A developer/implementer is responsible for writing code that implements the algorithm.

Introduction to Computer Science I and II (CSE1020 and CSE1030)

Developer

- databases
  Introduction to Databases (CSE3412)
- networks
  Computer Network Protocols and Applications (CSE3214)
- applications
  Introduction to Computer Science I and II (CSE1020 and CSE1030)

Tester

A tester is responsible for checking whether the code satisfies the specification.

Software Engineering Testing (CSE4313)

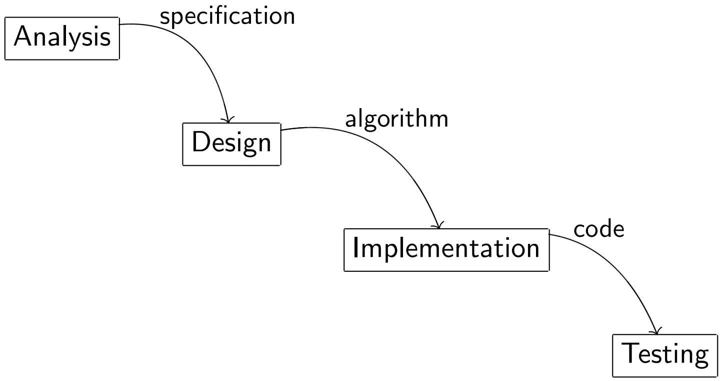
Team Composition

A team may be composed of

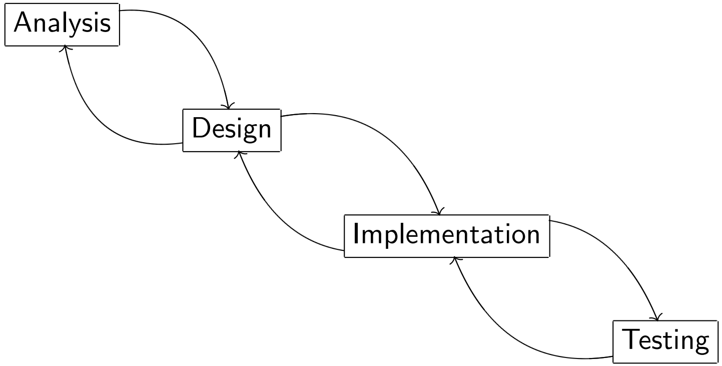
analysts	25%
designers	10%
developers	40%
testers	25%

These numbers are estimates provided by someone in the field of software development.

How does the information flow?

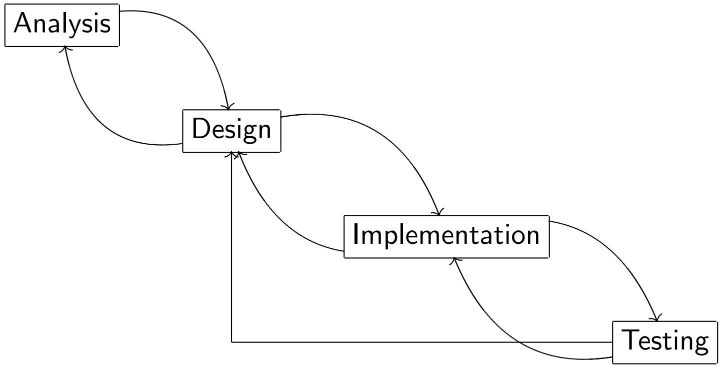


How does our team collaborate?



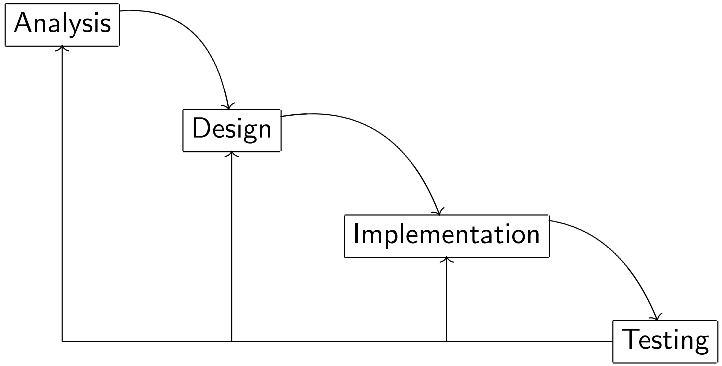
In an ideal world, a phase only has impact on the ones immediately before and after it. However, . . .

Testing may have impact on design



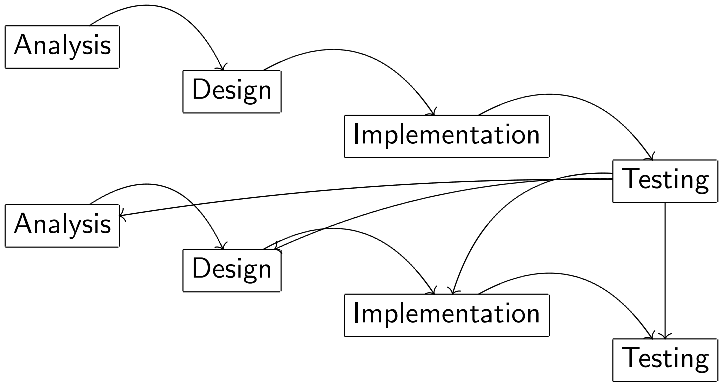
Winston W. Royce. Managing the development of large software systems. In *Proceedings of WESCON*, pages 1–9, Los Angeles, CA, USA, August 1970. IEEE.

Waterfall model



Although the waterfall model is often attributed to Royce, neither the above diagram nor the term "waterfall model" can be found in his paper.

Royce's model



Winston W. Royce. Managing the development of large software systems. In *Proceedings of WESCON*, pages 1–9, Los Angeles, CA, USA, August 1970. IEEE.

Overview of development methodologies

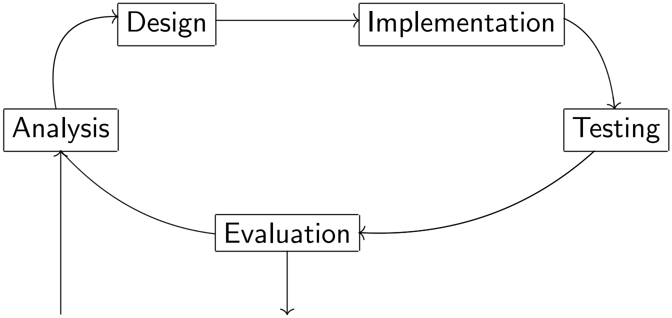
waterfall model	do it once	risky
Royce's model	do it twice	less risky
	do it . . .	even less risky

Overview of development methodologies

waterfall model	do it once	risky
Royce's model	do it twice	less risky
IID	do it many times	even less risky

IID = iterative and incremental development

Iterative and Incremental Development



Example of IID projects

project: command and control system for submarine
iterations: four iterations of six months each

Craig Larman and Victor R. Basili. Iterative and incremental development: a brief history. *IEEE Computer*, 36(6):47–56, June 2003.

Example of IID projects

project: light airborne multipurpose system
iterations: 45 iterations of one month each

Craig Larman and Victor R. Basili. Iterative and incremental development: a brief history. *IEEE Computer*, 36(6):47–56, June 2003.

Different IID methodologies

- extreme programming (XP)
  Software Design (CSE3311)
- rational unified process (RUP)
- . . .

EXTREME PROGRAMMING
I CAN'T GIVE YOU ALL OF THESE FEATURES IN THE FIRST VERSION.
AND EACH FEATURE NEEDS TO HAVE WHAT WE CALL A "USER STORY."
OKAY, HERE'S A STORY: YOU GIVE ME ALL OF MY FEATURES OR I'LL RUIN YOUR LIFE.
scottadams@aol.com
www.dilbert.com
© 2002 United Feature Syndicate, Inc.

Unified Modeling Language (UML)

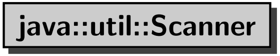
UML was designed by “the three amigos” Grady Booch, Ivar Jacobson and James Rumbaugh in the mid 1990s.

UML provides a large variety of different types of diagrams:

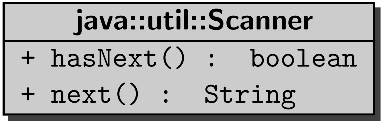
- class diagrams
- object diagrams
- activity diagrams
- interaction diagrams
- ...

These diagrams can be used to model software.

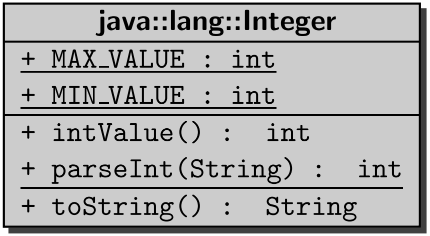
A Class Diagram



A Class Diagram



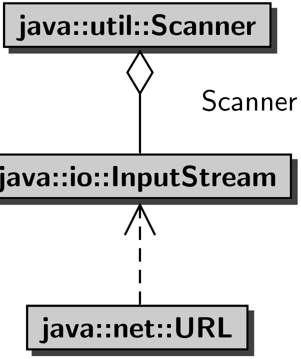
A Class Diagram



How are these classes related?

```
URL url = new URL("http://www.sochi2014.com/en/medals");
InputStream stream = url.openStream();
Scanner urlInput = new Scanner(stream);
...
while (urlInput.hasNextLine()) {
  String line = urlInput.nextLine();
  ...
}
urlInput.close();
```

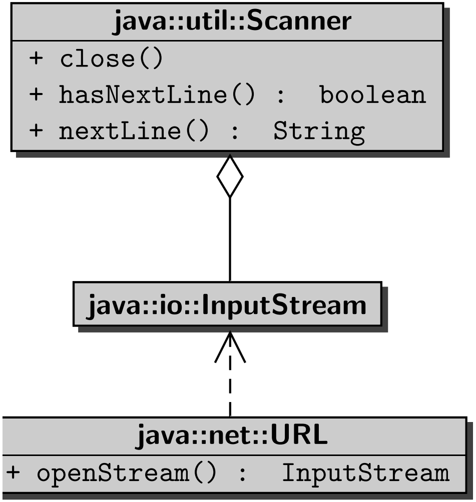
A Class Diagram



Scanner has-a InputStream (Chapter 8)

URL uses InputStream

A More Detailed Class Diagram



Testing

Question

Should we test?

Based on the software developer and user surveys, the national annual costs of an inadequate infrastructure for software testing is estimated to range from \$22.2 to \$59.5 billion.

The Economic Impacts of Inadequate Infrastructure for Software Testing. Planning Report 02-3. May 2002.

Testing

Question

Should we test?

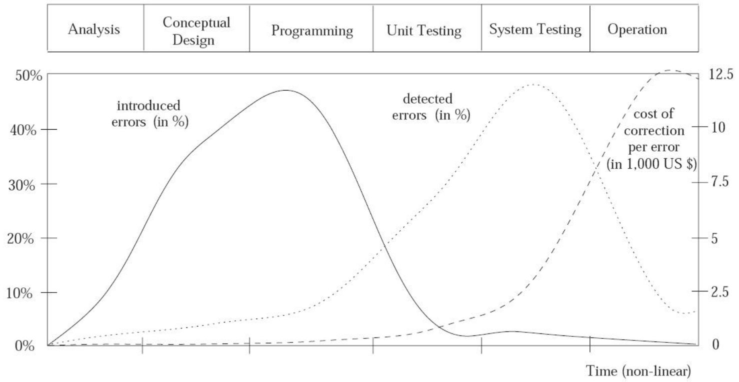
Based on the software developer and user surveys, the national annual costs of an inadequate infrastructure for software testing is estimated to range from $22.2 to $59.5 billion.

The Economic Impacts of Inadequate Infrastructure for Software Testing. Planning Report 02-3. May 2002.

Answer

Yes!

Testing



P. Liggesmeyer, M. Rothfelder, M. Rettelbach and T. Ackermann. Qualitätssicherung Software-basierter technischer Systeme. *Informatik Spektrum*, 21(5):249–258, 1998.

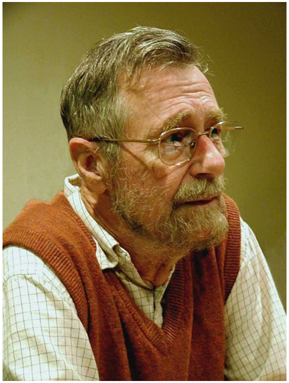
However ...

“Program testing can be used to show the presence of bugs, but never to show their absence!”

Edsger W. Dijkstra. Notes on structured programming. Report 70-WSK-03, Technological University Eindhoven, April 1970.

Edsger Wybe Dijkstra

- Member of the Royal Netherlands Academy of Arts and Sciences (1971)
- Distinguished Fellow of the British Computer Society (1971)
- Recipient of the Turing Award (1972)
- Foreign Honorary Member of the American Academy of Arts and Sciences (1975)


Edsger Wybe Dijkstra

(1930–2002)

Another Way to Find Bugs

Formal verification: proving that code satisfies particular properties of interest.

The two most used approaches to formal verification are

- model checking
- theorem proving

Introduction to Program Verification (CSE3341)

How to Test Code?

$$\begin{tikzcd} {} \arrow[r, "\text{input}"] & \boxed{\text{Code}} \arrow[r, "\text{output}"] & {} \end{tikzcd}$$

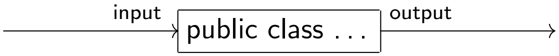
- Provide the input.
- Run the code.
- Compare the output with the expected output.

Terminology

Test case: an input that satisfies the precondition.

Test suite/test vector: a collection of test cases.

White Box Testing



Black Box Testing

$$\xrightarrow{\text{input}} \boxed{\text{Code}} \xrightarrow{\text{output}}$$

Why Black Box Testing?

A Java archive (JAR) file usually only contains the bytecode and not the Java code.

Developers can obfuscate JAR files so that a user of the JAR file does not get much information regarding the original Java code.

How to Provide the Test Cases?

- Enter the test cases manually.
- Read the test cases from files.
- Generate the test cases by an app.
- Use the `launch` method of the `ToolBox` class.
- Use a testing framework such as JUnit.

How to Determine the Expected Result?

- Use a different solution to the problem that is known to be correct.
- Use an approximate solution to the problem.
- ...

How to Compare the Result with the Expected Result?

- Check it manually.
- Read the expected result from a file.
- Generate the expected result by an app.
- Use a testing framework such as JUnit.

Sometimes, it is much easier checking that the output is correct than computing the output. For example, it is much easier checking that a list of elements is sorted than sorting a list of elements.

Which Test Cases?

- Likely cases (black box and white box testing).
- Boundary cases (black box and white box testing).
- Cases that cover all execution paths (white box testing only).

Testing

Exercise

Test the method `parseInt` of the class `Integer` of the package `com.cheapbutquestionable`. Its API can be found here and its jar can be found here.

To do

- Study Chapter 7 of the textbook.