Welcome to Programming for Mobile Computing

EECS 1022

`moodle.yorku.ca`

Course webpage

moodle.yorku.ca (not set up yet)
and
www.eecs.yorku.ca/course/1022

Instructor

- Name: Franck van Breugel
- Email: franck@eecs.yorku.ca
- Office: Lassonde Building, room 3046
- Office hours: Mondays and Wednesdays, 17:30-18:20, or by appointment

Timeline

- Saturday, 4pm: instructor might not be able to teach the course.
- Sunday, 8pm: instructor is not able to teach the course.
- Monday, 1pm: I decided to teach the course.
- Monday, 7pm: first lecture is underway.

Evaluation

- Weekly: labs (25%)
- Week of July 10: midterm (25%)
- Exam period (August 2–11): final exam (50%)

The midterm will consist of a written part of one hour that will take place during lecture time and a programming part of one hour that will take place during lab time.

The final exam will be three hours and will consist of a written part and a programming part.

Labs

- William Small Center, room 106 and 108
- Tuesdays and Thursdays, 15:30-18:30 and 18:30-21:30
- Attend the lab for which you are registered

Labs will start tomorrow. See `www.eecs.yorku.ca/course/1022/labs/lab1.html` for the first lab.

Drop deadline

July 21

Until this date you can drop the course without getting a grade for it and, hence, it will not affect your gpa.

July 22–31

During this period you can still drop the course but you will receive a W on your transcript. The W will not affect your gpa.

`www.registrar.yorku.ca/enrol/dates/su17.htm` contains important dates.

Academic honesty

"If you put your name on something, then it is your work, unless you explicitly say that it is not."

http://secretariat-policies.info.yorku.ca/policies/academic-honesty-senate-policy-on/ contains more details.

Learning outcomes

By the end of the course, the students will be able to:

- Understand software development within an object-oriented framework using a modern programming language and tool set.
- Use a set of computing skills such as reasoning about algorithms, tracing programs, test-driven development, and diagnosing faults.
- Explain and apply fundamental constructs in event-driven programs, including variables and expressions, control structures (conditionals/loops), and API usage.
- Write simple programs using a given software infrastructure, API, and tool chain.
- Gain exposure to a comprehensive mobile computing framework.
- Gain exposure to user interface design.

Key topics

- Abstraction and separation of concerns
- The software development cycle
- Object-oriented programming

- Android app development
- User interface design
- The Java programming language

Approach

- Builds on EECS 1012
  Separation of concerns, computational thinking
- Industrial-strength tools
  User interface via XML (not HTML); Behaviour via Java (not JavaScript).
- Platform
  Operating system is Android; Integrated development environment is Studio
- Experiential Pedagogy
  Foundational concepts in class; Projects in lab

The Software Development Cycle

EECS 1022

`moodle.yorku.ca`

Software development

The process of software development consists of several phases including

- analysis
- design
- implementation
- testing
- deployment
- maintenance

Analyst

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

Software Engineering Requirements (EECS 4312)

Designer

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

Fundamentals of Data Structures (EECS 2011) and Design and Analysis of Algorithms (EECS 3101)

Developer

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

Advanced Object Oriented Programming (EECS 2030)

Developer

- databases
  Introduction to Databases (EECS 3412)
- networks
  Computer Network Protocols and Applications (EECS 3214)
- mobile
  This course

Tester

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

Software Engineering Testing (EECS 4313)

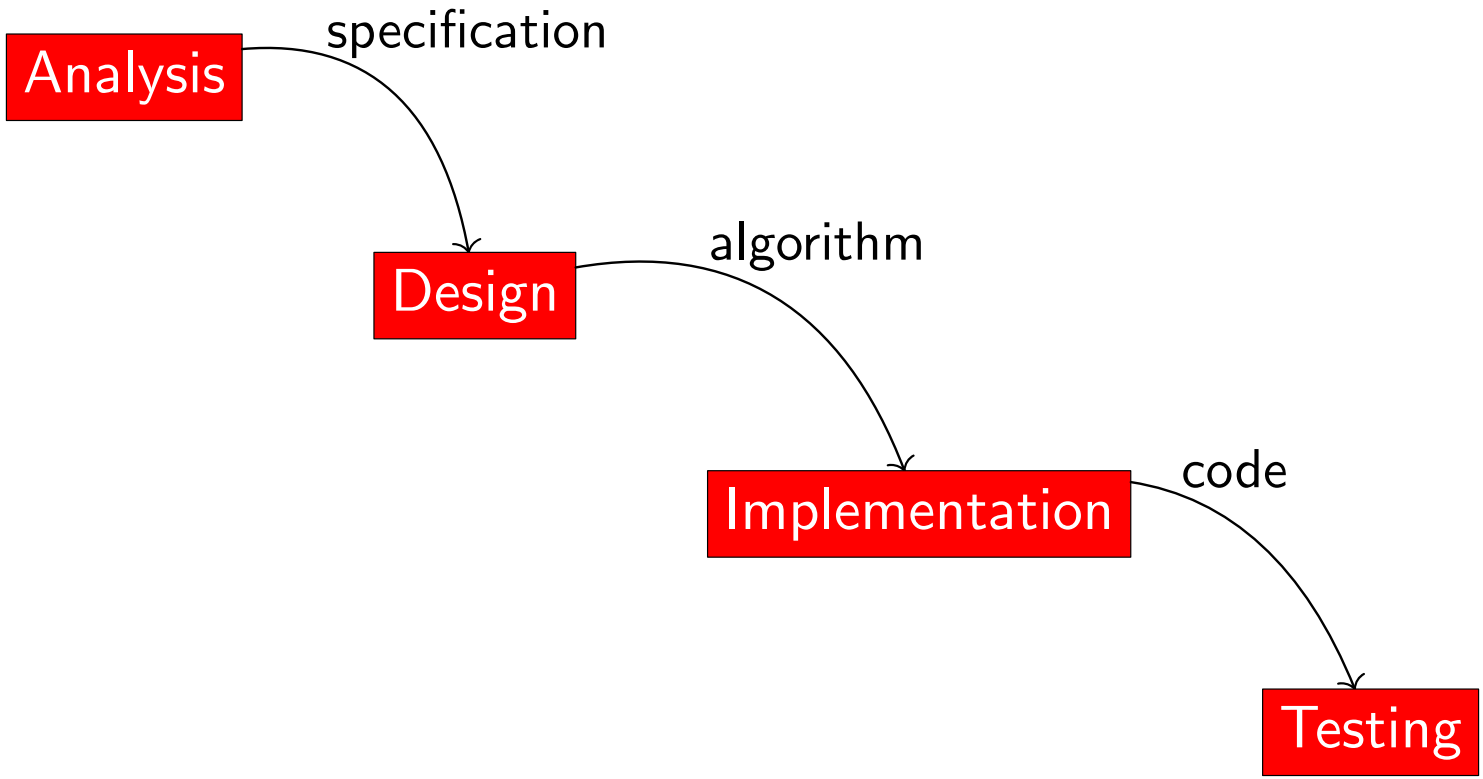
Team composition

A team may be composed of

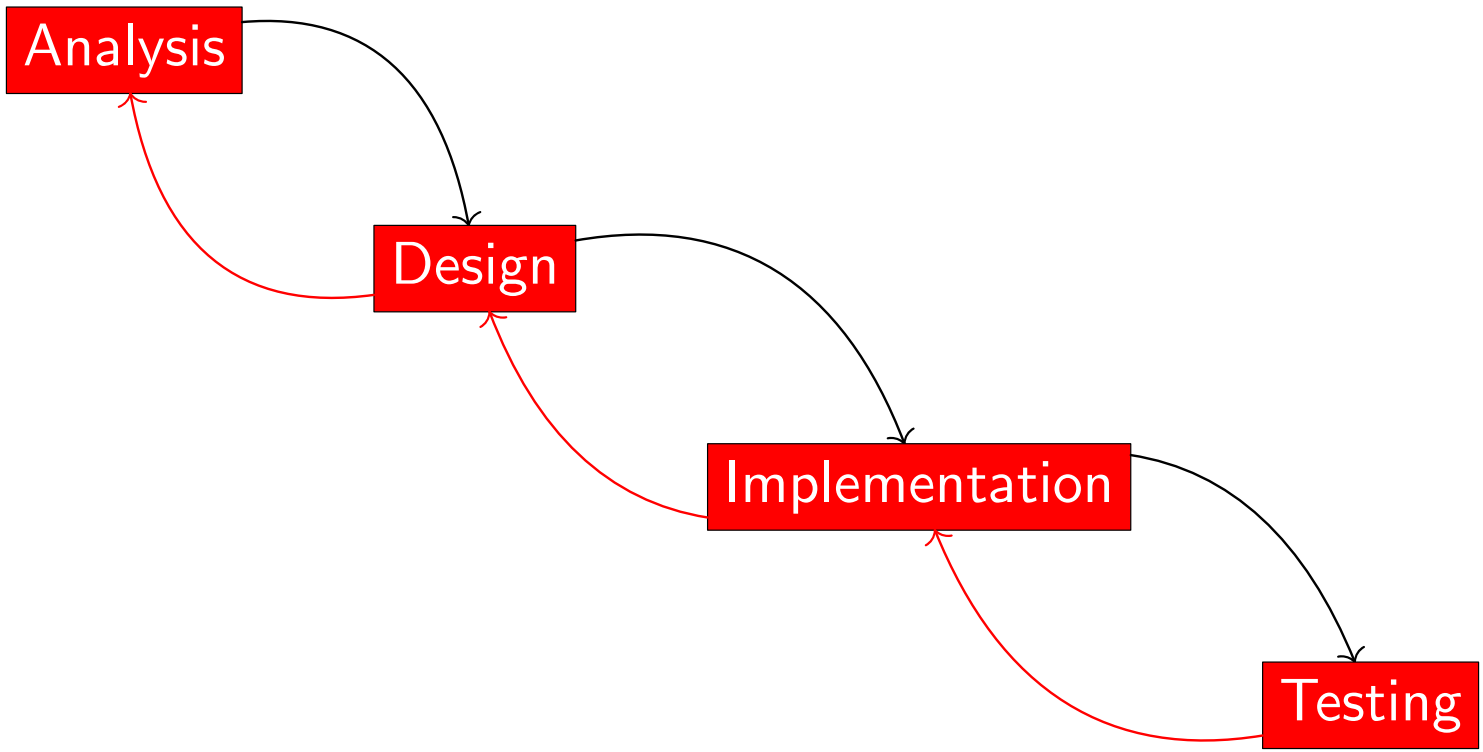
analysts	25%
designers	10%
developers	40%
testers	25%

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

How does the information flow?

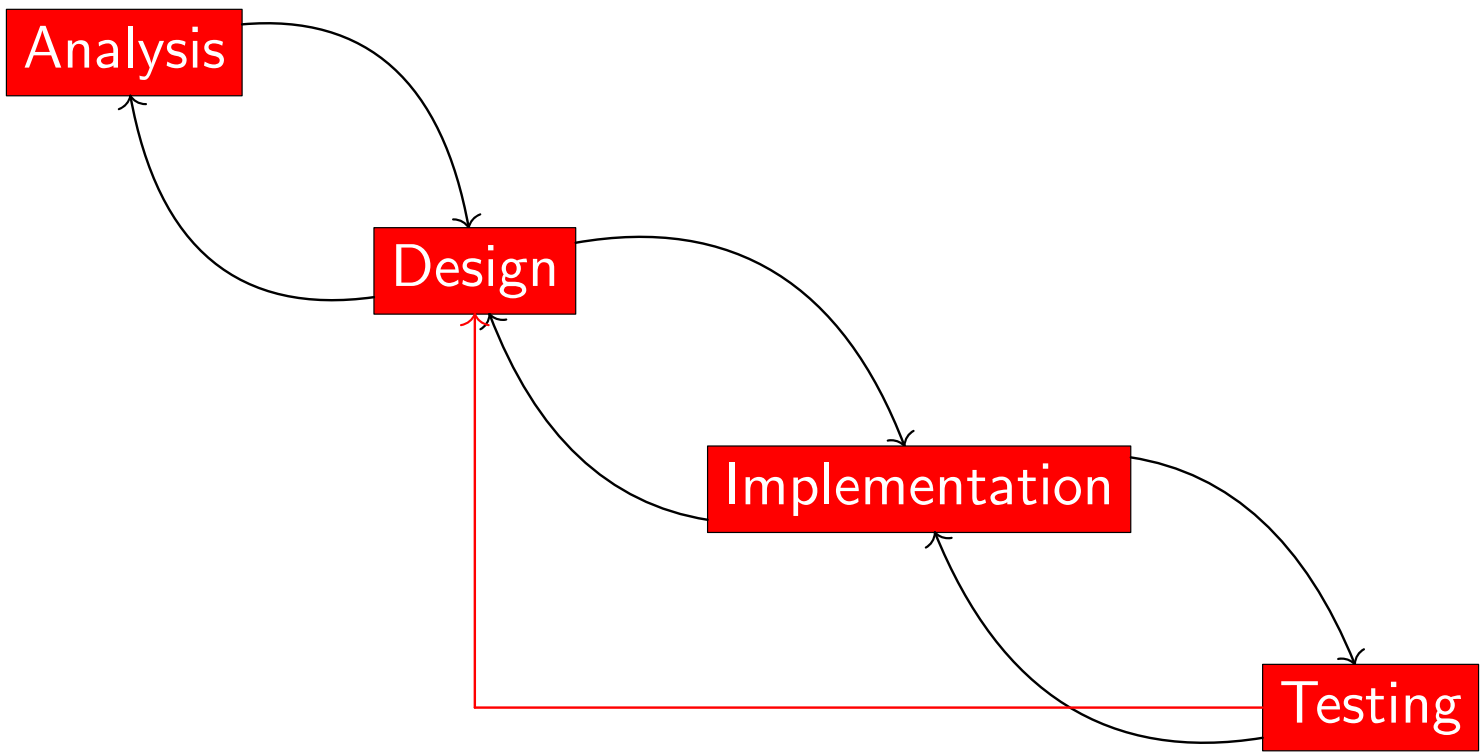


How does our team collaborate?



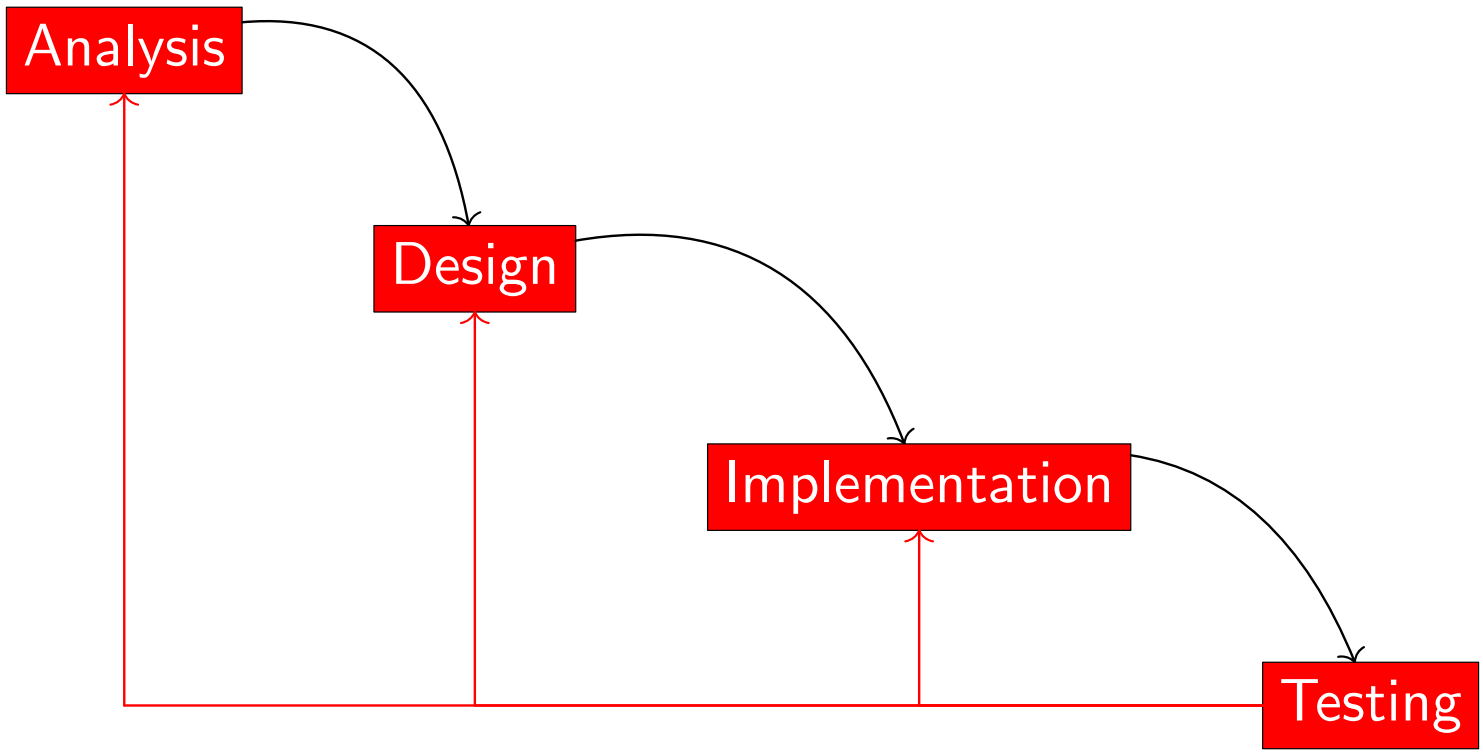
In an ideal world, a phase only has impact on the ones immediately before and after it. However, ...

Testing may have impact on design



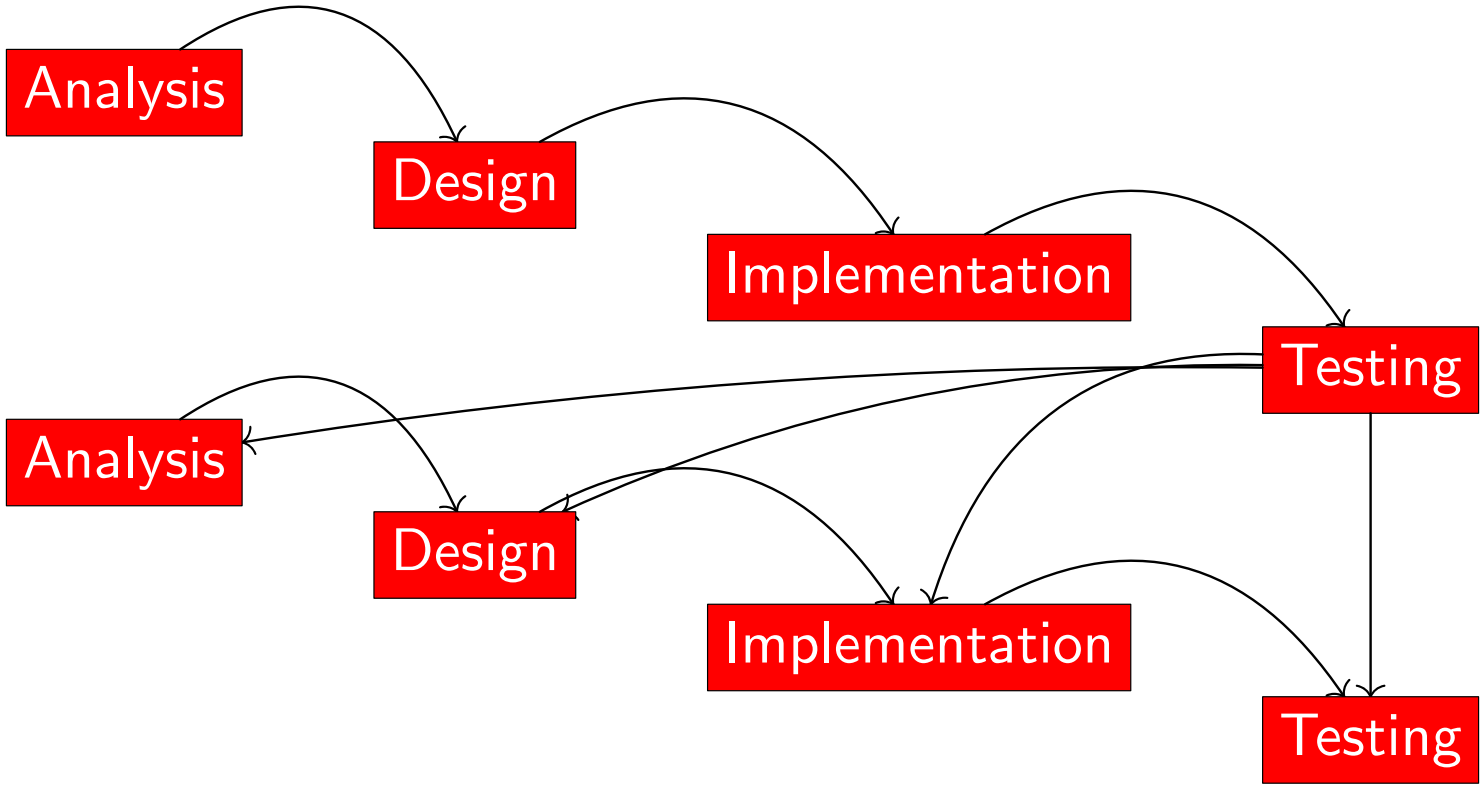
Winston W. Royce. Managing the development of large software systems. In *Proceedings of WESCON*, pages 1–9, Los Angeles, CA, USA, August 1970. IEEE.

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

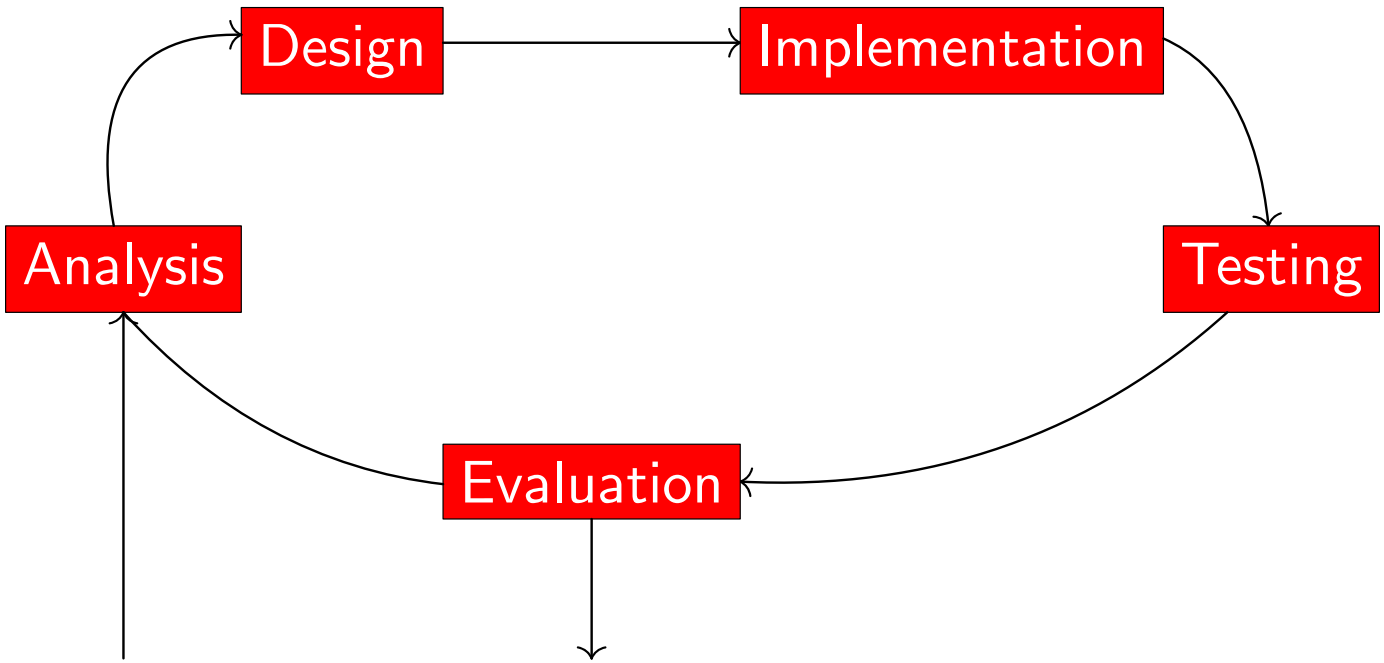
waterfall model	do it once	risky
Royce's model	do it twice	less risky
	do it ...	even less risky

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 (EECS 3311)
- rational unified process (RUP)
- ...

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.

Question

Should we test?

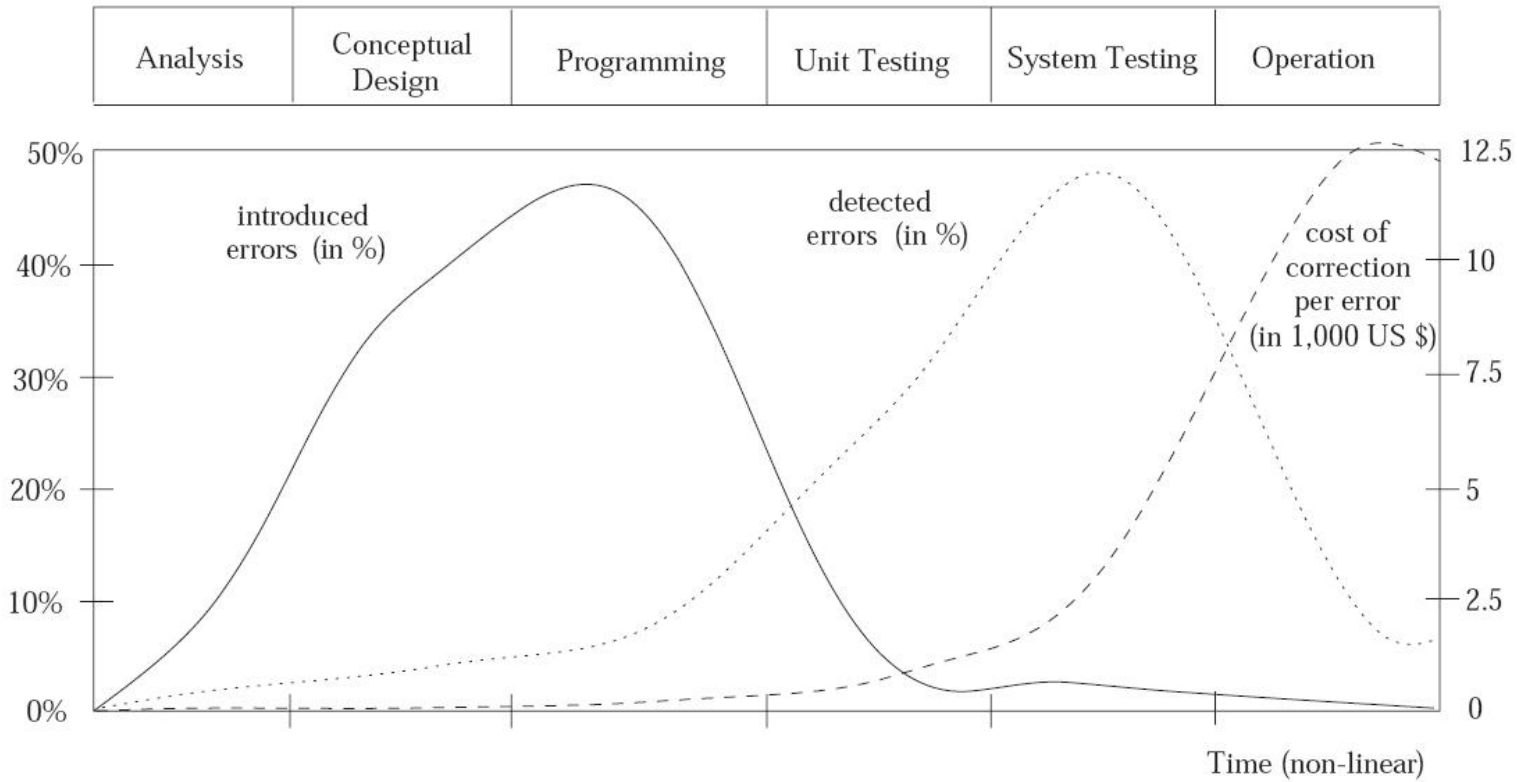
Based on the software developer and user surveys, the national annual costs of an inadequate infrastructure for software testing is estimated to range from $22.2 to $59.5 billion.

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.

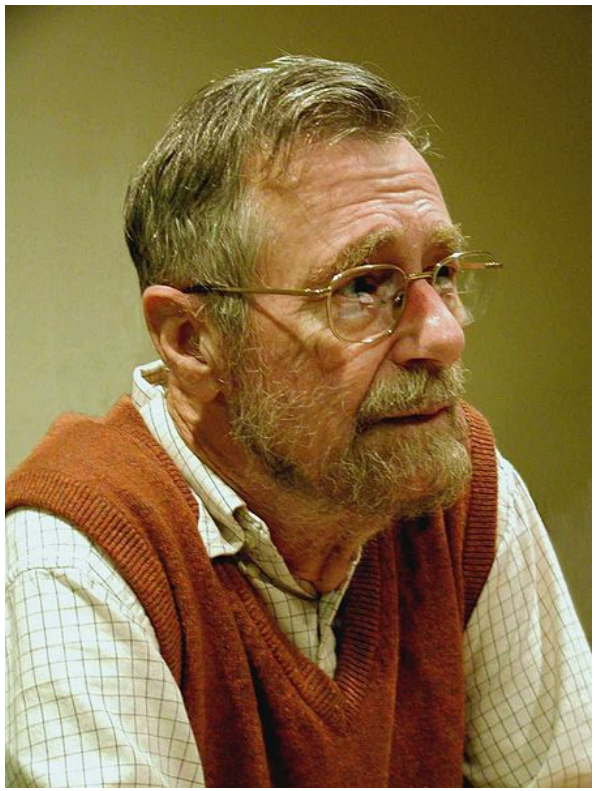
However ...

"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 (1930–2002)

- 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

Source: Hamilton Richards

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 (EECS 3341)