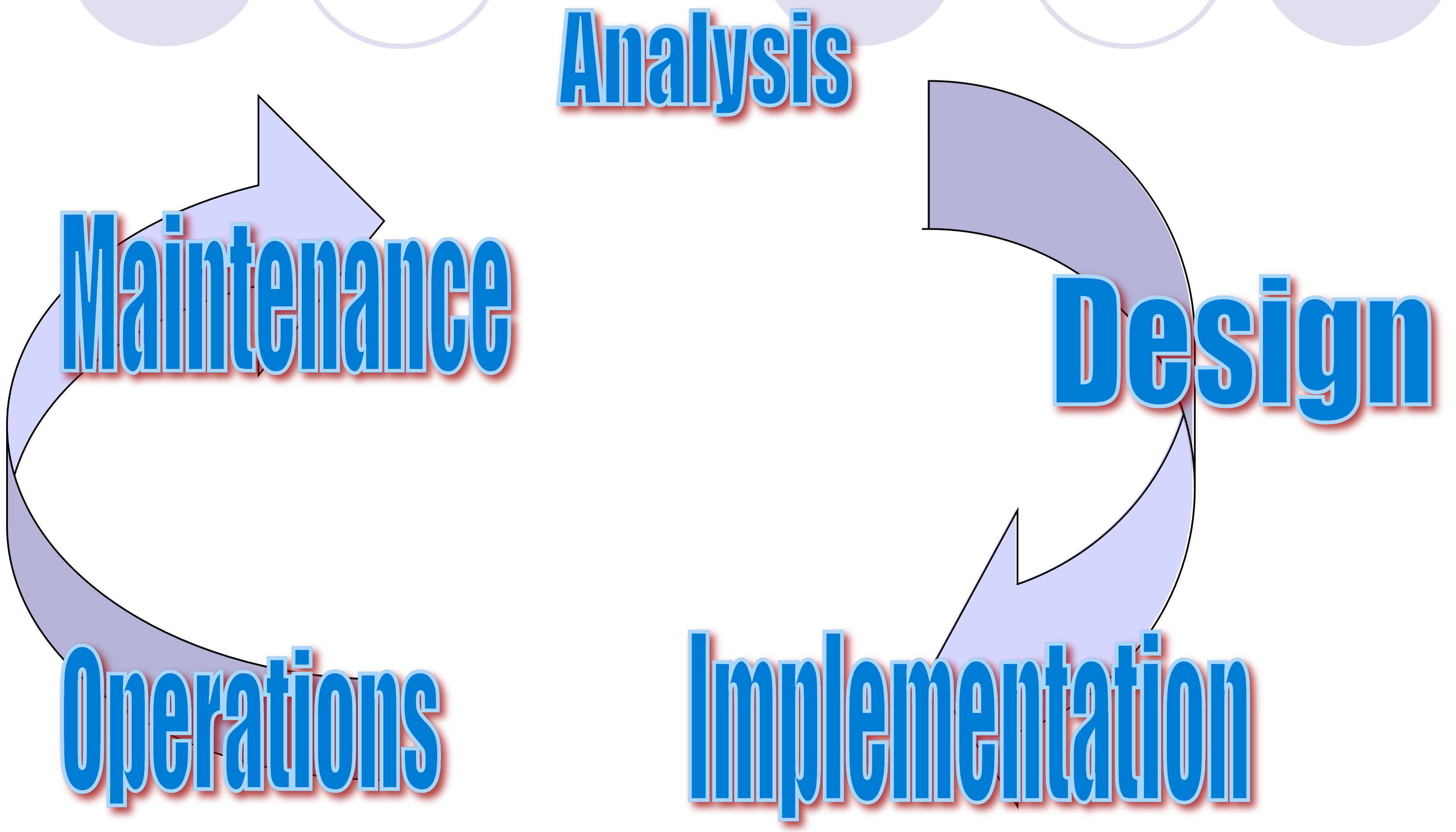
Section 1.1

Systems Life Cycle

Slides 1 to 13 are by Larry Baker for IB Computer Science

Systems Life Cycle - 1.1.1 : Major stages

5 PHASES



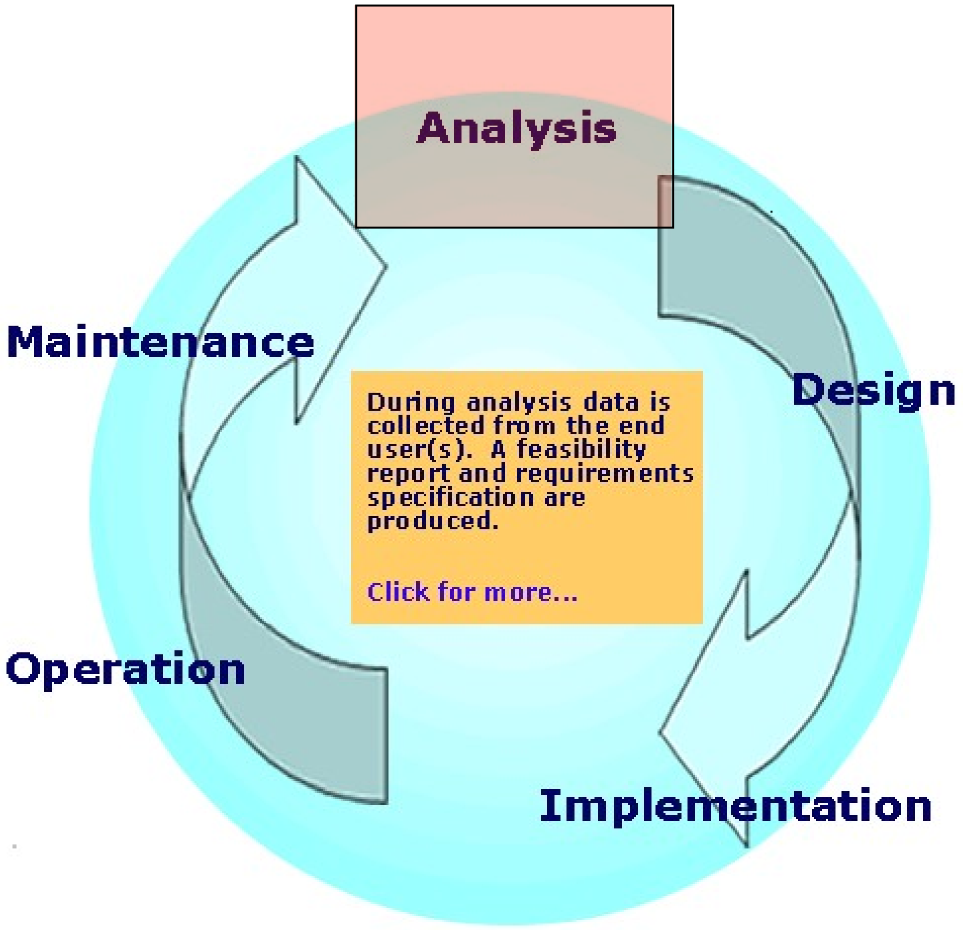
Let’s look at these phases in detail…….

Systems Life Cycle

1. Analysis
   - Collect and examine data
   - Analyze current system and data flow
2. Design
   - Plan your system, its data flow, data structures, algorithms, and modules
3. Implementation
   - Coding
   - Schedule implementation goals and milestones
   - Unit testing
4. Operation
   - Installing the system
   - How to use it? (documentation!)
5. Maintenance
   - Finding bugs and patching them

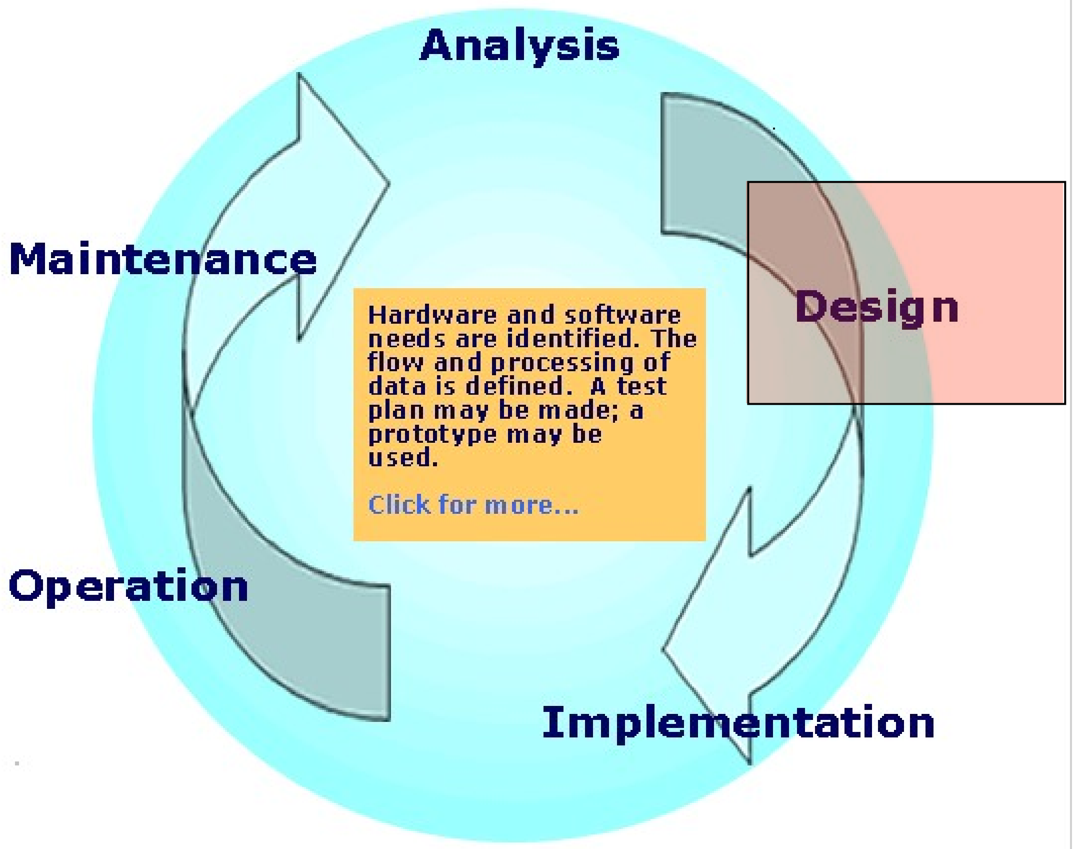
- WHY IS IT CYCLICAL?

Systems Life Cycle - 1.1.1 : Major stages



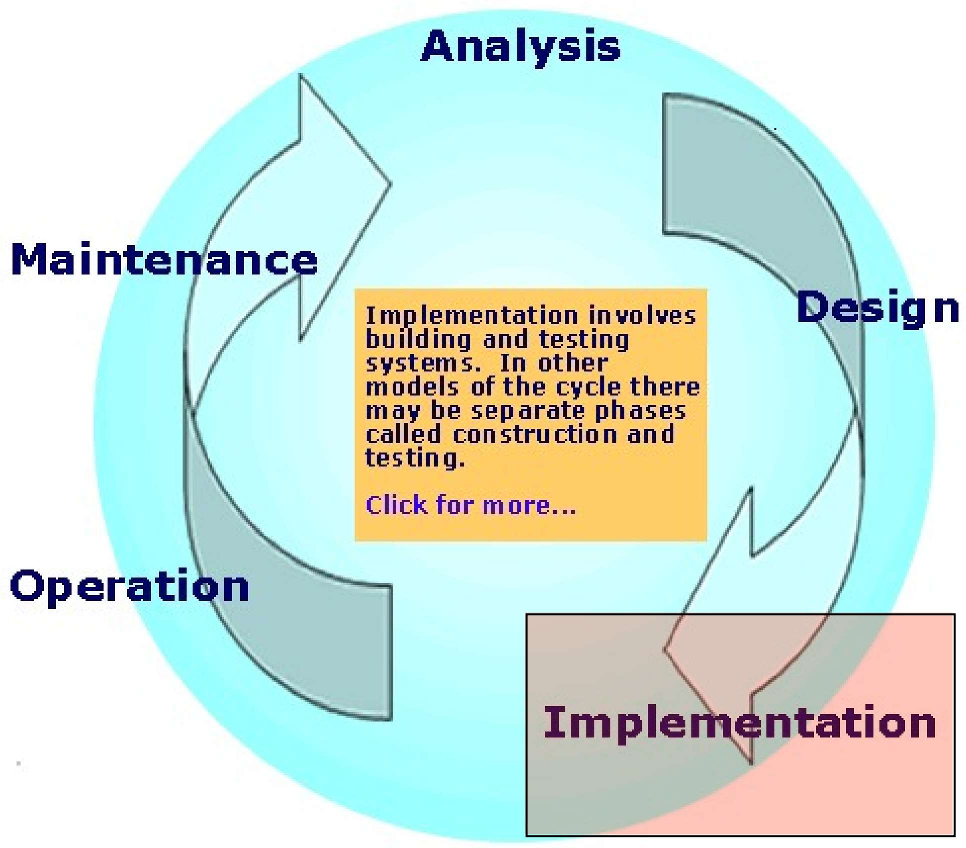
During analysis data is collected from the end user(s). A feasibility report and requirements specification are produced.

Systems Life Cycle - 1.1.1 : Major stages



Hardware and software needs are identified. The flow and processing of data is defined. A test plan may be made; a prototype may be used.

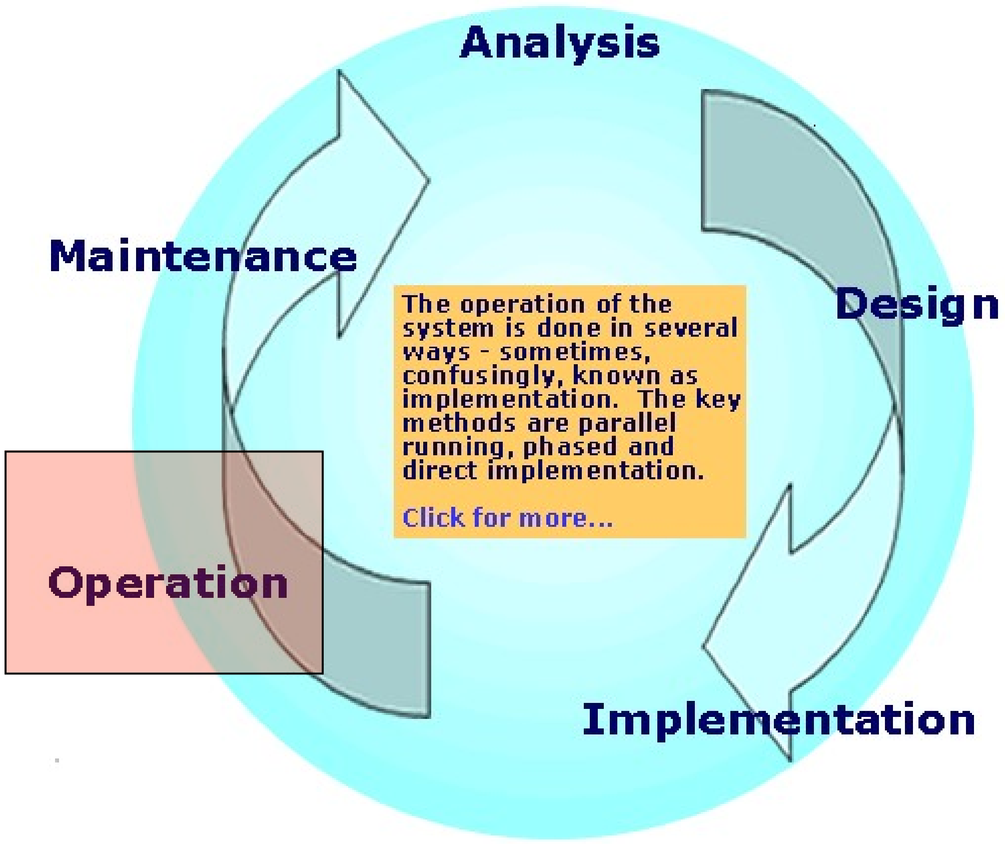
Systems Life Cycle - 1.1.1 : Major stages



Implementation involves building and testing systems. In other models of the cycle there may be separate phases called construction and testing.

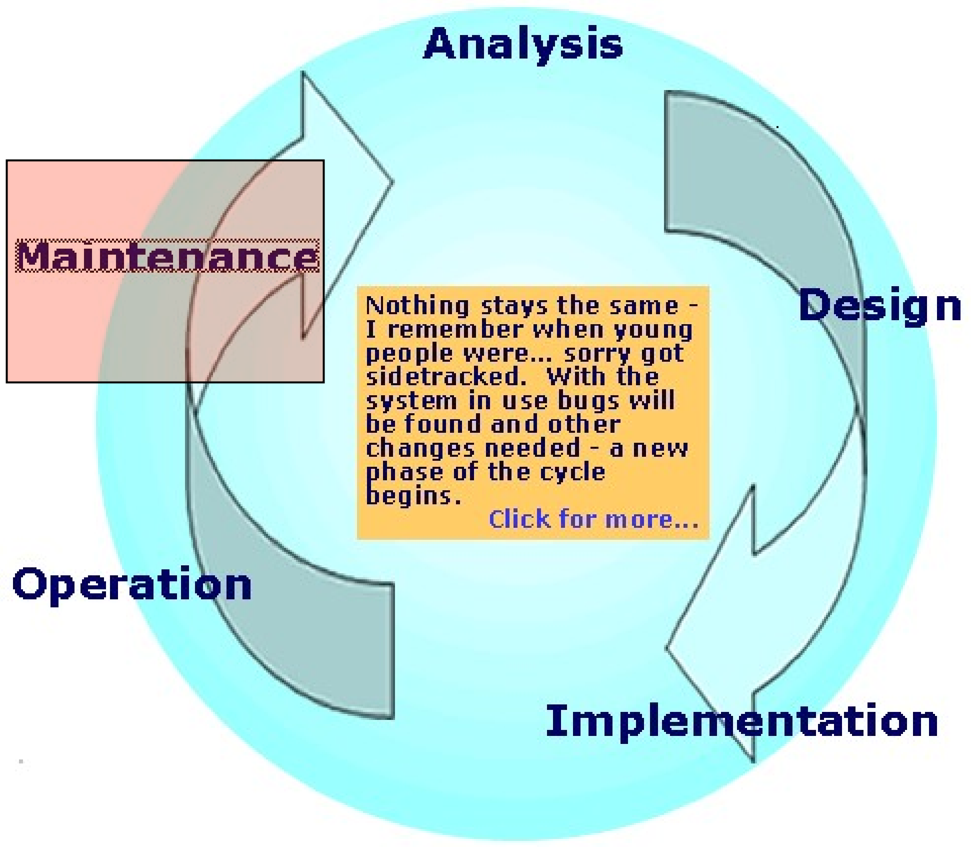
Click for more...

Systems Life Cycle - 1.1.1 : Major stages



The operation of the system is done in several ways - sometimes, confusingly, known as implementation. The key methods are parallel running, phased and direct implementation.

Systems Life Cycle - 1.1.1 : Major stages


Analysis
Design
Implementation
Operation
Maintenance
Nothing stays the same - I remember when young people were... sorry got sidetracked. With the system in use bugs will be found and other changes needed - a new phase of the cycle begins.
Click for more...

Data collection in the Analysis Phase

- Identifies
  - Who inputs data to the system
  - What form the data is in
  - Any validation that is needed
  - What processing is done to produce the required results
- With projects of any size, the analysis stage of gathering data is essential
  - Gain clear insight into data input, processing and output without thinking about the computer system that may eventually be used.

Collecting Data

- Methods of data collection:
  - Interviews
  - Questionnaires
  - Search existing documents
  - Background research
  - Observe people using the current system

Data Collection Techniques – 1.1.3

- **Data collection**
  - This process is sometimes referred to (or sometimes taken to include) fact finding. The classic fact-finding methods are:
    - **Conduct interviews**
    - **Carry out questionnaires**
    - **Study existing documents**
    - **Search the literature for other solutions to the same problem.**
    - **Observe people working with the existing system**

Requirements Specification - 1.1.4

- If there is an initial feasibility report and it is accepted then more detail will be required, this comes in the form of a requirements specification. This may include:
  - the data inputs and outputs required for the system to work
  - a list of tools that might be used to produce the system
  - a list of people needed to build the system
  - a schedule for the subsequent stages of the project
  - the likely cost of the system
  - the likely economic impact (eg staff training, benefits to customers etc)

Requirements Specification

- Software and hardware requirements
- Descriptions of the specific things that your system will be able to do
- Formal agreement (contract) between the client and the developer

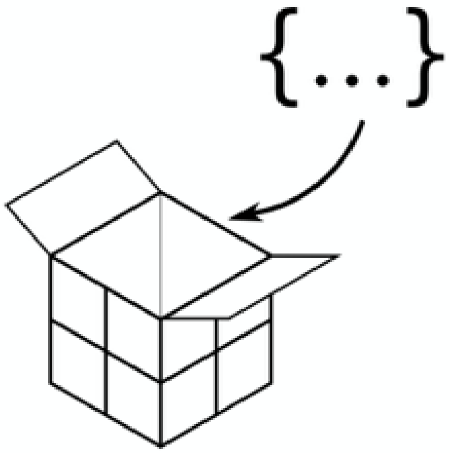
Why do we study software engineering?

- A computer is a programmable device
    - So programming it is a fundamental activity
- We ask a lot from our software
    - Complexity
    - Heterogeneous tasks
        - inside a car vs. inside a server
- Managing that complexity requires more than just programming skill

The rest of the slides in this lecture are by Kostadin Damevski at VCU

What should you expect to learn in this course?

- Methodologies
- Techniques
- Tools
- …to build high-quality software that fits budget



VCU School of Engineering | Computer Science

Software Development Effort

Size	Example
10^2 LOC	Class Exercise
10^3 LOC	Small Project
10^4 LOC	Term Project
10^5 LOC	Business Application
10^6 LOC	Word Processor
10^7 LOC	Operating System

Programming Effort

Software Engineering Effort

Software Failure Du Jour

- "Southwest Airlines computer glitch causes cancellations, delays for third day" —

  www.washingtonpost.com (published 7/22/2016; retrieved 8/26/2016)

  - "Despite explanations and apologies, Southwest Airlines delays and cancellations caused by a computer glitch continued for the third day in a row. […] The company said that "more than 250" of its 3,900 daily departures were canceled."
  - "Delays and cancellations persisted for Southwest Airlines customers around the country Thursday as the company dealt with the fallout of a massive software glitch that occurred Wednesday."

Software Failures Hall of Fame

- Ariane 5 rocket explosion
- Therac-25 lethal radiation overdose
- Mars Climate Orbiter disintegration
- FBI Virtual Case File project abandonment
- HealthCare.gov

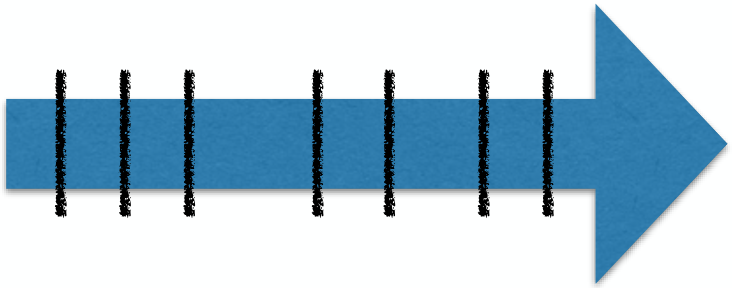
Engineering SW is different from engineering HW

- ***Question:*** Why so many SW disasters and few HW disasters?
- *One possible answer:* Nature of the two media (easy to modify vs. hard to modify) and subsequent cultures that developed
- *Another possible answer:* Software engineering is a young discipline
- *Another possible answer:* Software complexity is unprecedented

Engineering SW is different from HW

- Cost of field update
    - On hardware it's very large
        - Hardware designs have to be finished before they are shipped
        - Bugs => return hardware => lost $$$
    - On software it's very small
        - Expect software to get better over time
        - Bugs => wait for an upgrade
- HW decays slowly, while software can be long lasting

Software Development



The SW engineering task is split into multiple steps according to a ***software development process***

Software Development Processes

- Waterfall vs. Spiral vs. Agile

Do we need a software process?

- Designing simple software (such as homework assignments) has two steps
  - Step 1: Think!
  - Step 2: Code!
- Both steps are creative
  - Programmers are happy doing them
  - Customers are happy to pay programmers to do this

Do we need a software process?

- The two step process doesn't scale up with complexity, for example:
    - How do we split the work among a team of people?
    - How do we ensure we know what the customer meant when they asked for feature X?
    - How do we ensure we give them only what they pay for?
    - What about cross-cutting concerns, such as security, accountability, performance, upgradeability etc?

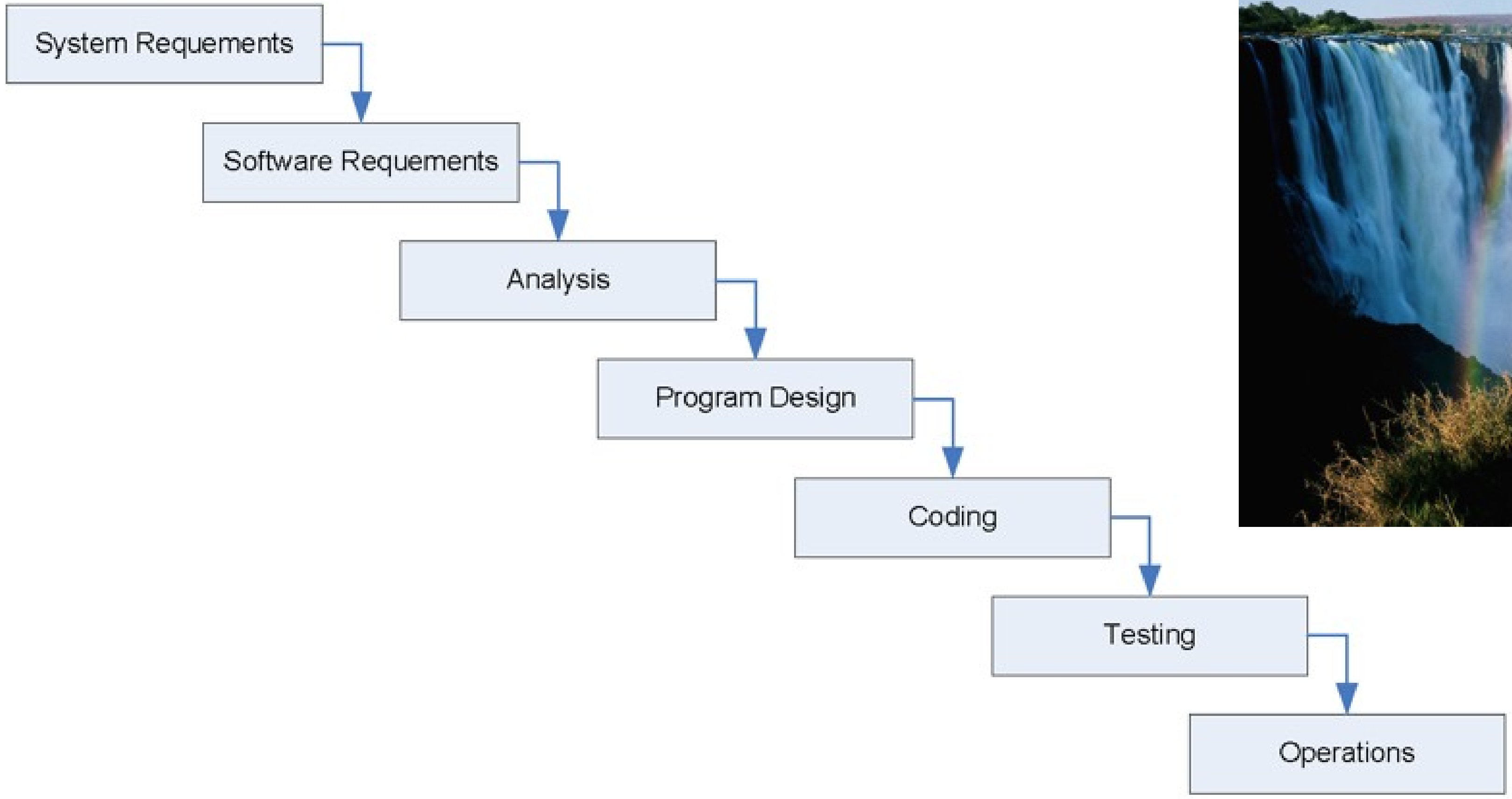
Stages in engineering software

- Typical stages in software engineering:
    - Requirements Analysis
    - Software Design
    - Implementation
    - Testing
    - Maintenance
- Some of these can be further split (e.g. system req's and software req's), and more can be added (e.g. deployment)

What is a software development process?

- Let’s revisiting this question in the context of stages
- A software process model determines the order of the stages involved in software development and evolution
- It provides the answer to the following two questions?
    - What shall we do next?
    - How long shall we continue to do it?

1st Development Process: Waterfall (1970)



How well does the waterfall model work?

- *"And the users exclaimed with a laugh and a taunt: 'It's just what we asked for, but not what we want.'" — Anonymous*
- The Good
    - Simple!!!
    - Plenty of documentation, which is good (allows for management of project)
    - Still in use since the 70s
- The Bad
    - Testing is towards the end of the model, and it may expose fundamental problems, requiring rework
    - Customer is not involved

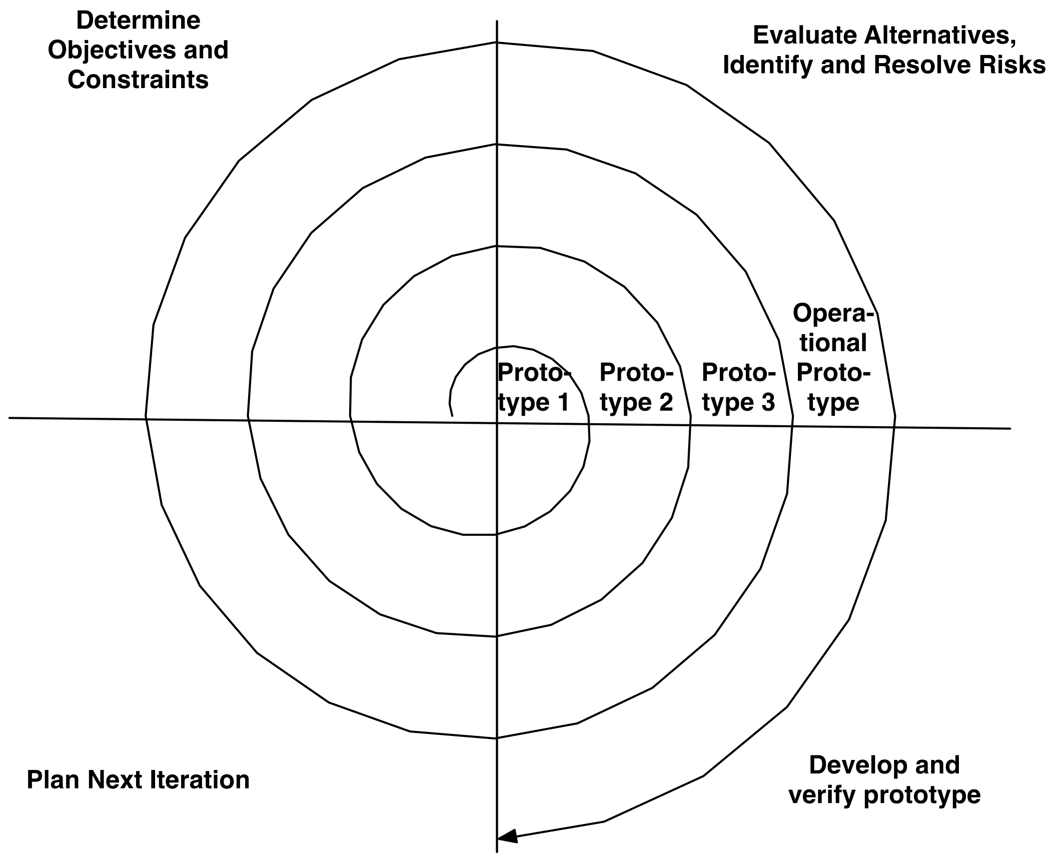
How well does the waterfall model work?

- *"Plan to throw one [implementation] away; you will, anyhow."*
  — Fred Brooks, Jr.
  (1999 Turing Award winner)
- Often after building first one, developers learn right way they should have built it

Spiral model (1986)

- Combine Plan-and-Document with prototypes
- Rather than plan & document all requirements 1st, develop plan & requirement documents across each iteration of prototype as needed and evolve with the project

Spiral model


Determine Objectives and Constraints
Evaluate Alternatives, Identify and Resolve Risks
Proto-type 1
Proto-type 2
Proto-type 3
Opera-tional Proto-type
Plan Next Iteration
Develop and verify prototype

Spiral model, good and bad

- The Good
    - Iterations involve the customer before the product is completed
        - Reduces chances of misunderstandings
    - Risk management part of lifecycle
    - Project monitoring easy
    - Schedule & cost more realistic over time
- The Bad
    - Iterations 6 to 24 months long
        - Time for customers to change their minds!
    - Lots of documentation per iteration
    - Lots of rules to follow, hard for whole project
    - Cost of process is high
    - Hard to meet budget and schedule targets

Plan and document models failed often

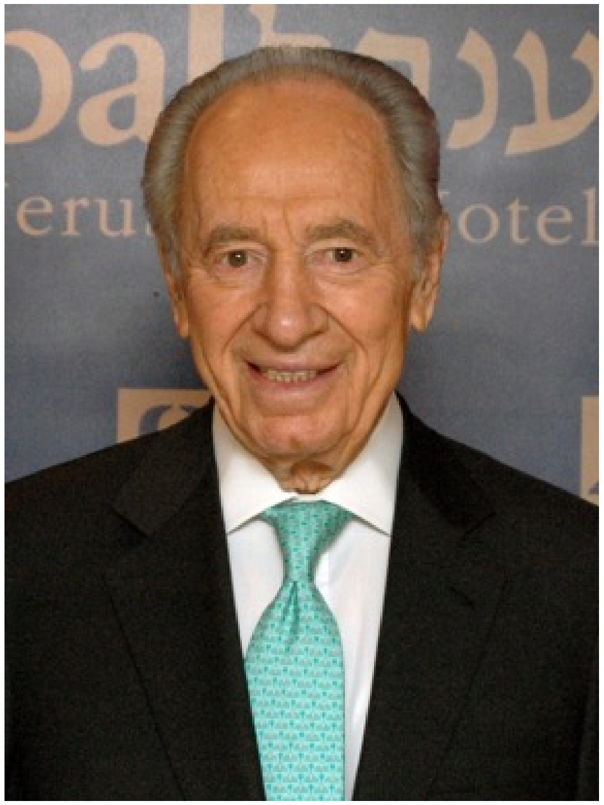
- Often missing the cost, schedule, & quality target
- P&D requires extensive documentation and planning and depends on an experienced manager
    - Can we build software effectively without careful planning and documentation?
    - How to avoid “just hacking”?

- *Plan & Document: Waterfall & Spiral Models*

Peres' Law

- *"If a problem has no solution, it may not be a problem, but a fact, not to be solved, but to be coped with over time."*

  — Shimon Peres (winner of 1994 Nobel Peace Prize for Oslo accords)



Agile Development Processes

WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
www.dilbert.com
scottadams@aol.com
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
11-26-07 © 2007 Scott Adams, Inc./Dist. by UFS, Inc.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.

Agile manifesto

- “We are uncovering better ways of developing SW by doing it and helping others do it. Through this work we have come to value
- ***Individuals and interactions*** over processes & tools
- ***Working software*** over comprehensive documentation
- ***Customer collaboration*** over contract negotiation
- ***Responding to change*** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.”

Agile development model

- Embraces change as a fact of life: continuous improvement vs. strict phases
- Developers continuously refine working but incomplete prototype until customers happy, with customer feedback on each Iteration (every ~1 to 2 weeks)
- Agile emphasizes Test-Driven Development (TDD) to reduce mistakes, written down User Stories to validate customer requirements, Velocity to measure progress

Extreme programming (version of agile programming)

- If short iterations are good, make them as short as possible (weeks vs. years)
- If simplicity is good, always do the simplest thing that could possibly work
- If testing is good, test all the time. Write the test code before you write the code to test.
- If code reviews are good, review code continuously, by programming in pairs, taking turns looking over each other's shoulders.

Agile Then and Now

- Controversial in 2001
    - *"… yet another attempt to undermine the discipline of software engineering… nothing more than an attempt to legitimize hacker behavior." — Steven Ratkin, "Manifesto Elicits Cynicism," IEEE Computer, 2001*
- Accepted in 2013
    - 2012 study of 66 projects found majority using Agile, even for distributed teams

Yes = Plan and Document; No = Agile

- Is specification required?
- Are customers unavailable?
- Is the system to be built large?
- Is the system to be built complex (e.g., real time)?
- Will it have a long product lifetime?
- Are you using poor software tools?
- Is the project team geographically distributed?
- Is team part of a documentation-oriented culture?
- Does the team have poor programming skills?
- Is the system to be built subject to regulation?

Questions