# MIT 850

## Life Cycle and Maturity Models for IT

## Lecture 1

### Traditional processes

22 July 2016

# Admin matters

- All details are available in the study guide on the MIT 850 ClickUP website

## Lecturer


Dr. Patricia E. N. Lutu
**Office:** IT 5-46 **Phone:** +27 12 420 4116
**e-mail:** plutu@cs.up.ac.za **web:** http://www.cs.up.ac.za/~plutu


## Schedule for lectures

| Lecture | Friday Date: | Time | Venue: Tswelopele | Topic(s) |
|---|---|---|---|---|
| Lecture 1 | 22 July | 8:00 - 10:30 | IT 4-64 | Traditional processes |
| Lecture 2 | 26 Aug | 8:00 - 10:30 | IT 4-64 | Agile processes |
| Lecture 3 | 23 Sept | 8:00 - 10:30 | IT 4-64 | Project management |
| Lecture 4 | 21 Oct | 8:00 - 10:30 | IT 4-64 | Quality assurance & maturity models |

# Course overview

## Objectives

- The module aims to introduce the fundamental techniques of **software engineering,**
- **by looking at:**
  - maturity models,
  - software life cycles and methods,
  - standards and procedures to assess and measure quality of processes.

## Study material

- Study material will be made available on ClickUP
- Students are expected to **read and study the prescribed material prior to each lecture.**
- **The aim of lectures** is to consolidate the content of the prescribed material and enable **class discussions**.

# Assessment

- The assessment for this module will consist of theory assignments and a written examination.
- The semester mark will be computed from the assignment marks for the first 3 assignments.
- The final mark will be computed as:
  - **60% semester mark + 40% exam mark.**
- A **sub-minimum of 40%** for the semester mark is required for **exam admission.**
- **Exam:**
  - **'half-open' book: you will be allowed to** prepare an 8 page document that you can consult during the examination.
  - **The 4th assignment** will also be preparation for the exam.

# Assignments

## Individual Assignments

- The specification for each assignment will be made available on ClickUP. **The first 3 assignments will count 60% towards the final mark.**

- If you hand in any assignment within:
  - **1 week after the due date, there will be a penalty of 10%.**
  - **2 weeks after the due date, there will be a penalty of 20%.**

- You are required to **submit the assignments electronically on ClickUp (and NOT by e-mail)**

- **Include the cover page** as specified in the **MIT 2016 brochure** and to adhere to the **plagiarism policy** of the University.

# In this lecture

- **Traditional processes**
  - **Software Engineering**
    - definition, layers, processes, methodologies
  - **Waterfall**
  - **Spiral**
  - **Challenges identified for traditional processes**

| Unit / Topic | Activities | Dates |
|---|---|---|
| **Unit 1 Traditional processes** | Reading | 14 July to 22 July |
| | **Lecture & class discussions** | **22 July** |
| | Work on and submit assignment | 23 July to **8 August** |

# Readings for this lecture

**A. Historical perspective of Software Engineering**

1. Boehm, B.W. (1988) A spiral model of software development and enhancement, *IEEE Computer,* vol. 21, no. 5, pp. 61-72, May 1988.

2. Brooks, F. P. (1987) No silver bullet: essence and accidents of software engineering, *Computer*, vol. 20, no. 4, pp. 10-19, April 1987.

3. Brooks, F.P. (1995) *The Mythical man-month: Essays on Software Engineering*, 20th Anniversary Edition, MA: Addison-Wesley.

4. Royce, W.W. (1970 & 1987) Managing the development of large scale software systems: concepts and techniques, *Proceedings of Wescon*, August, 1970. Also available in *Proceedings of ICSE 9, Computer Society Press*, 1987.

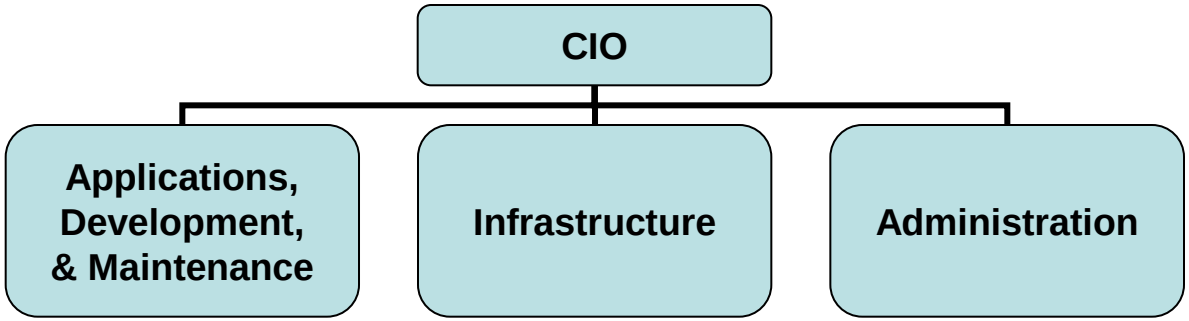
**B. Challenges in Software Engineering**

1. Charette, R.N. (2005) Why software fails, *IEEE Spectrum,* Vol. 42, Issue 9, Sept.2005, pp. 42-49.

2. Dalcher, D. (1999) Disaster in London: The LAS Case study, *Proceedings of the IEEE Conference and Workshop on Engineering of Computer-Based Systems* (*ECBS*) 1999

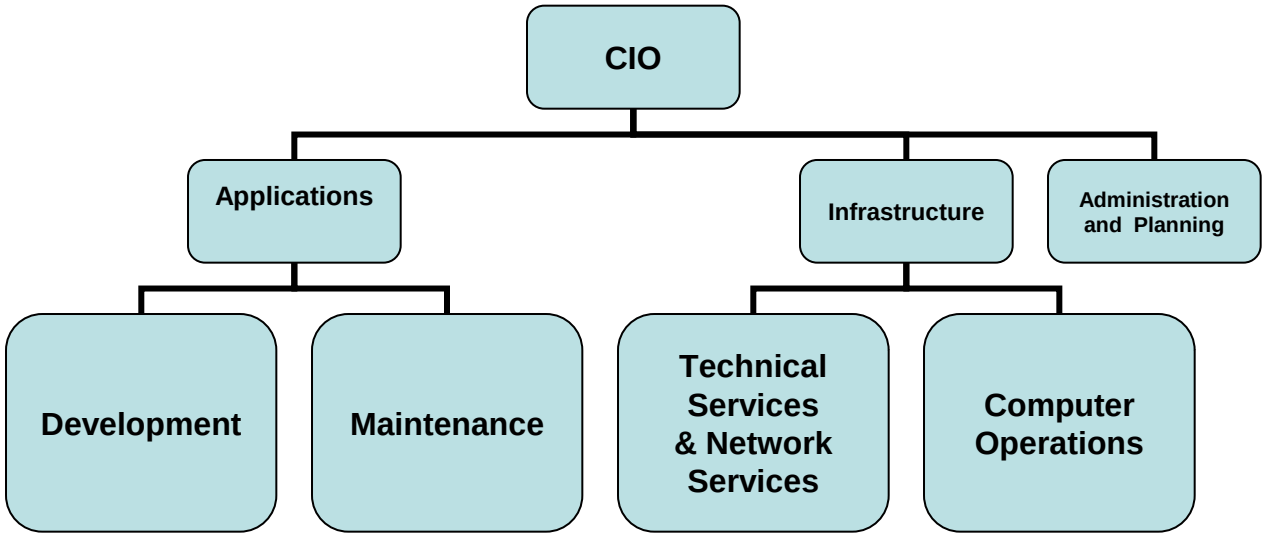
# Basic IT Organization with Administration

Where is software development located in the IT organisation?



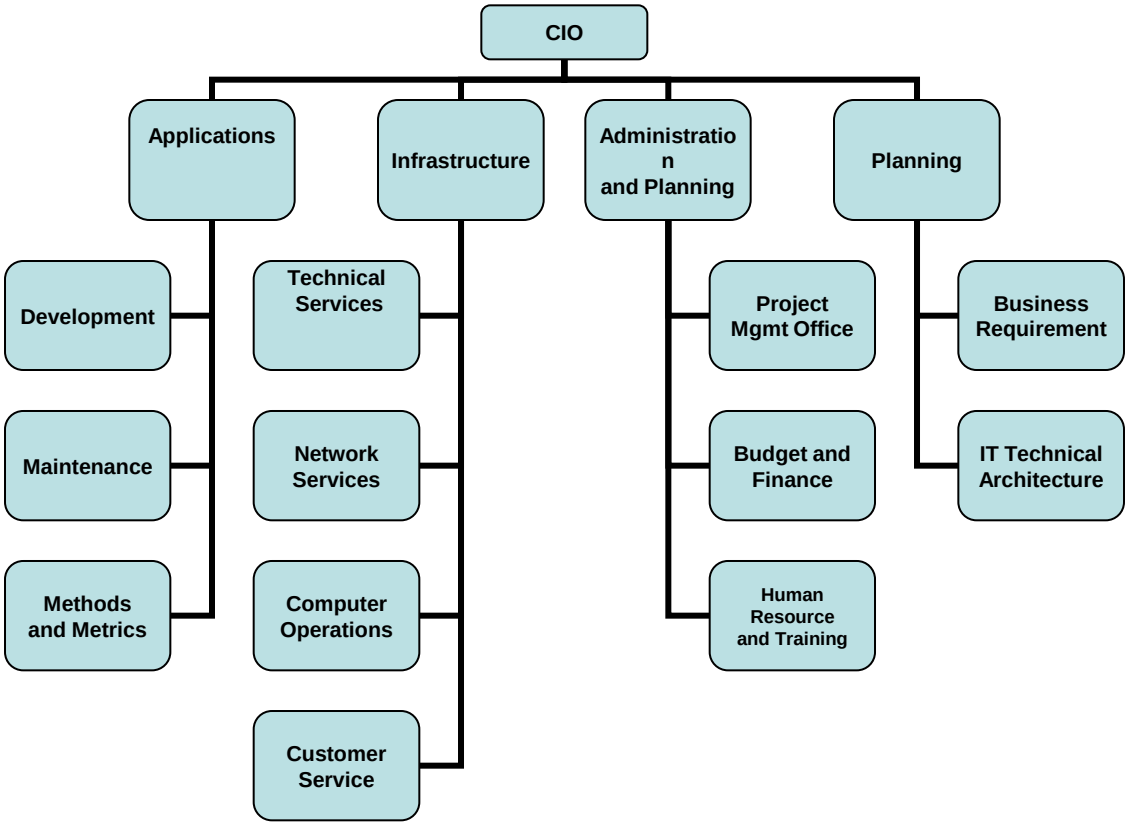
# IT Organization with Dual Management Levels

Where is software development located in the IT organisation?



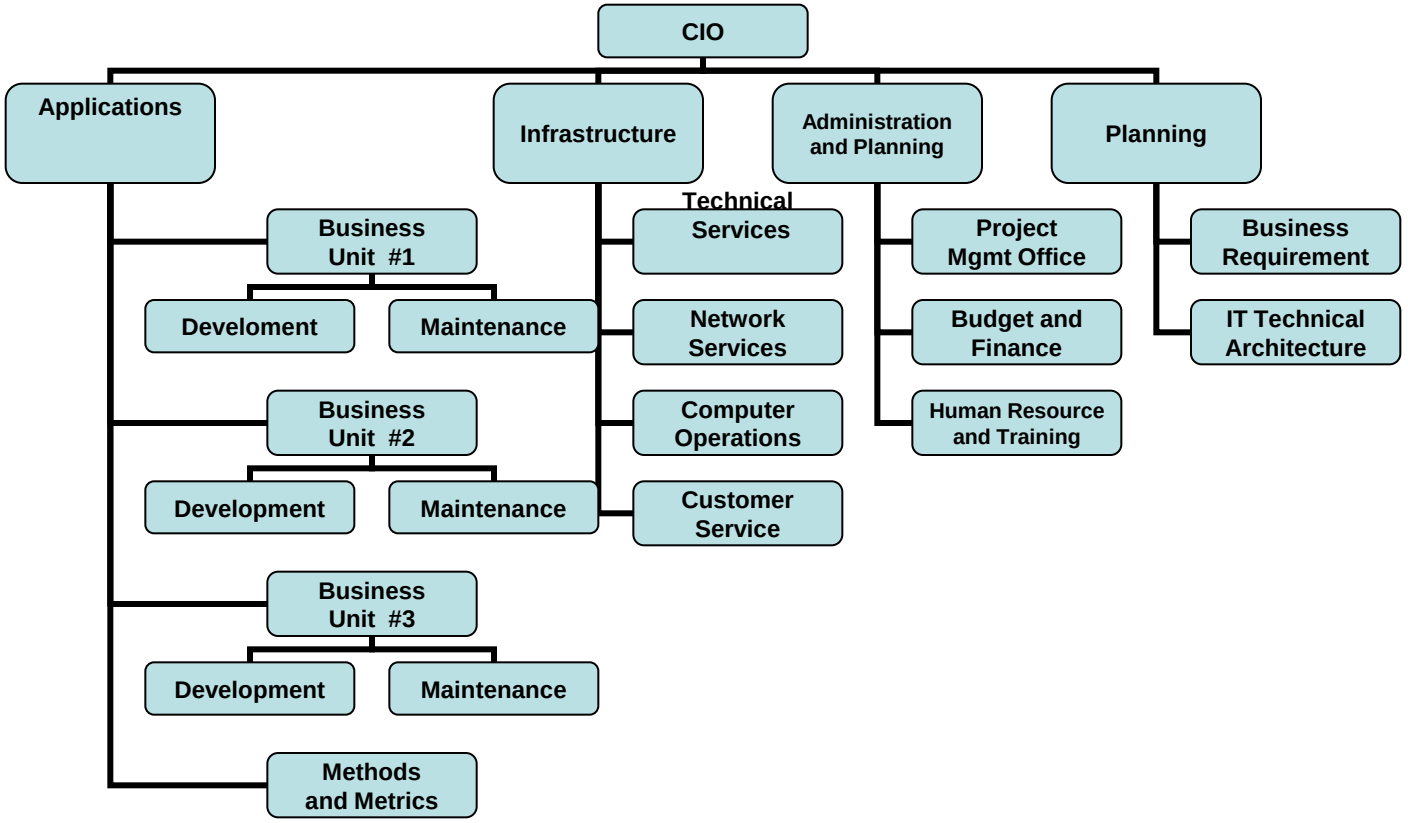
# IT Organization with Three Management Levels

Where is software development located in the IT organisation?



# IT Organization—Three Management Levels with Business Units

Where is software development located in the IT organisation?



# Some definitions of Software Engineering

- **Definition 1 (IEEE in 1993)**

(1) **the application of a systematic disciplined, quantifiable approach** to **the development, operation, and maintenance of software**, i.e. the application of **engineering practices** to software development and

(2) the study of approaches as in (1)

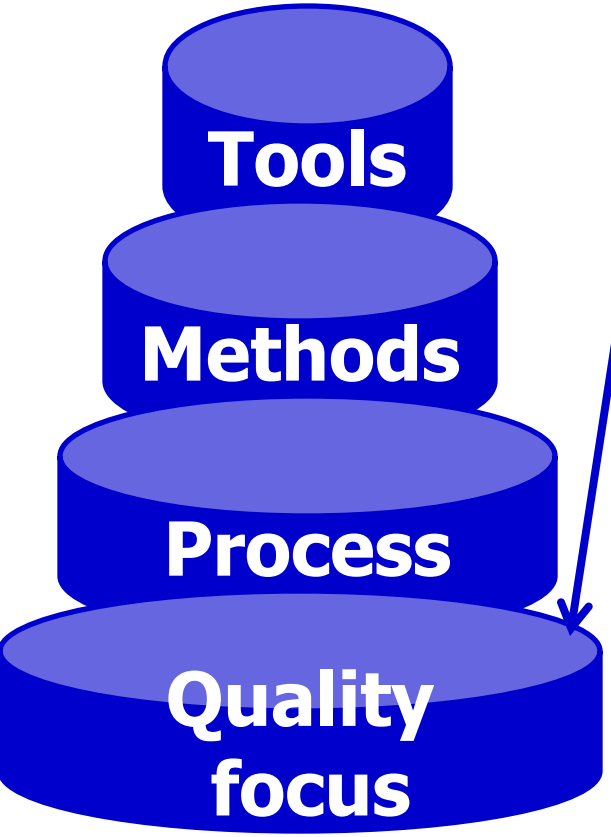
- **Definition 2 (Pressman, 2005)**

..the establishment and use of **sound engineering principles** in order

to **obtain economically, software** that is **reliable** and **works efficiently** on real machines.

# Software Engineering: a layered technology(1)

Pressman (2005)



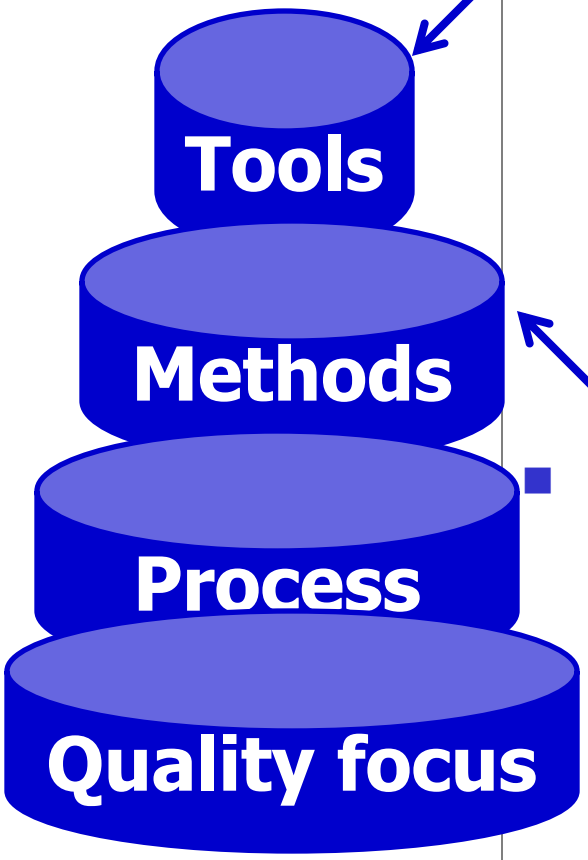

- End result of a SE project is a **product (system)**
- **Quality focus**
  - Activities in layers must focus on producing a **high quality product**
- **What is the meaning of 'quality'?**

  ***Conformance to:***

  - **explicitly stated** functional & performance requirements,
  - **explicitly documented** development standards,
  - & **implicit characteristics** expected of all professionally developed software

# Software Engineering: a layered technology(2)

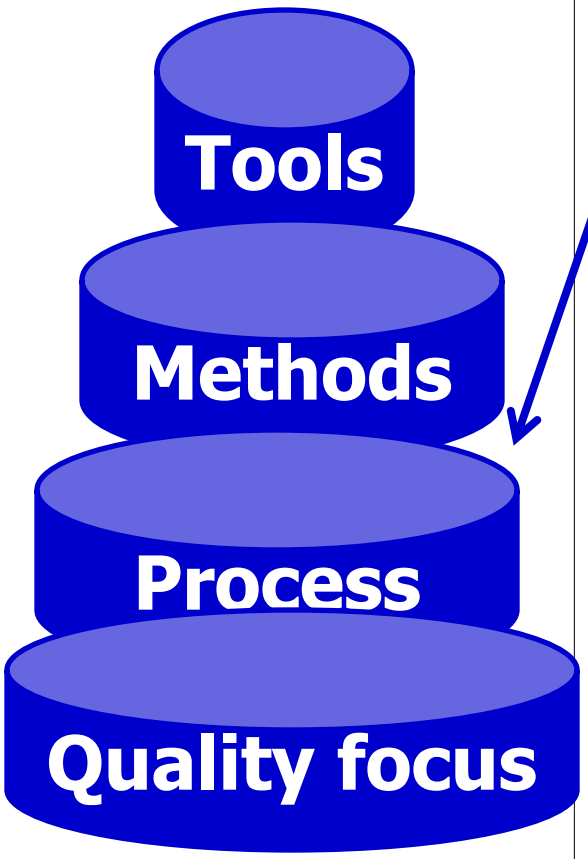
Pressman (2005)




- **Tools**
  - provide automated / semi-automated support methods for the whole process.
    - **e.g. of tools:** UML, requirements tracing, unit testing, integration testing, project management, etc.
- **Methods**
  - **technical 'how-to'** for the SE tasks, e.g. of tasks for:
    - requirements engineering & analysis
    - design
    - program construction etc

# Software Engineering: a layered technology(3)

Pressman (2005)




- **process:** foundation for SE
  - Roadmap / **framework** for tasks required to create timely, high quality software.
  - framework specifies **key process phases** & **activities:**

**(1) generic activities**

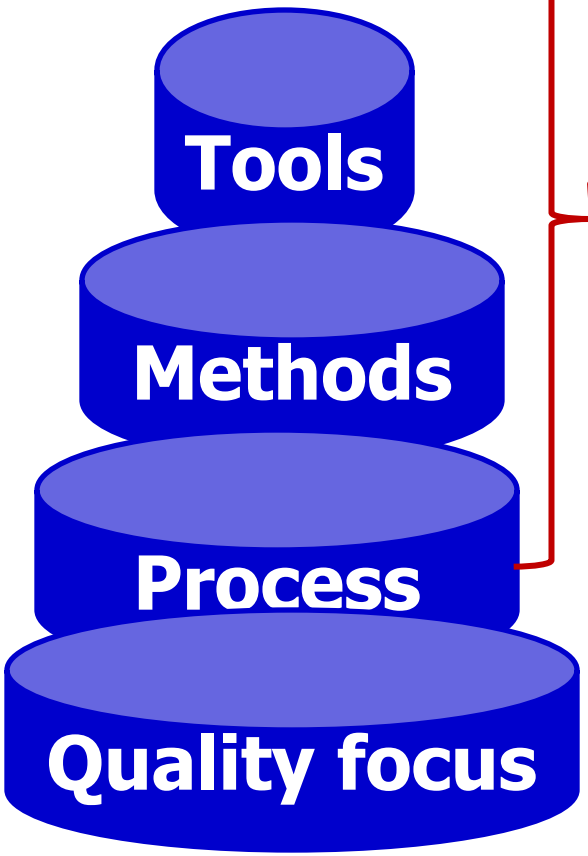
- **(applicable to all software projects)**
- communication, planning,
- modeling, construction, deployment

**(2) Umbrella activities**

- **(for entire software process)**
- tracking, control, risk management, quality assurance, technical reviews, measurement, config. management, etc.

# Software Engineering: a layered technology(4)

Pressman (2005)




- **Process model**

  **= process + methods + tools**

- **Chosen based on:**
  - nature of project & application
  - methods & tools to be used
  - Controls required
  - work products required

- **Three general categories**:
  - Formal processes (e.g. for Aviation)
  - Plan-driven (**e.g. waterfall**, spiral)
  - **Agile** (e.g. XP, scrum, etc.)

# Software development: definitions

## Software process

**(e.g. waterfall, spiral)**

- A software process is **a series of phases of activities** performed to construct a software system.
- Each phase **produces some artifacts** which are the inputs to other phases.
- each **phase has a set of entrance** criteria **&** a set of **exit criteria**

## Software methodology

**(e.g. Object Modeling Technique - OMT)**

- A software methodology **defines the steps** (or how) **to carry out the activities** of a software process.

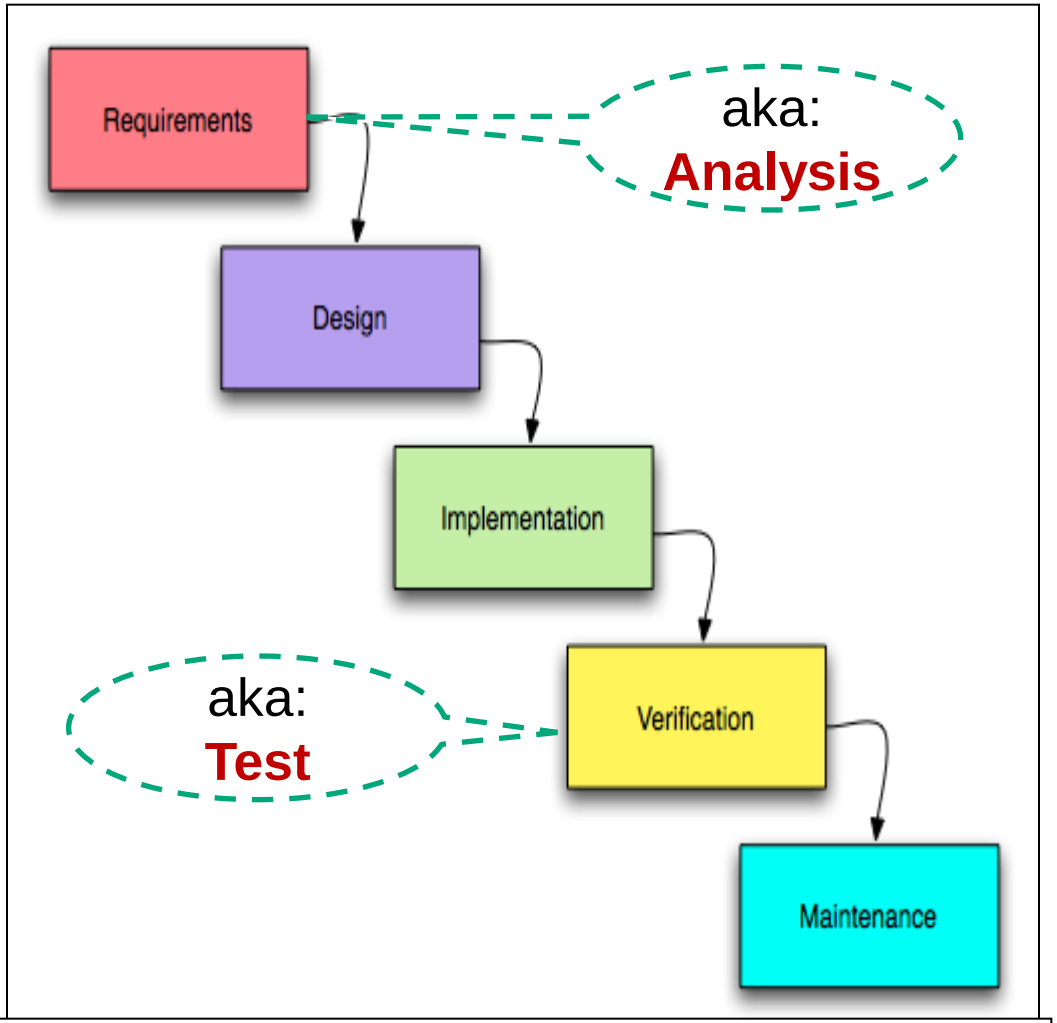
## Software development needs:

- **a software process**
  (defines the what)
  **and**
- **a software methodology**
  (defines the how)
  ( = methods, Pressman)

- **Software process model**
  **= process + methods + tools**
  (Pressman, 2005)

# Overview of Waterfall

- Defined sequence of phases
- One phase is signed off before next is started.
- Outputs of previous phases = input to next.
- Limited iterations.
- Origin: Bennington, 1956 (Developed for Sage defense system (IBM & MIT, 1960).




**Royce, W.W. (1970 & 1987)** Managing the development of large scale softwaresystems: concepts and techniques, *Proceedings of Wescon*, August, 1970. Also available in *Proceedings of ICSE 9,*

# Phases of the waterfall model

1. Requirements specification
2. Design: No separation of architecture & functional design.

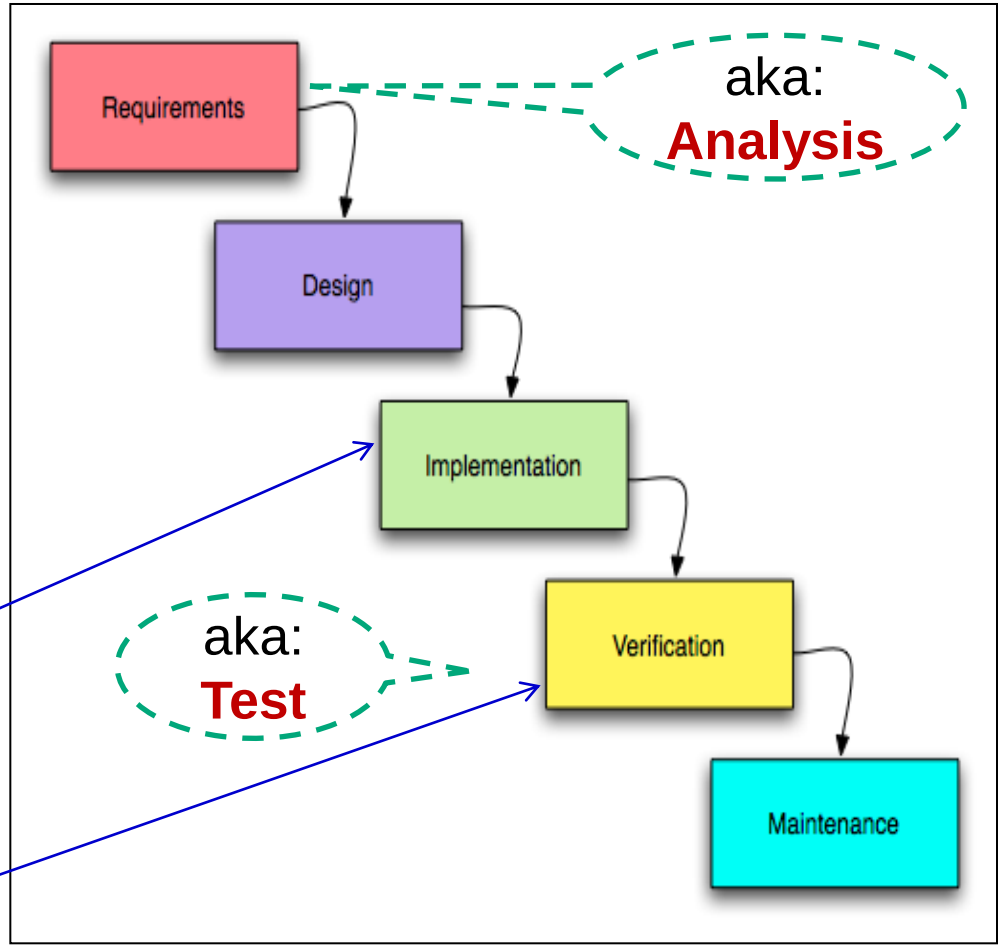
3a. Construction (implementation/coding)

3b. Integration (of higher level components)

4a. Testing & debugging

4b. Installation

5. Maintenance




- Phases executed sequentially
- sign-off between phases

# Strengths of waterfall

**1. Sign-off process:**

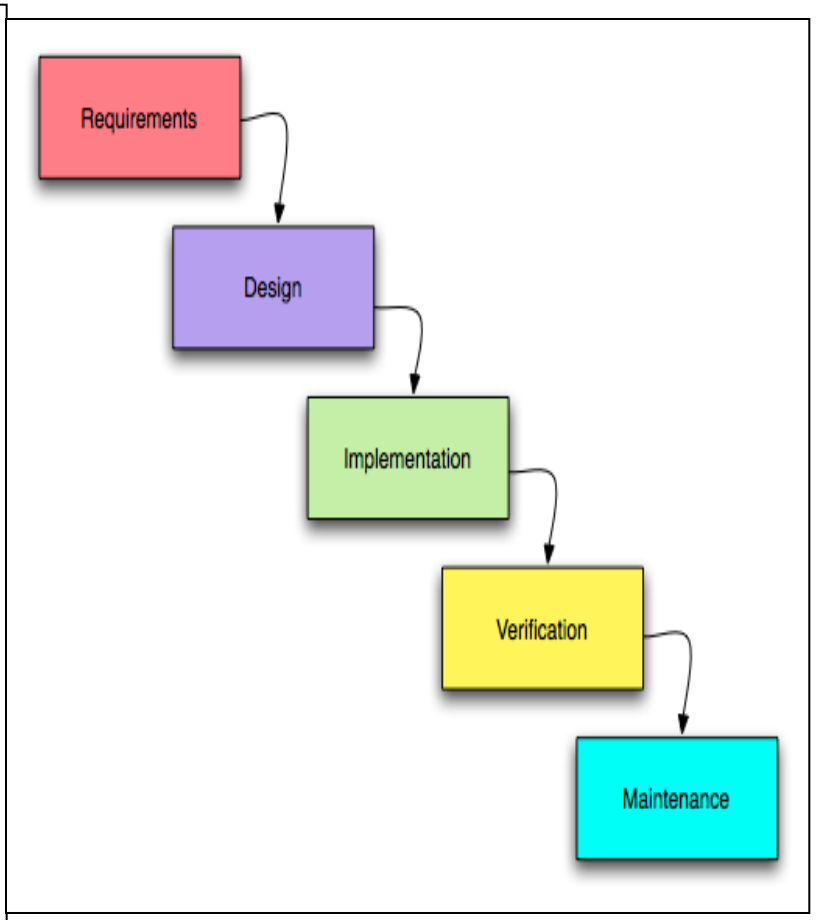
- aims stop cost of errors propagating into design, construction, etc.
- Errors in requirements incur 50 to 200 times more cost than coding errors.

**2. Documentation generated in phases:**

- is valuable asset
- reduces risk & maintenance costs
- provides valuable business information.

**3. Process is simple and well understood.**

**4. Simpler global estimation.**



# Weaknesses of waterfall

**1. Requirements:**

- need to be fully understood up front.
- are static (no provision for changing requirements)

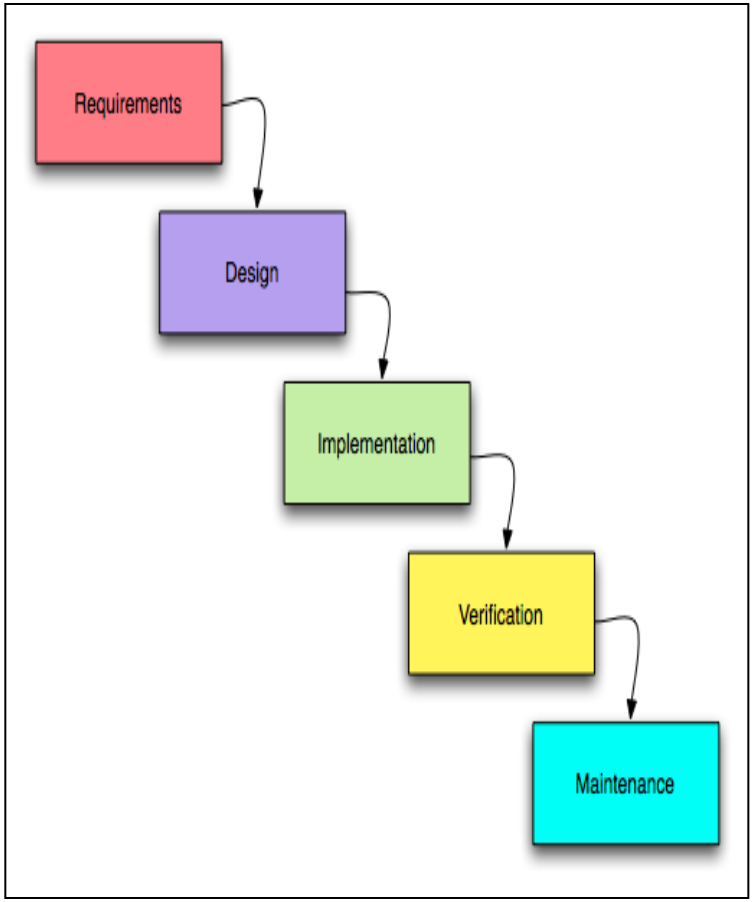
2. Cost implications for implementation & difficulties in requirements

- often only exposed after requirements are signed off.

3. Limited feedback cycles.

4. Difficult estimation if nothing is tested & used yet.

5. No opportunity of early & incremental value extraction.




**Lifecycle could take years**

# Original Spiral model (1)

**Ref: Boehm (1998)**

**Some complaints about waterfall:**

"Stop the life cycle, I want to get off!''

"Life-cycle concept considered harmful"

"The waterfall model is dead."

"No, it isn't, but it should be."

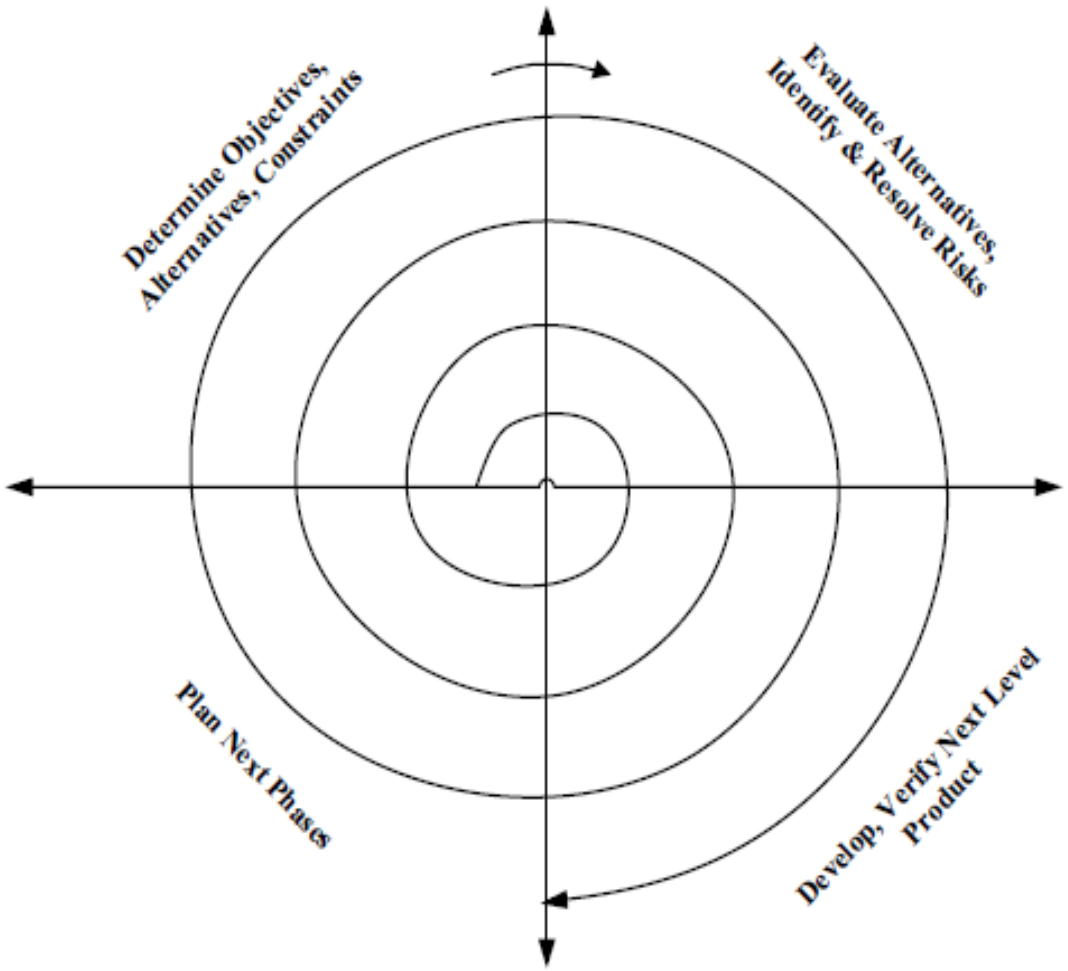



Figure 1. Spiral Software Process Model

# Original Spiral model (2)

**Ref: Boehm (1998)**

Incremental product development in phases

**Four quadrants depict the 4 phases**

**(1) Determine** Objectives, Alternatives, Constraints

**(2) Evaluate** Alternatives, Identify & Resolve Risks

**(3) Develop, Verify** Next Level Product

**(4) Plan** Next Phases

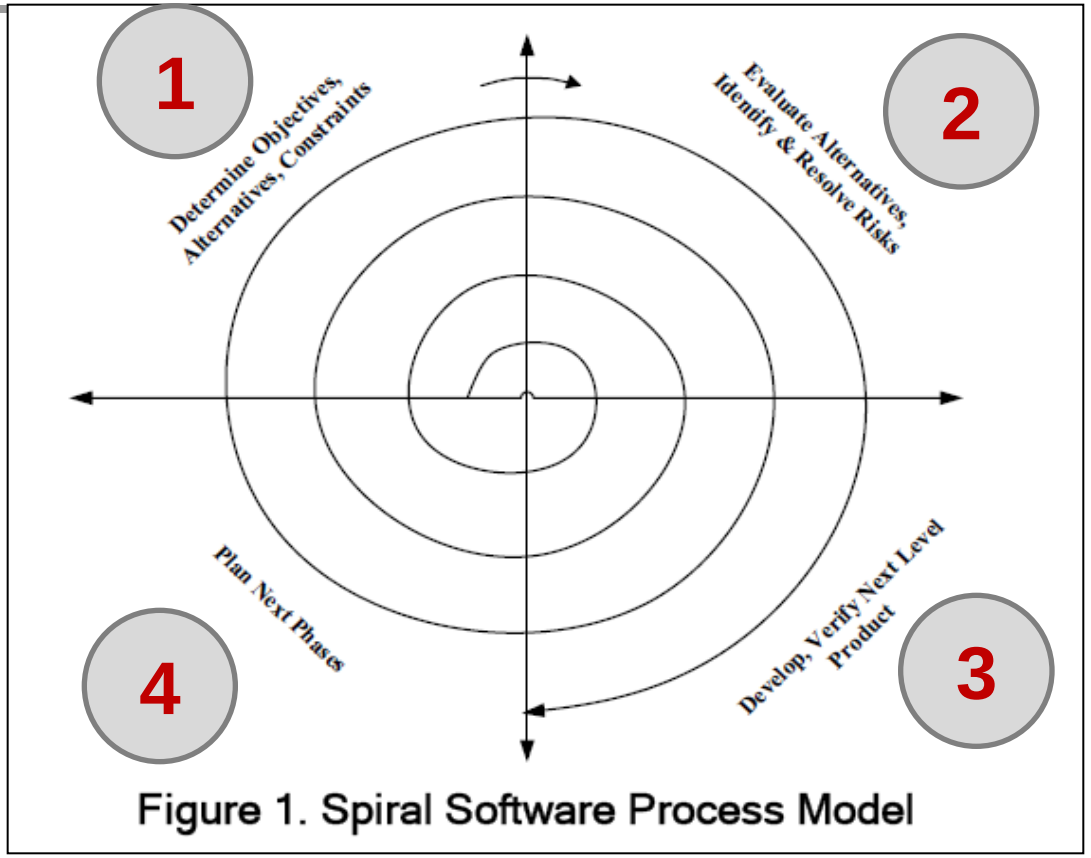


Figure 1. Spiral Software Process Model

Iterative & incremental development

# Some software project failures

Some project failures between 1992 and 2005

**(Ref: Charette (2005, pg. 44)**

- Oxford health plans – 1997
  - financial losses of approx. $4 billion
- Sydney Water Corp – 2002
  - financial loss of approx. $33 million
- London Stock exchange - 1993
  - financial loss of approx. $600 million
- London Ambulance Service – 1993

# Why software fails:

- 12 Reasons: from 1992 to 2005 (Ref: Charette (2005, pg. 45)

1. Unrealistic or unarticulated project goals
2. Inaccurate estimates of needed resources
3. Badly defined system requirements
4. Poor reporting of the project's status
5. Unmanaged risks
6. Poor communication among customers, developers and users
7. use of immature technology
8. Inability to handle the project's complexity
9. Sloppy development practices
10. Poor project management
11. Stakeholder politics
12. Commercial pressures

# Class exercise 2 (Solutions to be handed in)

**a. Refer to the paper: Why software fails**

i. Identify any three important points the paper makes?

ii. What are the major consequences of software failures?

**b. Refer to the LAS case study:**

i. Identify one problem in each of the generic process framework activities of *communication, planning, modeling, construction, deployment*.

ii. Identify one problem in each of the umbrella activities of *tracking, control, quality assurance, technical reviews, configuration management.*

iii. Give at least two examples of un-ethical behaviour in the case study.

# Historical perspective: No silver bullet (1)

- Paper by Brooks, F.P (1987) – 29 years ago:

  Brooks, F. P. (1987) No silver bullet: essence and accidents of software engineering, *Computer*, vol. 20, no. 4, pp. 10-19, April 1987.

- Number of citations to date: 546 (CiteSeer)

# Historical perspective: No silver bullet (2)

- Paper by Brooks, F.P (1987) – 29 years ago
- Number of citations to date: 546 (CiteSeer)

'Of all the **monsters that fill the nightmares** of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. For these, one seeks bullets of silver that can magically lay them to rest.

The ..**software project** has something of this character. It is usually innocent and straightforward, but it is capable of becoming a monster of missed schedules, blown budgets and flawed products...'

'So we hear desperate cries for a **silver bullet** – something to make the software costs drop as rapidly as ....'

# Historical perspective: No silver bullet (3)

- Significance of the term **'silver bullet'** to software engineering?
- Why were there no silver bullets in 1987 and there are still no silver bullets to date?
- **Answer:**
  - Two major types of **difficulties** in software technology
    - **(1) (due to) essence**
    - **(2) (due to ) accidents**

# Historical perspective: No silver bullet (4)

- **What are difficulties due to essence?**
  - The difficulties inherent in the nature of the technology
- **What is the 'essence' of software?**
  - Essence of a software entity is a construct of **interlocking components:**
    - **Data sets**
    - **Relationships among data items**
    - **Algorithms**
    - **Invokation of functions**
- What are the causes of the difficulties due to essence?
  - **Complexity, Conformity, Changeability, Invisibility**

# No silver bullet: essence (5)

- **Complexity**
  - **many possible states** => conceiving, describing, testing are hard
  - **Scaling is not a repetition of same elements** => increase in number of different & interacting elements
  - Difficulties are discussed on pg. 11 of Brook's paper
- **Conformity**
  - Software must conform to many diverse interfaces: other software, human institutions & systems
- **Changeability**
  - manufactured things get superceded by better models **BUT** all successful software gets changed
- **Invisibility:** software (in execution) cannot be visualised

# Historical perspective: No silver bullet (6)

- **What promising silver bullets (in 1987)?**
  - Buy vs build
  - **requirements refinement & rapid prototyping**
  - **incremental development (grow – don't build)**
  - great designers

# Historical perspective: No silver bullet (6)

- Two major types of difficulties in software technology
  - (1) essence **(2) accidents**

- **What are 'accidents'?**
  - The **difficulties** that attend the production of the technology, but are not inherent.

  - **What are the difficulties?**
    - **timely production**
    - **production of reliable software**

- **What promising silver bullets (in 1987)?**
  - HLLs, OOP, Expert Systems, Automatic programming, Graphical programming, environments and tools (focus is on construction)

# Major sources of SPM challenges (1)

**Paper 1: Brooks (1975 ) The mystical man-month**

*Restaurant Antoine (Fondé en 1840)*
*AVIS AU PUBLIC*
*Faire de la bonne cuisine demande un certain temps.*
*Si on vous faire attendre,*
*c'est pour mieux vous servir, et vous plaire.*

*Restaurant Antoine (established in 1840)*
*NOTICE TO THE PUBLIC*
*Good cooking takes time.*
*If you are made to wait,*
*It is to serve you better, and to please you.*

# Major source of SPM challenges (2)

Paper 1: **Brooks (1975)** The mystical man-month aka the MMM paper.

'More software projects have gone awry for lack of calendar time than for all other causes combined. Why is this cause of disaster so common?'

**In the period 1975 to 1998 the MMM paper:**

- **was cited 522 times**
- in 50 disciplines
- in different languages
- by authors in 28 countries

**So, what is a man-month?**

# Major source of SPM challenges (3)

'More software projects have gone awry for lack of calendar time than for all other causes combined. Why is this cause of disaster so common?'

- **Reasons** (**Brooks (1975)** paraphrased by Verner et al. 1999):
  1. Existing techniques of estimating are poorly developed (all will go well)
  2. Existing techniques of estimating confuse effort with progress
     - hidden assumption is that **men & months are interchangeable.**

# Major source of SPM challenges (4)

'More software projects have gone awry for lack of calendar time than for all other causes combined. Why is this cause of disaster so common?'

- **Reasons** (Brooks (1975) paraphrased by Verner et al. 1999):
  3. We are uncertain of our estimates & software project managers do not stubbornly support estimates
     - Like the chef at Restaurant Antoine
  4. schedule progress is poorly monitored
     - standard techniques used in other engineering disciplines are considered radical in SE
  5. When schedule slippage is recognised (mostly late in the schedule) the response is to add manpower
     - This makes the problem worse because **adding manpower late in the project makes it later (Brook's law)**

# Home work

1. What is 'essence'?
   - The difficulties inherent in the nature of the technology are due to:
     - **Complexity, Conformity, Changeability, Invisibility**
   - What promising silver bullets for essence(in 2016)?

2. What are 'accidents'?
   - The **difficulties** that attend the production of the technology, but are not inherent are:
     - **timely production , production of reliable software**
   - What promising silver bullets for accidents(in 2016)?

3. Why is software engineering very challenging?