

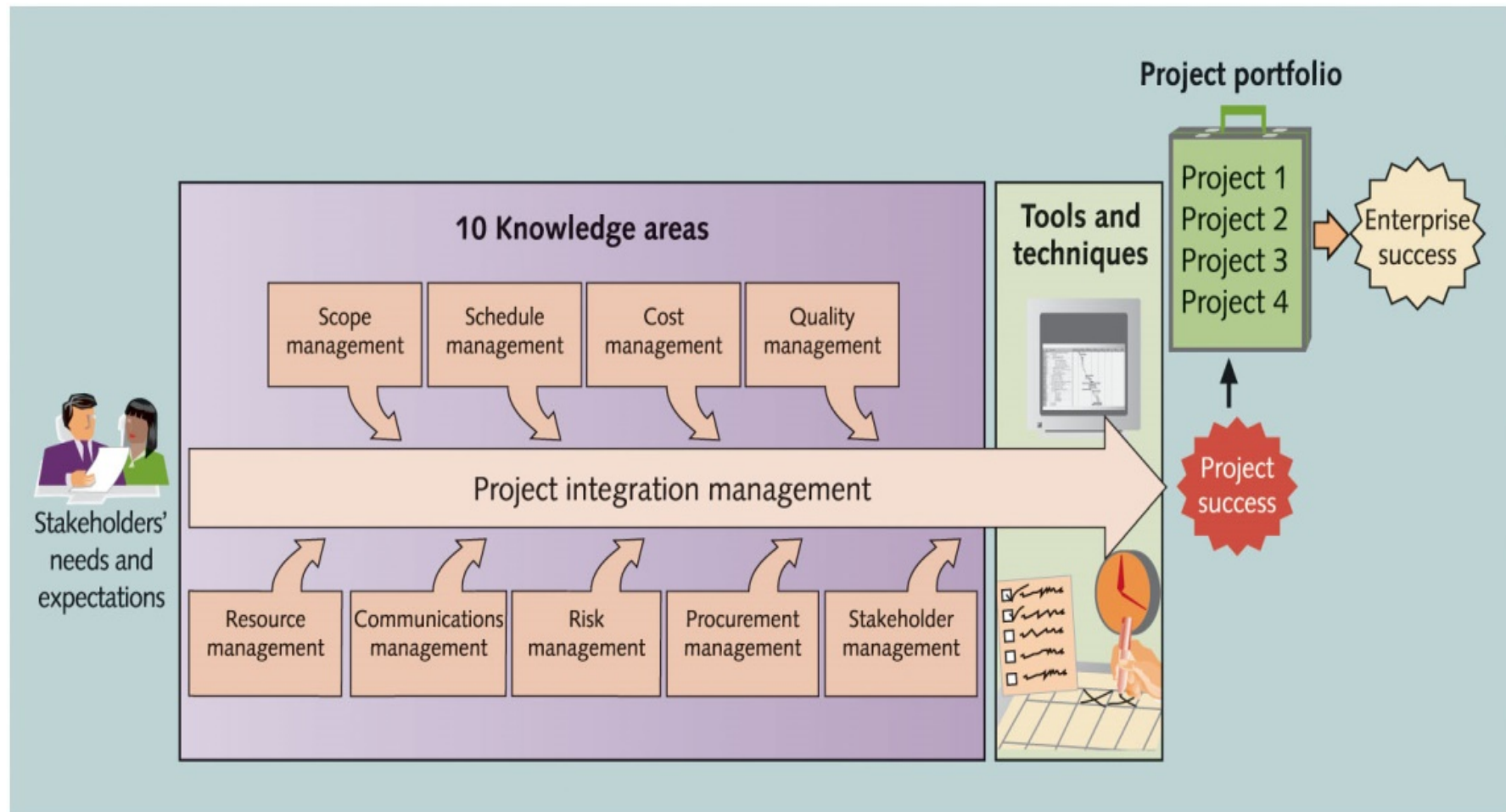
# CSIT214/CSCI814/HCSC814

## IT Project Management



### Quality management

# Project management framework (review)



**FIGURE 1-2** Project management framework

# “Software eats the World”

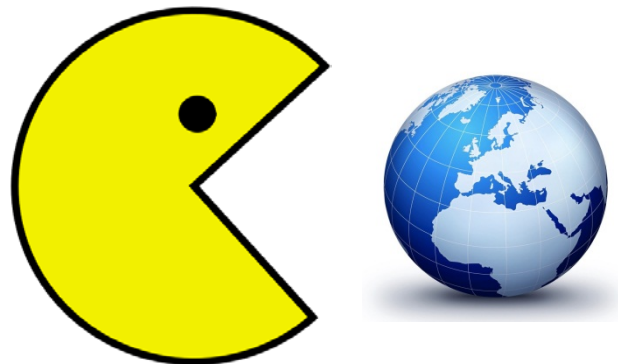
---

- *“We are in the middle of a dramatic and broad technological and economic shift in which software companies are poised to take over large swathes of the economy”*

(Marc Andreessen, “Why Software is Eating the World”,  
**The Wall Street Journal**,

<http://online.wsj.com/article/SB10001424053111903480904576512250915629460.html>)

- More and more major businesses and industries are being run on software and delivered online services.



# “Software eats the World” (cont.)

---

Good news for us ....

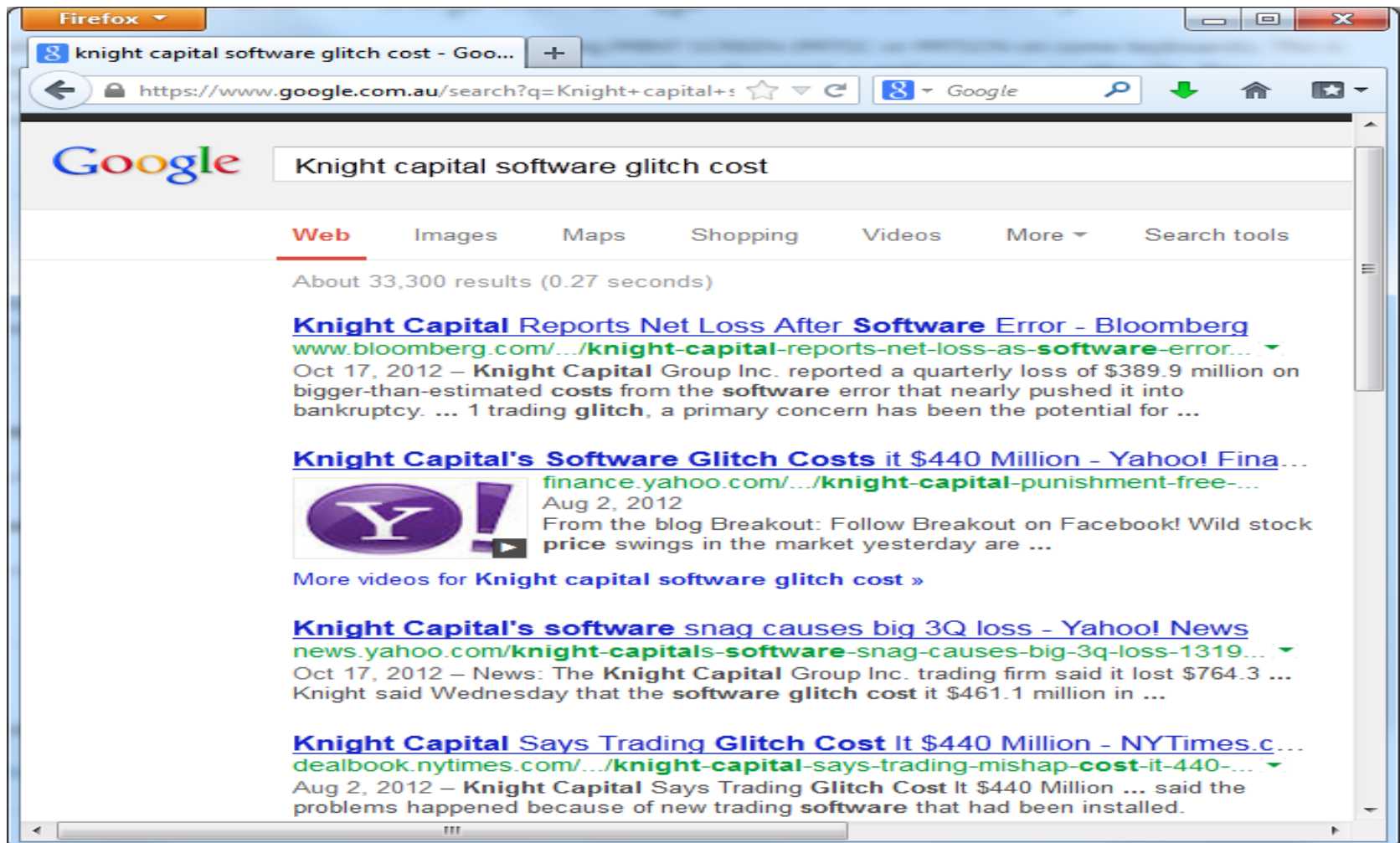


But "*With great power  
there must also come  
great responsibility*" ...



# Low-quality software costs jobs ...

<https://www.youtube.com/watch?v=7BKNnpJfWII>



# Low-quality software costs money ...

---

On the 4th June 1996 at 1233 GMT (UTC) the European Space Agency launched a new rocket, **Ariane 5**, on its maiden unmanned flight,

<https://www.youtube.com/watch?v=kYUrqdUyEpl>

“It took the European Space Agency **10 years** and **\$7 billion** to produce **Ariane 5**, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a **small computer program** trying to stuff a 64-bit number into a 16-bit space”

Source: <http://www.around.com/ariane.html>

# Low-quality software costs lives


Firefox

W Therac-25 - Wikipedia, the free ...

en.wikipedia.org/wiki/Therac-25

Google

Create account Log in



WIKIPEDIA  
The Free Encyclopedia

[Main page](#)  
[Contents](#)  
[Featured content](#)  
[Current events](#)  
[Random article](#)  
[Donate to Wikipedia](#)

Interaction

[Help](#)  
[About Wikipedia](#)  
[Community portal](#)  
[Recent changes](#)  
[Contact Wikipedia](#)

[Toolbox](#)  
[Print/export](#)

ArticleTalk

ReadEditView history

## Therac-25

From Wikipedia, the free encyclopedia

The **Therac-25** was a [radiation therapy machine](#) produced by [Atomic Energy of Canada Limited](#) (AECL) after the [Therac-6](#) and [Therac-20](#) units (the earlier units had been produced in partnership with [CGR of France](#)).

It was involved in at least six accidents between 1985 and 1987, in which patients were given massive [overdoses of radiation](#), approximately 100 times the intended dose.<sup>[2]<sup>425</sup></sup> These accidents highlighted the dangers of software [control](#) of safety-critical systems, and they have become a standard case study in [health informatics](#) and [software engineering](#).

**Contents** [\[hide\]](#)

- 1 Problem description
- 2 Root causes
- 3 See also
- 4 Notes
- 5 External links

### Therac 25 user interface<sup>[1]</sup>

PATIENT NAME	: JOHN DOE		
TREATMENT MODE	: FIX	BEAM TYPE: X	ENERGY (MeV) : 25
	ACTUAL	PREScribed	
UNIT RATE/MINUTE	0	200	
MONITOR UNITS	50 50	200	
TIME (MIN)	0.27	1.00	
GANTRY ROTATION (DEG)	0.0	0	VERIFIED
COLLIMATOR ROTATION (DEG)	359.2	359	VERIFIED
COLLIMATOR X (CM)	14.2	14.3	VERIFIED
COLLIMATOR Y (CM)	27.2	27.3	VERIFIED
WEDGE NUMBER	1	1	VERIFIED
ACCESSORY NUMBER	0	0	VERIFIED

# Questions of the day

---

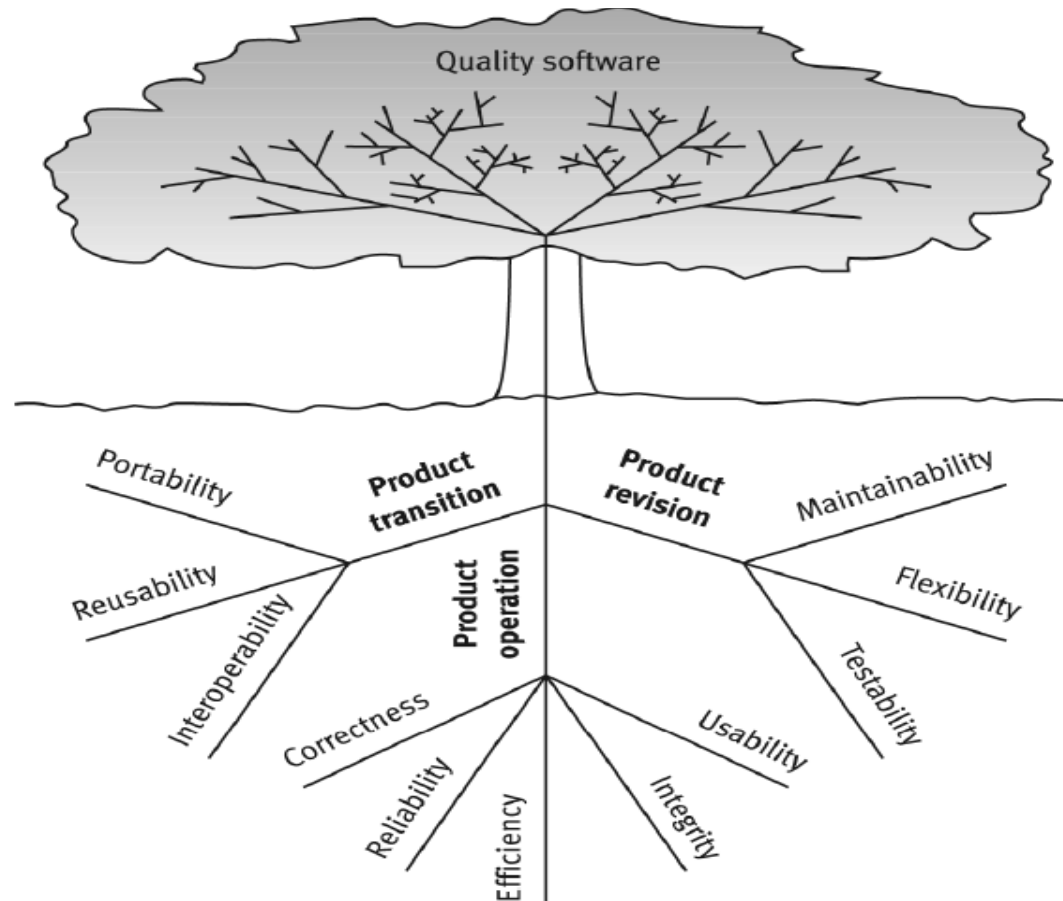


- ❑ What do you think defining the quality of a software?
- ❑ How do we deliver a high-quality software?



# Characteristics of software quality

---



McCall's Quality Factor Tree

# ISO 9126

---

- ❑ ISO 9126 “Software engineering — Product quality” is an international standard for the evaluation of software quality.
- ❑ Attributes of software product quality
  - External qualities: reflect the external view of software held by users.
  - Internal qualities: reflect the developers’ view of the software.

# ISO 9126 (external) software qualities

---

functionality	does it satisfy user needs?
reliability	can the software maintain its level of performance?
usability	how easy is it to use?
efficiency	relates to the physical resources used during execution
maintainability	relates to the effort needed to make changes to the software
portability	how easy can it be moved to a new environment?

## Sub-characteristics of **Functionality**

---

- ❑ Suitability
  - Providing an appropriate set of functions for specified tasks and user objectives.
- ❑ Accuracy
  - Providing the correct or agreed results.
- ❑ Interoperability
  - ability of software to interact with other software components
- ❑ Security
  - Protect information and data and control access to the system.

## Sub-characteristics of **Reliability**

---

### ❑ Maturity

- frequency of failure due to faults

### ❑ Fault-tolerance

- maintain a specified level of performance and recover the data directly affected in the case of a failure.

### ❑ Recoverability

- re-establish a specified level of performance and recover the data directly affected in the case of a failure.

## Sub-characteristics of Usability

---

- ❑ Understandability
  - easy to understand?
- ❑ Learnability
  - easy to learn?
- ❑ Operability
  - easy to use?
- ❑ Attractiveness
  - capability of the software product to be attractive to the user

## Sub-characteristics of **Efficiency**

---

- Time behaviour
  - e.g. response time
- Resource utilization
  - e.g. memory usage

## Sub-characteristics of **Maintainability**

---

- ❑ Analysability
  - identify the cause of a failure or parts to be modified.
- ❑ Changeability
  - how easy is software to change?
- ❑ Stability
  - avoid unexpected effects from modifications of the software.
- ❑ Testability
  - The capability of the software product to enable modified software to be validated.



## Sub-characteristics of **portability**

---

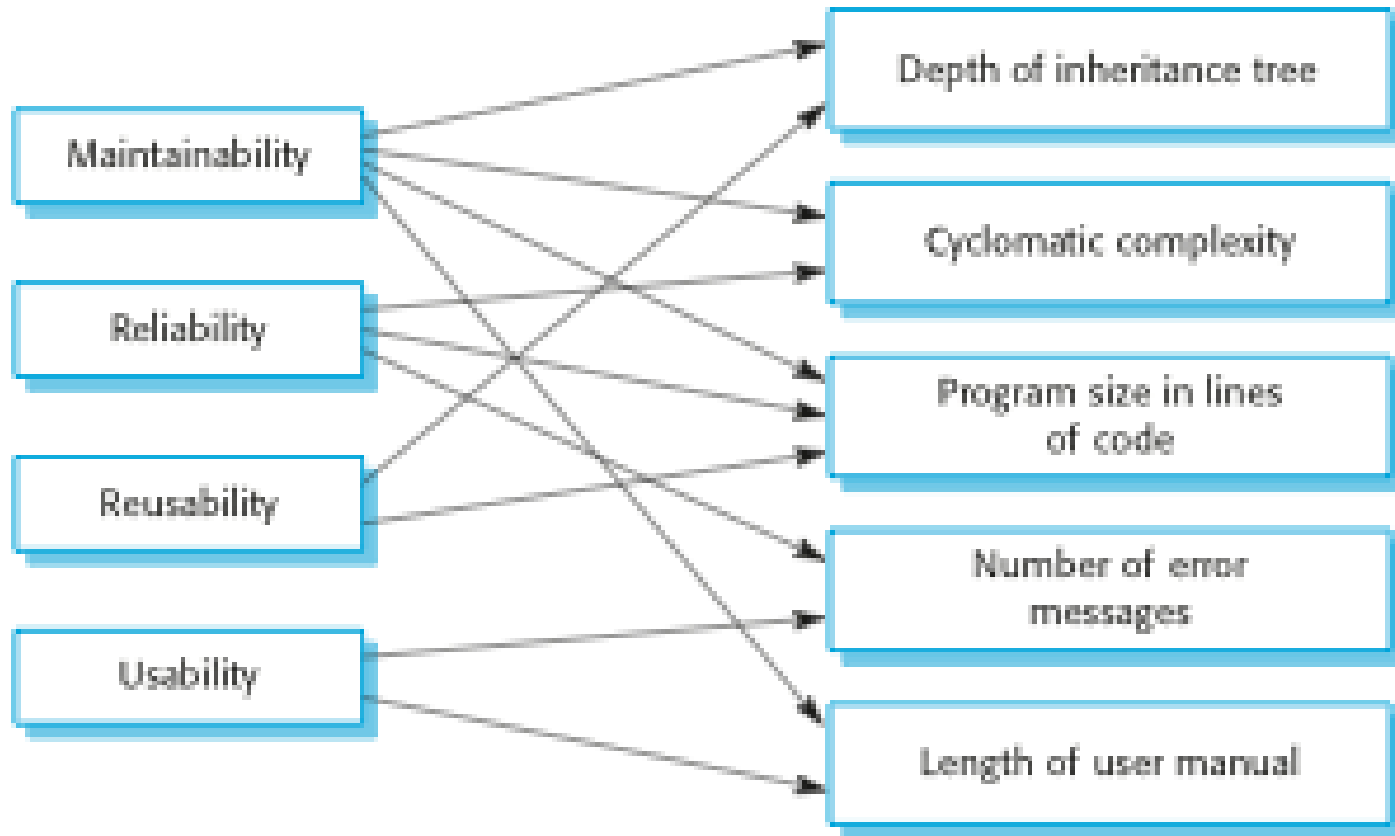
- ❑ Adaptability
  - adapted for different specified environments
- ❑ Installability
  - installed in a specified environment
- ❑ Co-existence
  - Capability of co-existing with other independent software products in a common environment sharing common resources.
- ❑ Replaceability
  - to be used in place of another specified software product for the same purpose in the same environment.

# Relationships between internal and external quality

---

## External quality attributes

## Internal attributes



# Using ISO 9126 quality standards

---

- Judge the importance of each quality for the application
  - for example, safety critical systems - *reliability* very important
  - real-time systems - *efficiency* important
- Select relevant external measurements within ISO 9126 framework for these qualities, for example
  - for reliability
    - Availability: the percentage of a particular time interval that a system is usable.
    - Mean-time between failures: the total service time divided by the number of failures
  - response-time for efficiency

## Using ISO 9126 quality standards

---

- map measurement onto ratings scale to show degree of user satisfaction – for example response time

<b>response (secs)</b>	<b>rating</b>
<2	Exceeds requirement
2-5	Target range
6-10	Minimally acceptable
>10	Unacceptable

# Pen and paper exercise

---

The new tutor management system has been installed, and is normally available to users from 8:00am until 6:00pm from Monday to Friday.

Over a four week period, the system was unavailable for one whole day because of problems with a disk drive and was not available on two other days until 10:00am because of problems with overnight batch processing runs.

*What are the availability and the mean time between failures of the software system?*

# Pen and paper exercise

---

The following is an excerpt from a report generated from a help-desk logging system.

Assess the maintainability of module AA247 from the point of view of:

- the user management;
- the developer management.

*Assume that the **mean effort to repair** and the **mean elapsed time to repair** for a software with high maintainability is **4 hours** and **7 days** respectively.*

Module	Date fault reported	Fault corrected	Effort (hours)
AA247	1.4.2004	2.4.2004	5
AA247	10.4.2004	5.5.2004	4
AA247	12.4.2004	5.5.2004	3
AA247	6.5.2004	7.5.2004	2

# How to deliver high-quality software?

---

## □ Testing it hard?

- Testing can only tackle **known unknowns**.

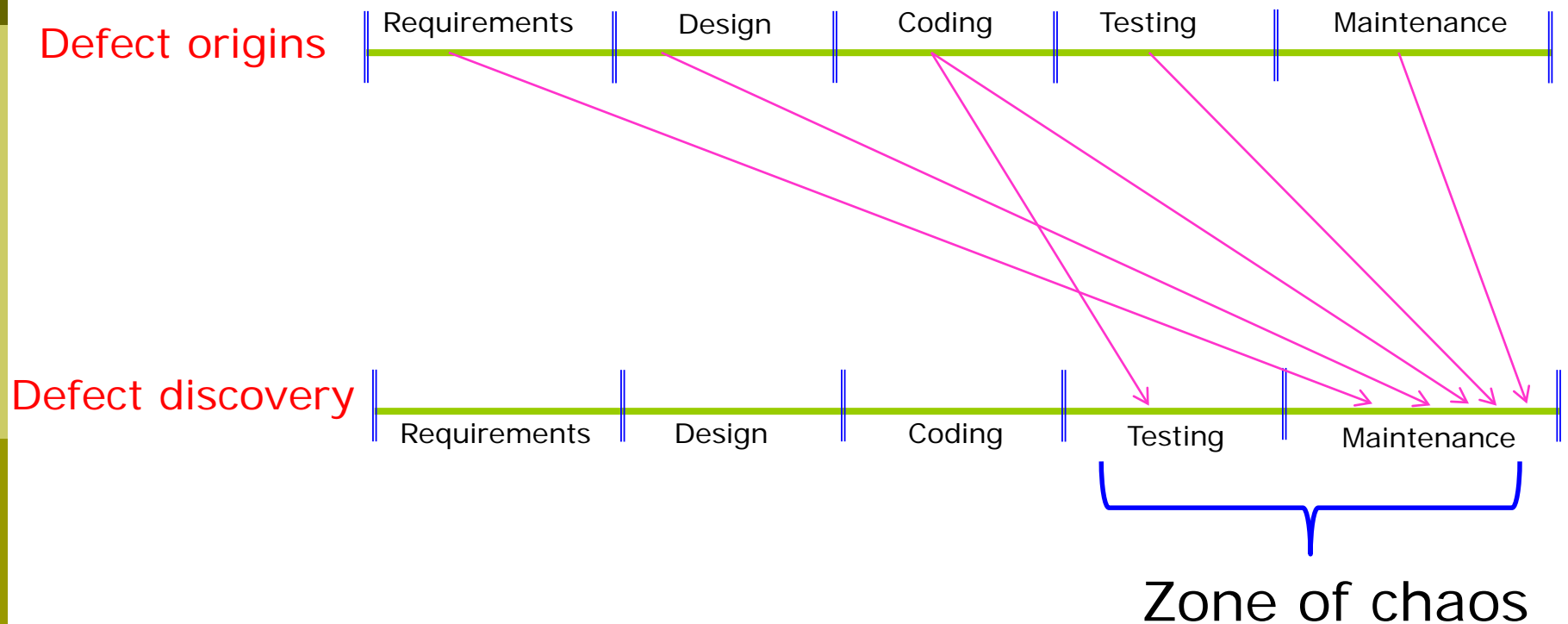
- If you don't know what you're testing for, you are not, by definition, conducting tests.

## □ Software by its very nature is subject to **unknown unknowns**.

- A software (e.g. its functionalities, behaviour, environment, etc.) constantly changes.

# Defect origins

---





# Product vs. Process Quality Management

---

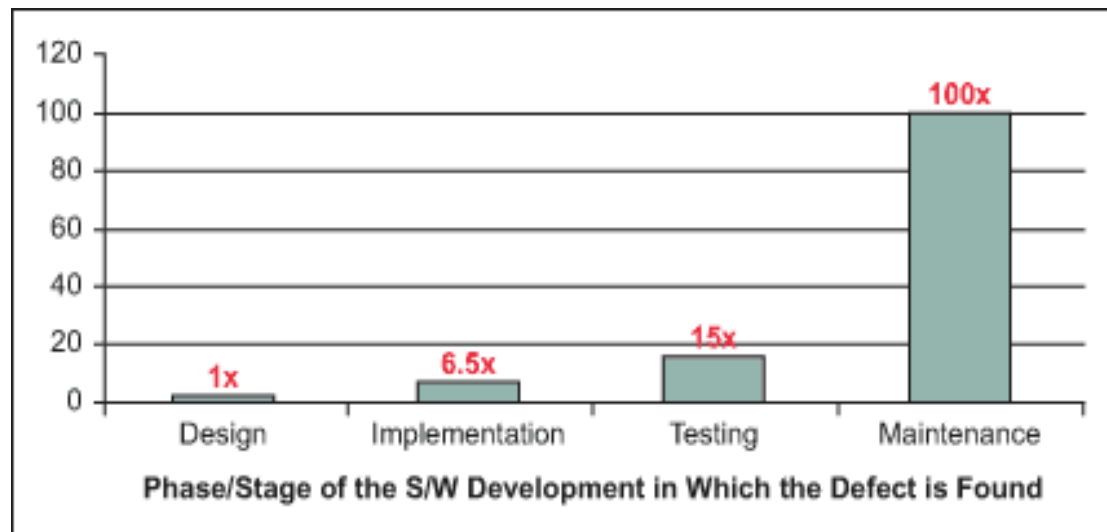
- ❑ The quality measurements described earlier are related to products.
    - It is difficult to use them to measure the quality of a product when it is not completed yet.
  - ❑ What is an alternative approach to ensure quality in software?
- ⇒ Process Quality Management

Quality of a product is heavily dependent on quality of the process which produces it.

# Process Quality Management

---

- ❑ The system development process comprises a number of (sub-)activities .
- ❑ Errors can enter the process at any stage.
- ❑ Errors not removed at early stages become more expensive to correct at later stages.



# Process Quality Management (cont.)

---

- ❑ **Errors** should therefore be **identified and removed** by careful examination of the deliverables of each activity before they are passed on to the next activity.
- ❑ One way of doing this is by having the following **process/activity requirements** for each step.

# For each activity

---

## □ Entry requirements

- these have to be in place before an activity can be started
- example: 'a comprehensive set of **test data** and **expected results** be prepared and independently reviewed against the system requirement before **testing** can commence'

For each activity, *define*

---

- Implementation requirements
  - these define how the activity is to be conducted
  - example 'whenever an error is found and corrected, ***all test runs*** must be completed, including those previously successfully passed'

For each activity, *define*

---

□ Exit requirements

- an activity will not be completed until these requirements have been met
- example: 'the **testing** phase is finished only when *all tests have been run in succession with no outstanding errors*'

# Q&A time

---

- ▣ What might be the entry, implementation, and exit requirements for the activity *coding*?