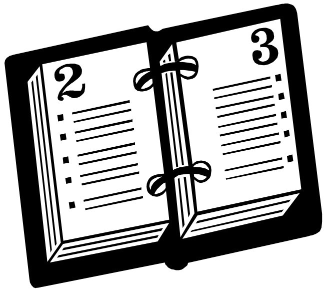
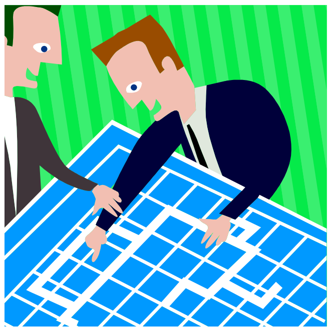
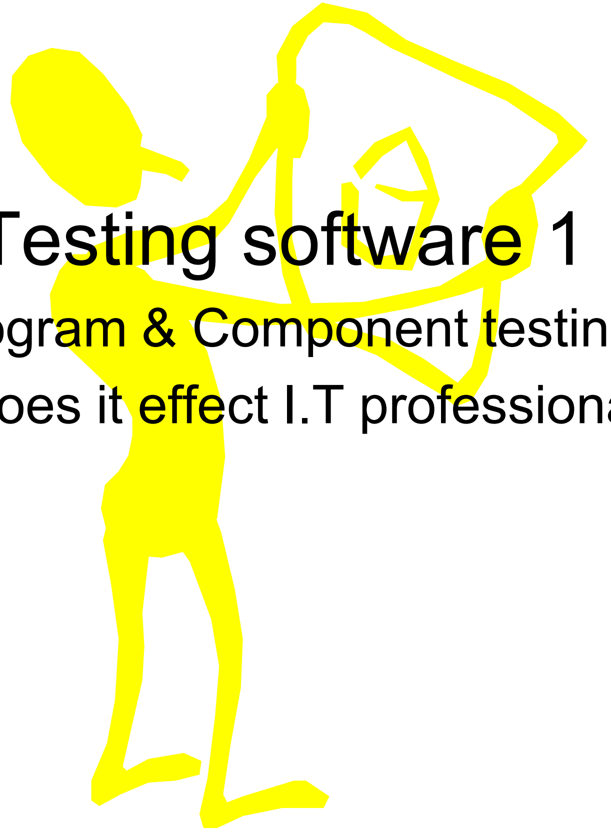
# INFO5990 Professional Practice in IT

Lecture 08 A & B





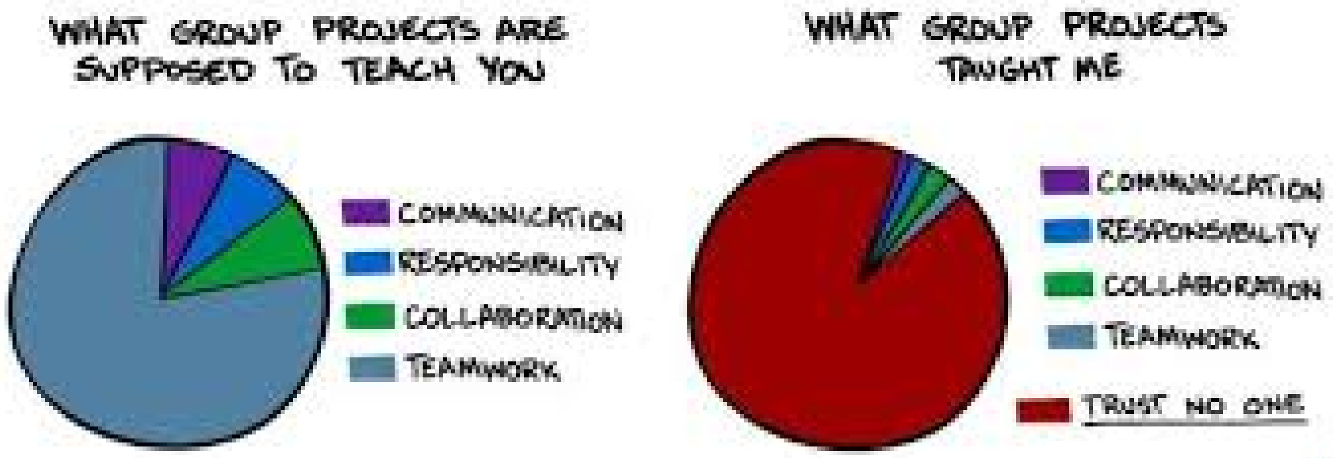


## Testing software 1

Program & Component testing

How does it effect I.T professionals ?

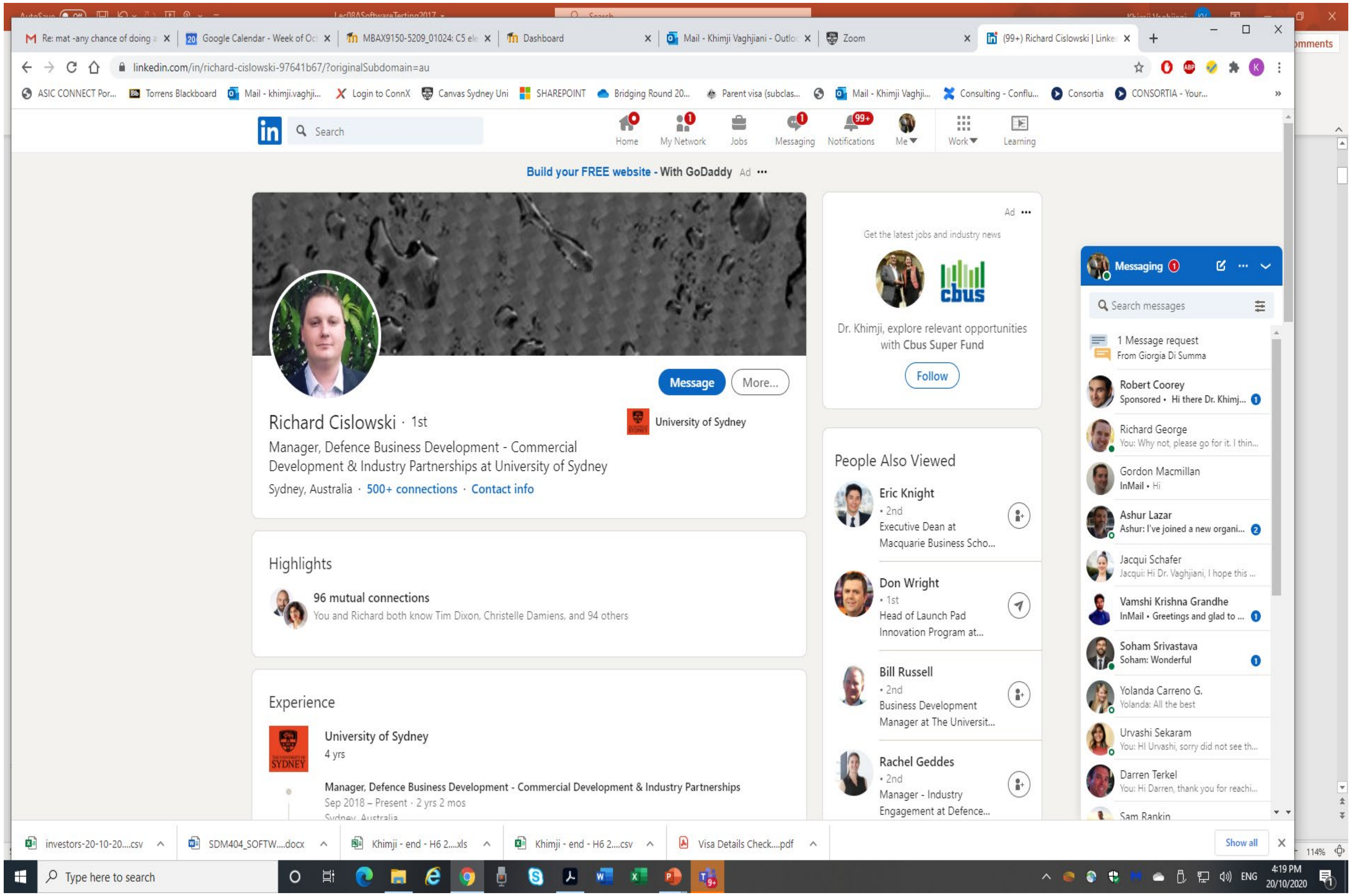
# Group Assignment



Any Questions ?

Open for submission 2nd Nov – 9th Nov

# Next week 9 Guest Lecture



Build your FREE website - With GoDaddy Ad

Richard Cislowski · 1st

Manager, Defence Business Development - Commercial Development & Industry Partnerships at University of Sydney

Sydney, Australia · 500+ connections · Contact info

University of Sydney

Highlights

96 mutual connections

You and Richard both know Tim Dixon, Christelle Damiens, and 94 others

Experience

University of Sydney

4 yrs

Manager, Defence Business Development - Commercial Development & Industry Partnerships

Sep 2018 – Present · 2 yrs 2 mos

Get the latest jobs and industry news

Dr. Khimji, explore relevant opportunities with Cbus Super Fund

People Also Viewed

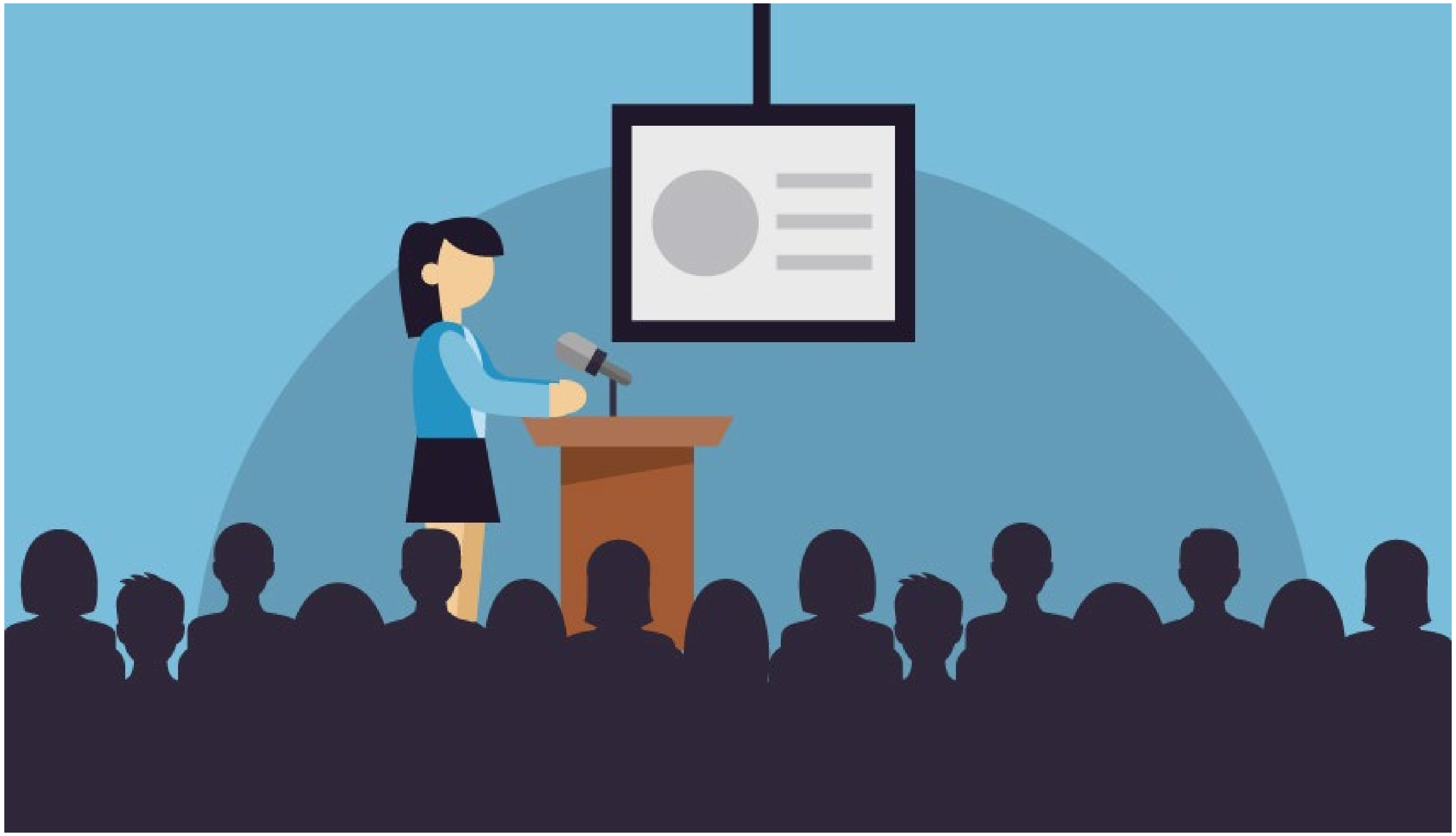
| | |
|---|---|
| Eric Knight · 2nd | Executive Dean at Macquarie Business Scho... |
| Don Wright · 1st | Head of Launch Pad Innovation Program at... |
| Bill Russell · 2nd | Business Development Manager at The Universit... |
| Rachel Geddes · 2nd | Manager - Industry Engagement at Defence... |

Messaging

| | |
|---|---|
| 1 Message request | From Giorgia Di Summa |
| Robert Coorey | Sponsored · Hi there Dr. Khimj... |
| Richard George | You: Why not, please go for it. I thin... |
| Gordon Macmillan | InMail · Hi |
| Ashur Lazar | Ashur: I've joined a new organi... |
| Jacqui Schafer | Jacqui: Hi Dr. Vaghjiani, I hope this ... |
| Vamshi Krishna Grandhe | InMail · Greetings and glad to ... |
| Soham Srivastava | Soham: Wonderful |
| Yolanda Carreno G. | Yolanda: All the best |
| Urvashi Sekaram | You: HI Urvashi, sorry did not see th... |
| Darren Terkel | You: Hi Darren, thank you for reachi... |
| Sam Rankin | |

Converting an idea and how to commercilise it.

## 2/3 teams to present to the CTO in class on progress at the end



## Software testing

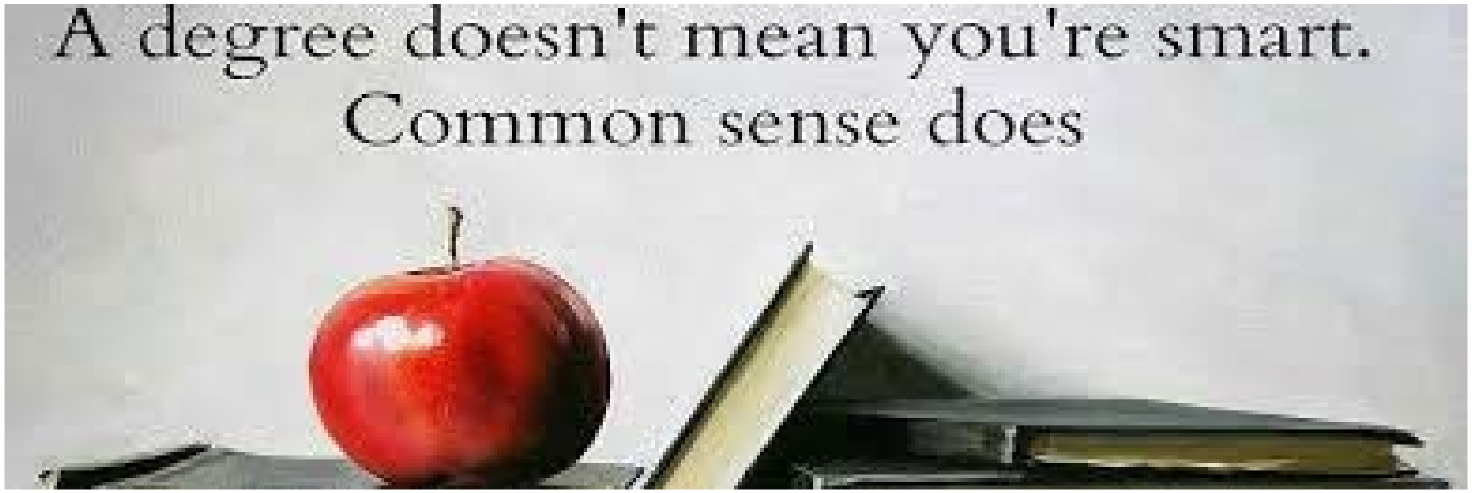
Software Testers/leads get paid

Between $350-$1200 per day

Depending on experience and testing field

Anyone interested?

## Common Sense !!



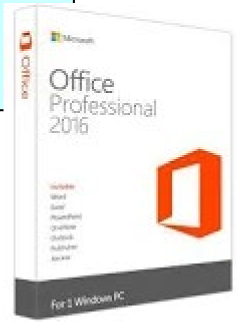
## By the end of this lecture you will be able to:

- Appreciate some of the difficulties of software testing
- Understand the terms error and failure
- Explain the concept of equivalence classes
- Appreciate the difference between ‘test data’ and ‘test cases’ and scenarios
- Construct a set of test cases to be used for program testing
- Go ahead with practice exercises

## Testing for bugs

# What makes software systems so different from engineering ones?

MS Office 2016 had over 30 million lines of code! By now … ?



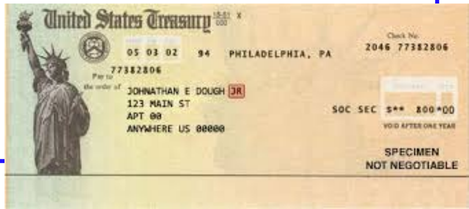
They are invisible and 'unvisualisable' compared with architectural & engineering constructs". (Fred Brooks, 1987)

The Sydney harbour bridge or the Opera House are both clearly visible, but, how can we visualise MS Office?

## Testing the SHB

# How much testing is enough? (1)

- **CASE STUDY 1:**
- In February 2003 the U.S. Treasury Department mailed 50,000 Social Security cheques without a beneficiary name.
  - **A spokesperson said that the missing names were due to a software program maintenance error.**
- In July 2001 a "serious flaw" was found in off-the-shelf software that had long been used in systems for tracking U.S. nuclear materials.
  - The software had recently been donated to another country and scientists in that country discovered the problem and told U.S. officials about it.



United States Treasury

Check No.
2046 77382806

05 03 02 94 PHILADELPHIA, PA

77382806

Pay to the order of
JOHNATHAN E DOUGH JR
123 MAIN ST
APT 00
ANYWHERE US 00000

SOC SEC $** 800*00
VOID AFTER ONE YEAR

SPECIMEN
NOT NEGOTIABLE

## How much testing is enough? (2)

- CASE STUDY 2:
- In October 1999 the $125 million NASA Mars Climate Orbiter (an interplanetary weather satellite) was lost in space due to a **data conversion error**.
  - Investigators discovered that software on the spacecraft performed certain calculations in English units (yards) when it should have used metric units (metres).

## Case Study

NextGen iPhone

## Testing strategy

What are you going to test ?
Functionality ?
Useability ?
What else ?

## Questions about software that only testing can answer

1. Does it really work as expected?
2. Does it meet the users’ requirements?
3. Is it what the users expected?
4. Do the users like it?
5. Is it compatible with users’ other systems?
6. Is its performance acceptable?
7. How does it scale as more users are added?
8. Is it ready for release?
9. Which areas need to be improved?

## What makes a program ‘good’?

### Eight important attributes of software

1. Correct
2. Reliable
3. Robust

Essential attributes

4. Useful
5. Usable
6. Maintainable
7. Efficient
8. Scalable

Quality attributes

## Three essential software attributes

1. **Correct**
   - behaves according to the functional requirements
   - produces the right result for any given set of inputs
2. **Reliable**
   - behaves as expected on every occasion
   - over any period of time
3. **Robust**
   - behaves in a predictable and controllable fashion even if the input not valid.
   - a program may be correct but not robust
     - e.g. division by zero or non-numeric input.

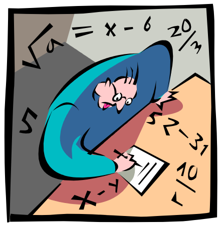
## Two ‘quality’ attributes

4. **Useful**
   - accomplishes something the user needs
   - output can be input to another program
5. **Usable**
   - ‘intuitive’ , i.e. can use without having to be shown how
   - minimum input  by the user
   - usability depends on the interface
   - compare  terms
     ‘user friendly’ and ‘ease of use’

## Three further ‘quality’ attributes

6. Efficient
   - requires minimum time and resource
7. Scalable
   - will continue to behave in an acceptable manner as size or volume of input is increased
   - scalability can depend on efficiency
8. Maintainable
   - easily modified for new requirements

# Why is software testing so difficult?



- Testing tends to be largely an informal task
- Testing can never guarantee to find all errors ...
- ... because there can never be a 'enough' test cases for testing to be said to be 'complete'
- **Testing is expensive, takes a lot of time and is very tedious**
- **Programmers often fail to find errors in programs that they have written**

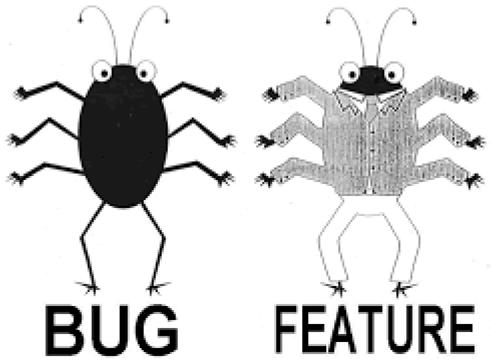
## Steps in testing a programme

For each functional specification (use case)

1. Specify a set of inputs
2. **Determine the expected output or result**
3. Execute the software
4. Compare the result with expected
5. Determine Pass/Fail

## What terms should we use?

- **Failure,**
- **Mistake,**
- **Error,**
- **Fault,**
- **Defect,**
- **Bug?**



## Component or unit testing

Making sure that each component or unit works as expected and according to the specification

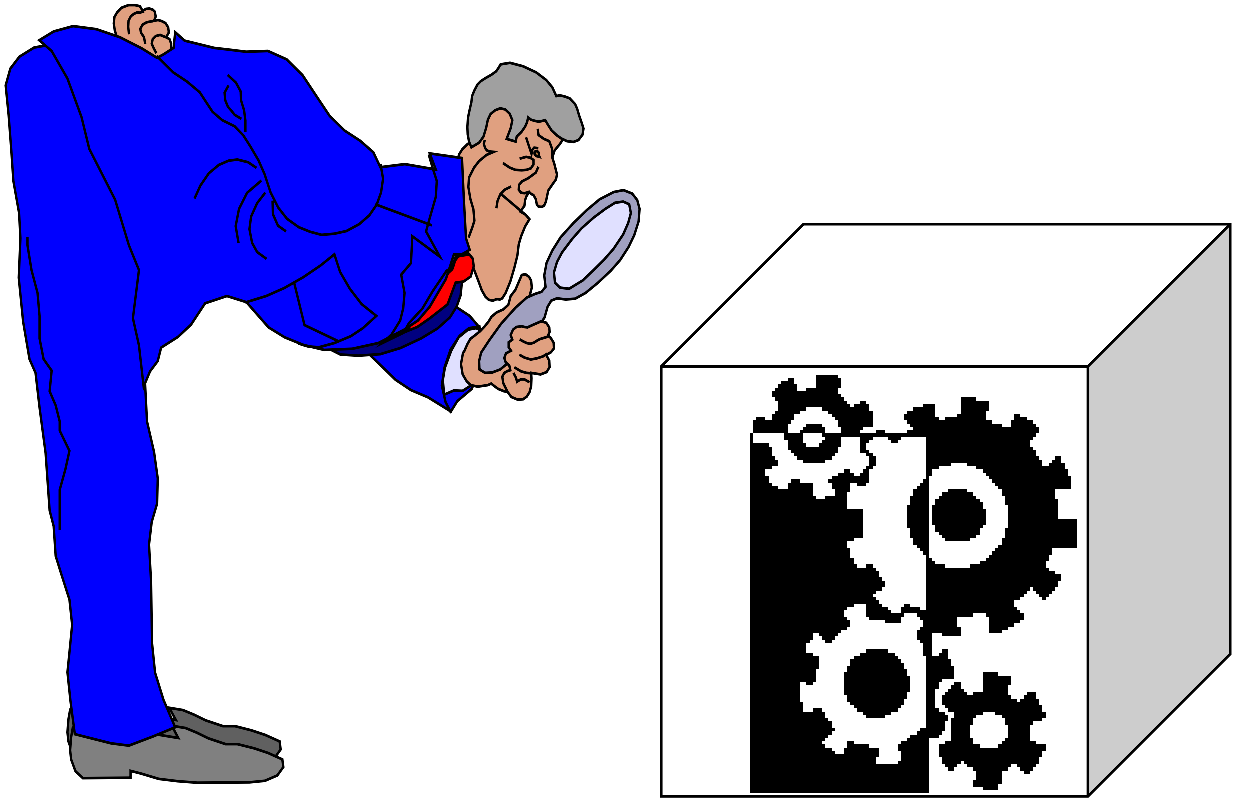
## The secret of software testing is to use good test data

It is not sufficient to use a ‘scatter gun approach’

- Test data must designed with care.
- Random input data generated automatically has low probability of finding defects

## Two types - White box testing

If we can "see" what's going on inside ...



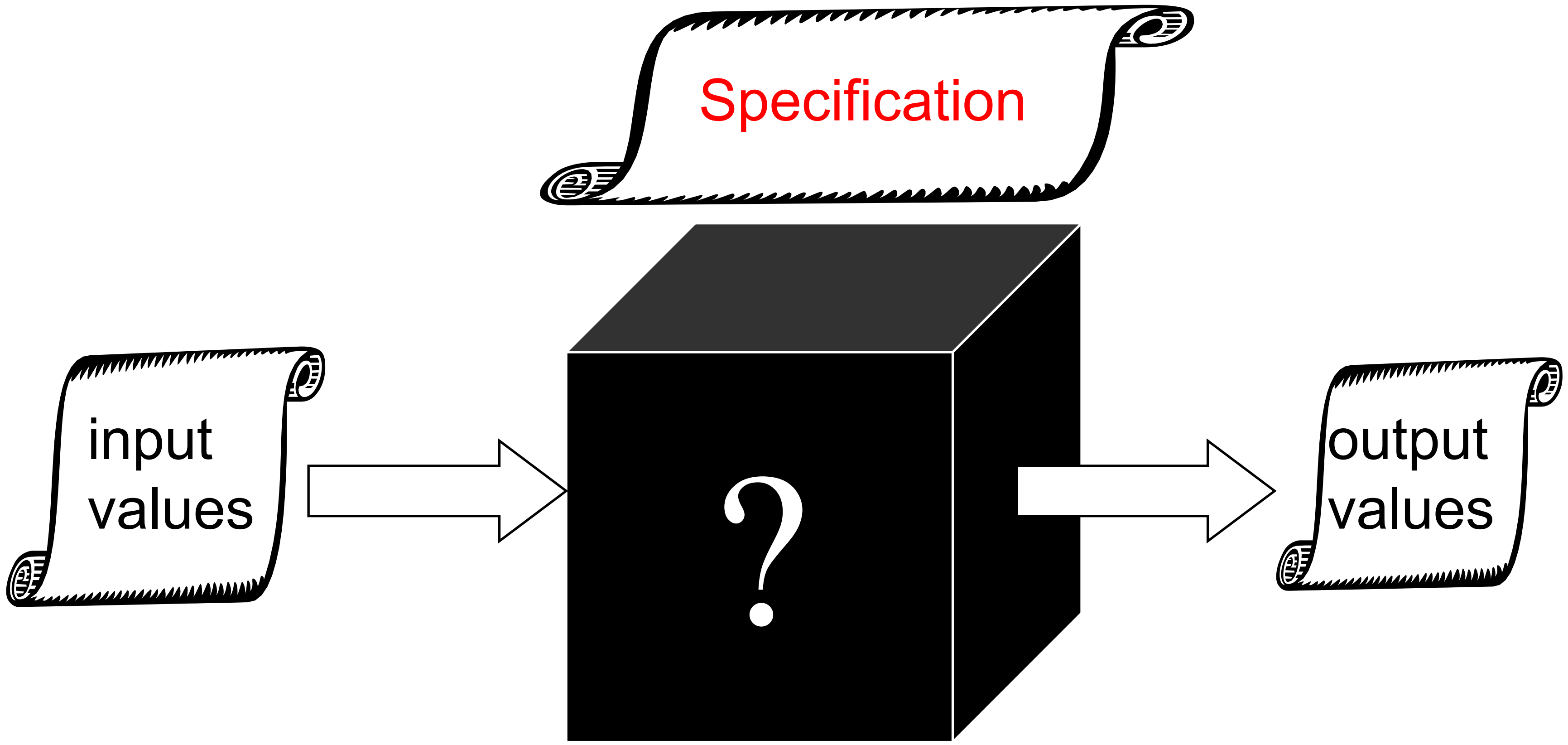
... then we know how best to test it!

## “White-Box” testing

- Applicable only to individual programs
- Used by programmers when they are writing software
- Can be more insightful because the programmer knows about the internal structure of a program
  - devises tests that have the highest probability of causing the program to fail
  - tries to exercise all possible logic paths through the program

# Black box testing

We don't know what's going on inside ...



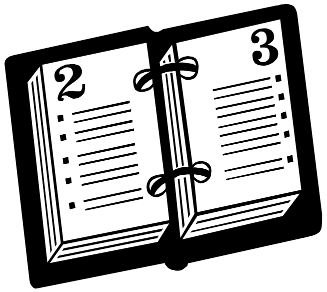
... but, we do know what output we should get according to the specification

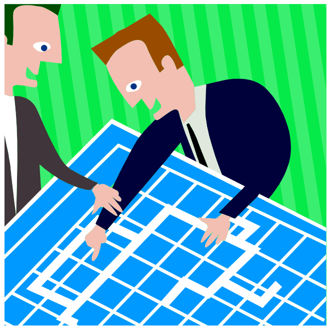
## “Black-Box” (input-output) testing

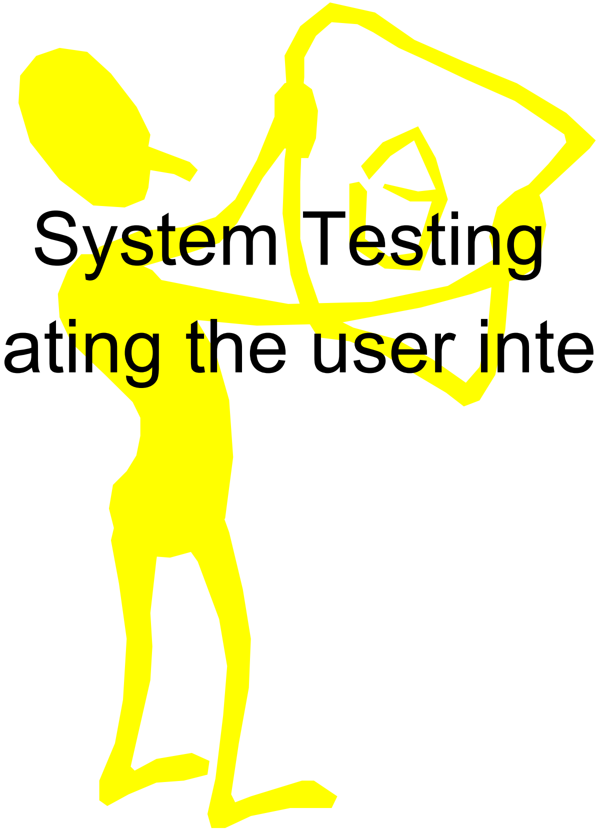
- Assumes nothing about the inner structure of the program.
- Input data values should be ‘sensible’ according to the application domain and the client specifications.
- Expected outputs for each test input must be worked out in advance, according to specifications and ‘business rules’.
- ‘Outrageous’ input values can be used to test robustness of program

# INFO5990 Professional Practice in IT

Lecture 08B







System Testing

Evaluating the user interface

## By the end of this lecture you will be able to:

- Understand what is involved in system testing
- Appreciate the value of continuous testing and the significance of the V-Model
- Appreciate the importance of the user interface
- Distinguish usability and ease of use
- Explain how ‘user satisfaction’ is assessed
- Make wise software choices

## User interface specialists are paid Between $700-$1000 per day

## Reducing cost
## Avoiding chaos!

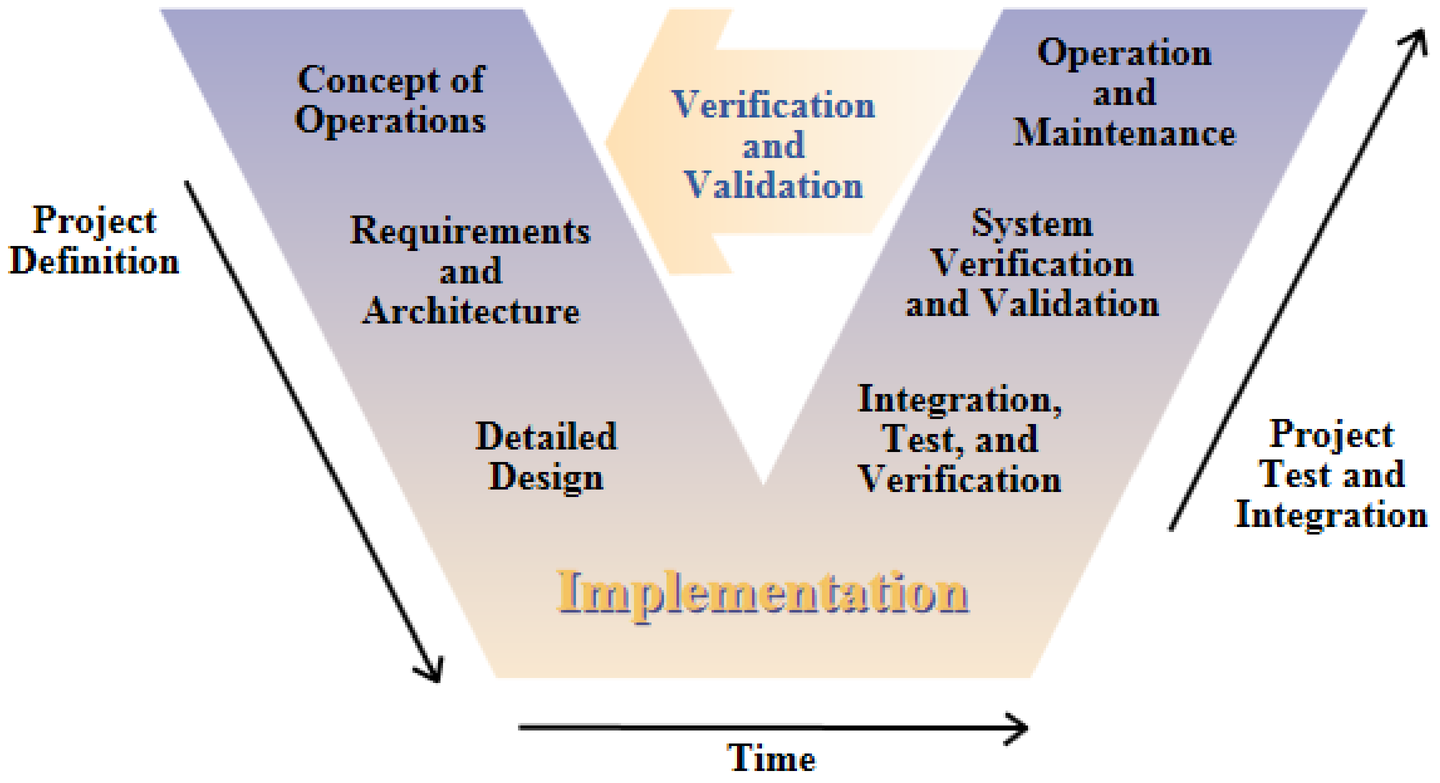
Continuous testing policy during system development

## The cost of rectifying software defects

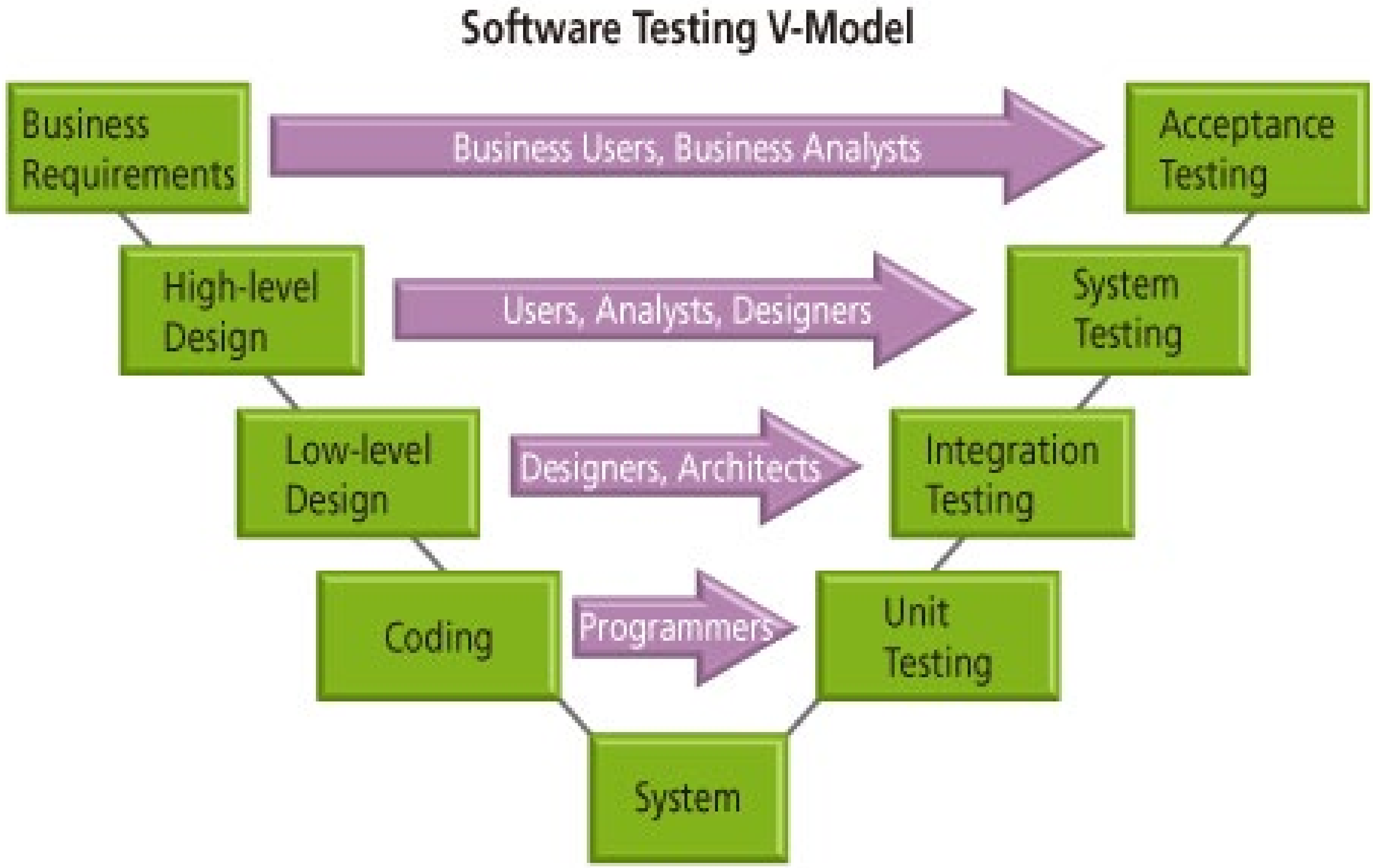
| **Stage at which Defect is Detected** | **Typical Cost of Correction** |
|---|---|
| User Requirements | $100-$1,000 |
| Coding/Unit Testing | $1,000 or more |
| System Testing | $7,000 - $8,000 |
| Acceptance Testing | $1,000 - $100,000 |
| ***After Implementation*** | ***Up to millions of dollars*** |

The sooner a defect is discovered the cheaper and simpler it is to rectify.

# The V-Model for software development


Concept of Operations
Requirements and Architecture
Detailed Design
Verification and Validation
Operation and Maintenance
System Verification and Validation
Integration, Test, and Verification
Implementation
Project Definition
Project Test and Integration
Time

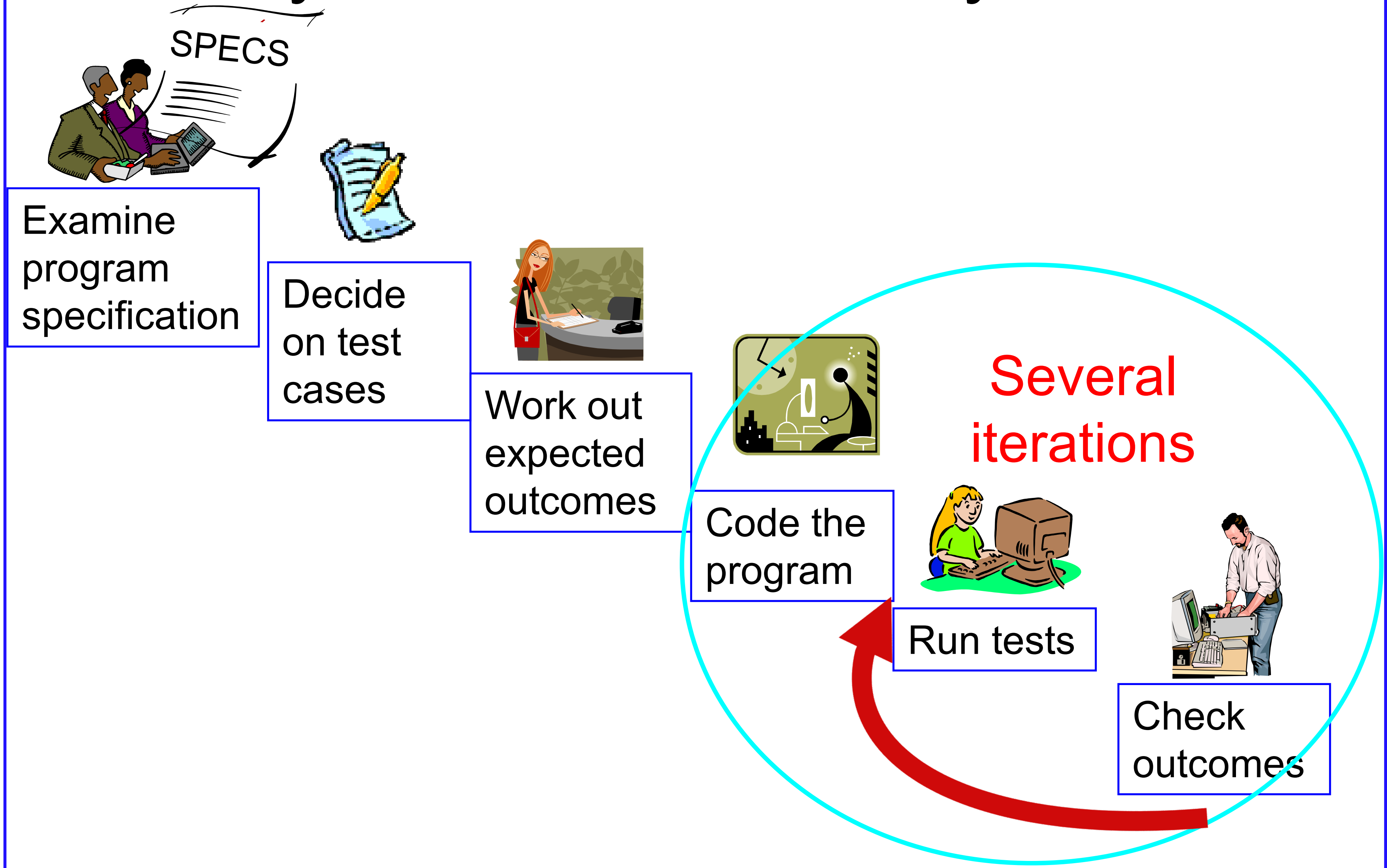
## The V-Model implies a policy of continuous testing Another view !



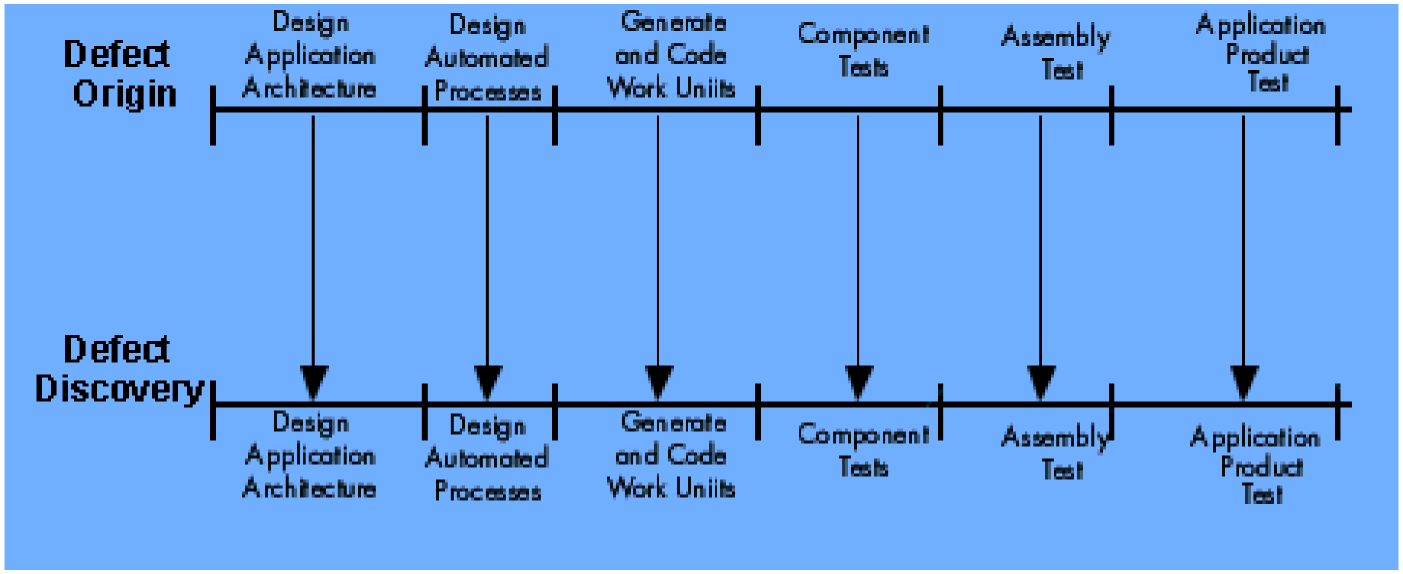
# Verification & Validation

- Verification: *Are we doing the job right?*
  - Checking for conformance and consistency with the specification
  - Process oriented
  - Static testing, using reviews, inspections, walk-through carried out by programming team
- Validation: *Are we doing the right job?*
  - Checking  that the specification is what the user actually wanted
  - Product oriented
  - Dynamic testing using test scripts, scenarios
  - Sponsor and end users involved in testing

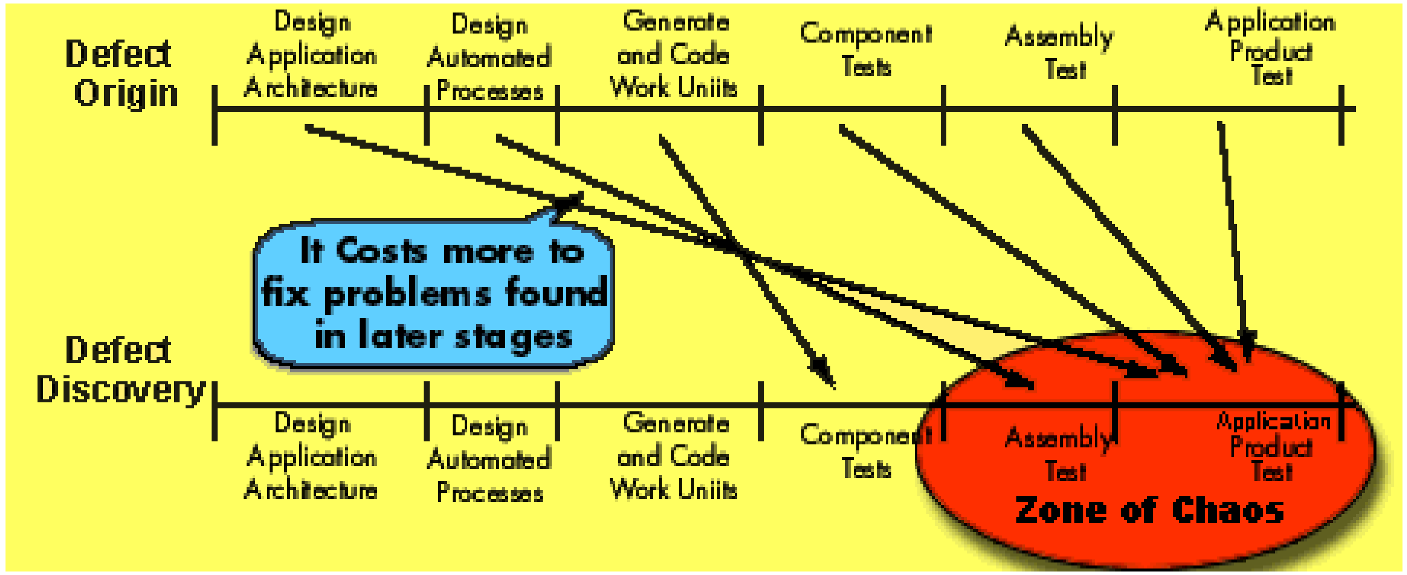
# System verification cycle



# The benefit of continuous testing


Defect Origin
Design Application Architecture
Design Automated Processes
Generate and Code Work Uniits
Component Tests
Assembly Test
Application Product Test
Defect Discovery
Design Application Architecture
Design Automated Processes
Generate and Code Work Uniits
Component Tests
Assembly Test
Application Product Test

"Continuous testing"


Defect Origin
Design Application Architecture
Design Automated Processes
Generate and Code Work Uniits
Component Tests
Assembly Test
Application Product Test
It Costs more to fix problems found in later stages
Defect Discovery
Design Application Architecture
Design Automated Processes
Generate and Code Work Uniits
Component Tests
Assembly Test
Application Product Test
Zone of Chaos

The alternative: "Put it off until later" policy

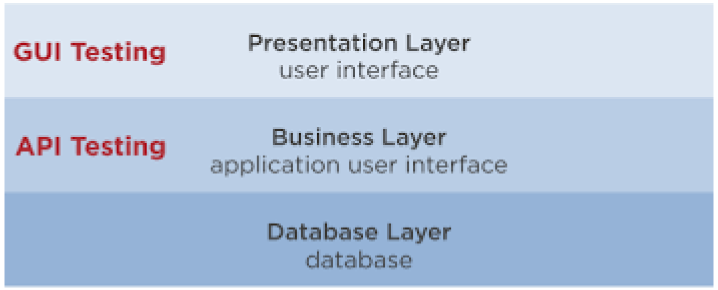
# Implications of the V-model

- Means that testing is considered early in the development life cycle, well before coding
- Avoids chaos towards the end of the project
- System design is continuously checked
  - against specifications (verification) and
  - against user requirements (validation)
- Means that the probability that the final product will satisfy the user's needs is much improved

## Case Study
## Using the V-MODEL

What would you do to ensure the V-model method was being used ?

## The user interface



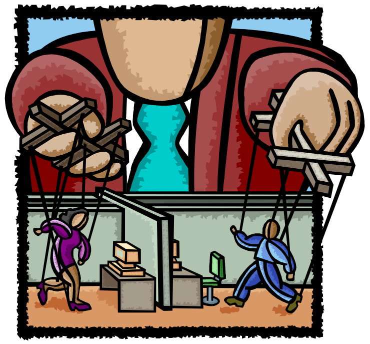
| | |
|---|---|
| GUI Testing | Presentation Layer<br>user interface |
| API Testing | Business Layer<br>application user interface |
| | Database Layer<br>database |

## What is the role of the user?

- Owner of the system
- Payer of bills

- Supplier of data
- Consumer of information

- Controller of processing

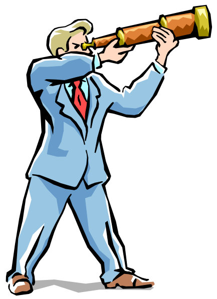


## Functions of the human-computer interface

- **From point of view of the Human:**
  - initiate programs
  - input data (facts about the real world)
  - set parameters for processing
- **From point of view of the Computer:**
  - request input (prompt the human)
  - present output requested by the human
  - report status or errors encountered in processing

# How do users see software

- Usefulness
  - does the job
  - makes them more efficient
  - gives them additional power; they can do more
- Usability
  - ease of use
  - easy to learn
  - 'user friendly'
- Subjective appeal
  - aesthetic appeal
  - similarity to other products used
  - good experiences



## Difficulty of testing usability

- To test the usability of new products must have skilled, highly-experienced users

  So, have to train subjects to high levels of skill before can even begin the experiment…

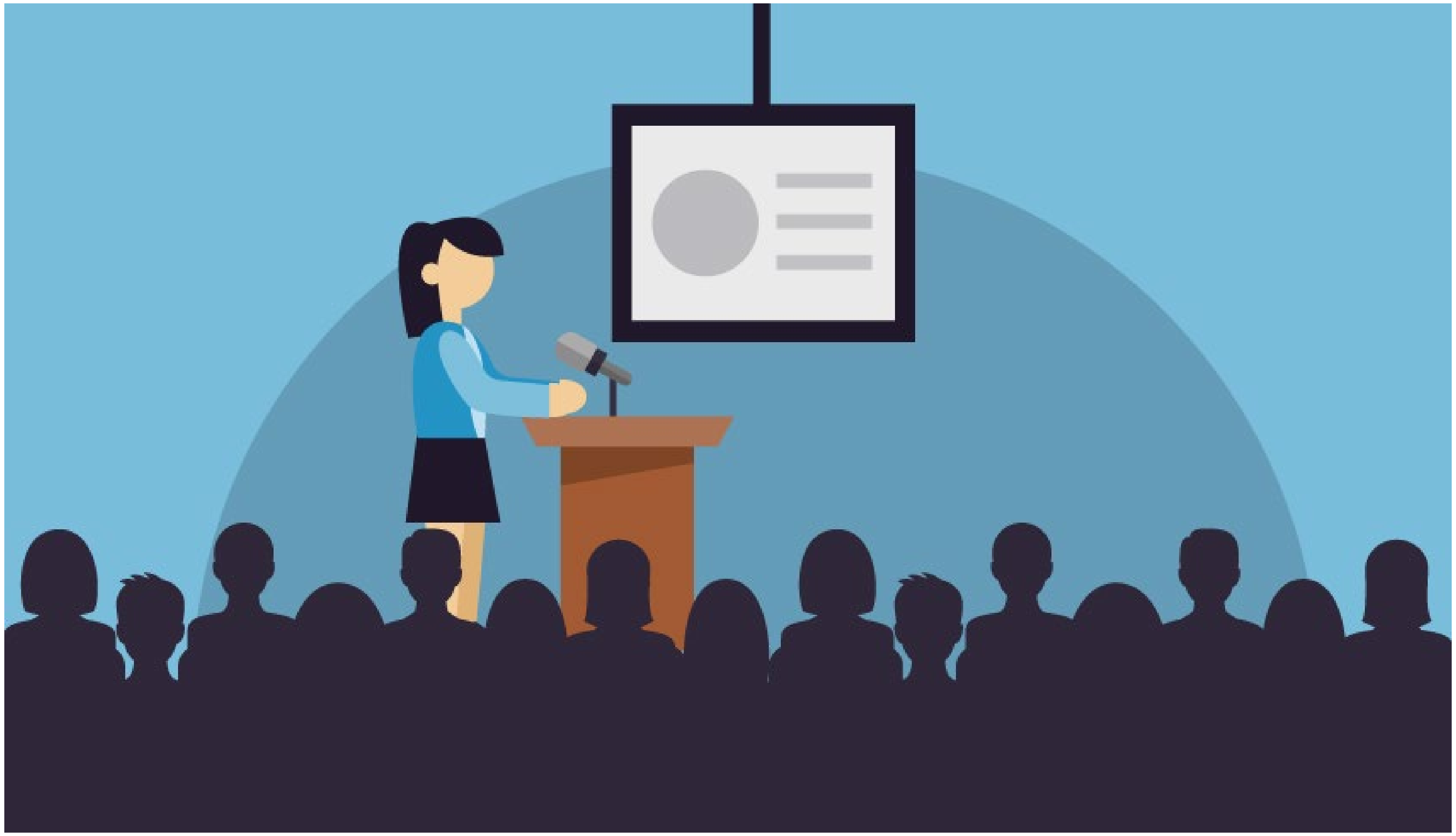
- "Intuitive" and "easy to learn" may not be synonymous when evaluating software, but often they are taken to be equivalent.

Larry Constantine, "*Persistent Usability*", *1994, OzCHI Proceedings, Australian Centre for Human-Computer Interaction*

## Case study :Dreamliner 787

1. Global supply chain
2. Multiple vendors
3. Specifications not integrated
4. I.T Integration and testing issues:
5. Major delays

## 2/3 teams to present to the CTO in class on progress at the end



Supplementary slides follows

Review at your own leisure

## User interface design guidelines

## Jakob Nielsen (1993) *Nine Heuristics* for interface design

1. Simple and natural dialogue
2. Speak the user's language
3. Minimize the user's memory load
4. Be consistent
5. Provide feedback
6. Provide clearly marked exits
7. Provide shortcuts
8. Provide good error messages
9. Prevent errors

Jakob Nielsen, *Usability Engineering*, 1993, Morgan Kaufmann, San Francisco

## Usability studies

## Evaluating ‘usability’

* *Ben Shneiderman, “Designing the User Interface”*, Addison-Wesley 1987

- Need to know about the user and the task
- Need to go deeper than a checklist of subjective guidelines
- Need to develop criteria that can be evaluated
- User friendliness “*is a meaningless concept*”*

# Larry Constantine (1994)
# *Eight Golden Rules* for interface design

1. Strive for consistency
2. Enable ‘power’ users to take shortcuts
3. Offer informative feedback
4. Design dialogues to yield *closure*
5. Offer simple error handling
6. Permit easy reversal of actions
7. Support ‘internal locus’ of control
8. Reduce short-term memory load

Larry Constantine, *Persistent Usability, OzCHI Proceedings, 1994*

# Bruce Tognazzini (2003)
# 16 Principles for user-interface design

1. Anticipation
2. Autonomy
3. Colour  blindness
4. Consistency
5. Defaults
6. Efficiency of the user
7. Explorable interface
8. Fitts' Law*
9. Human-interface objects
10. Latency reduction
11. Learnability
12. Metaphor use
13. Protect user's work
14. Readability
15. Track state
16. Visible Navigation

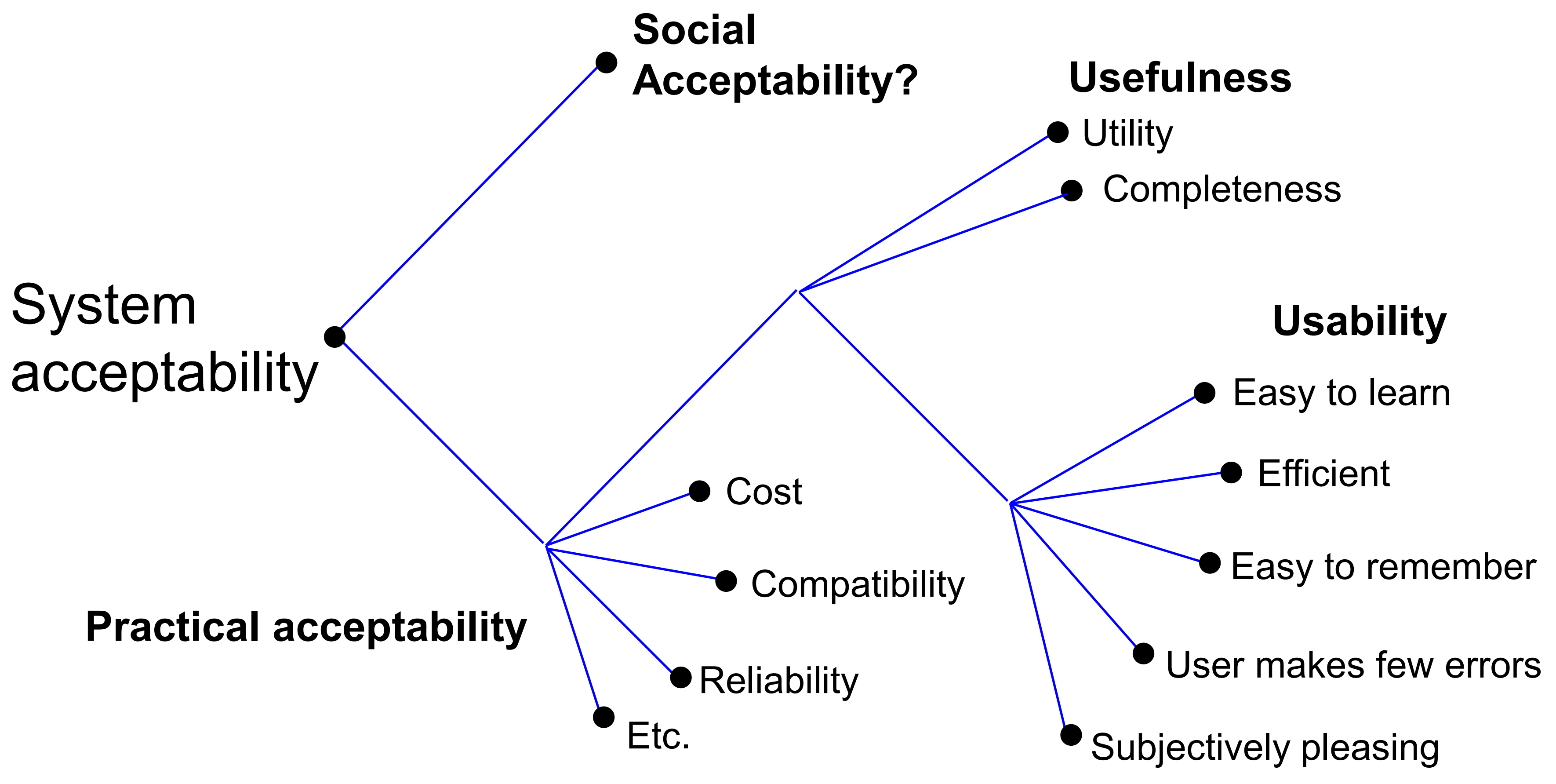
* Paul Fitts  (1954)  "The time required to rapidly move to a target area is a function of the distance to the target and the size of the target".

## Shneiderman’s usability criteria*

1. Time taken to learn basic/advanced skills
2. Speed of performance of *skilled* users
3. Retention of syntactic knowledge over time
4. Error rates and ease of correction
5. Subjective satisfaction

* *Ben Shneiderman, “Designing the User Interface”*, Addison-Wesley 1987

# Nielsen Usability Measures



Jakob Nielsen, *Usability Engineering*, 1994

## Larry Constantine on ‘usability’

- The Great Law of Usability
  - “A system should be usable - without assistance or instruction - by someone inexperienced with the system but knowledgeable and experienced in the domain of the application”
- The Lesser Law of Usability
  - “A system should not substantially impede or interfere with efficient and sophisticated use by highly experienced users”

Larry Constantine, *Persistent Usability, OzCHI Proceedings, 1994*

## Web site usability: what is measured?

Web site usability research, Lonergan Research Pty Ltd, 2009

- Learnability
  - How easy it is for users to achieve tasks when visiting your web site for the first time
- Efficiency
  - How quickly tasks can be achieved once a user is familiar with your site
- Memorability
  - How quickly proficiency can be re-established when users have not visited your web site for some time
- Errors & Problem Resolution
  - How often errors are made,
  - How serious these errors are, and
  - How easy it is for users to recover from these errors

# Measuring user satisfaction

C1. Does the system provide the precise information you need?
C2. Does the information content meet your needs?
C3. Does the system provide reports that seem to be exactly what you need?
C4. Does the system provide sufficient information?

A1. Is the system accurate?
A2. Are you satisfied with the accuracy of the system?

F1. Do you think the output is presented in a useful format?
F2. Is the information clear?

E1. Is the system user friendly?
E2. Is the system easy to use?

T1. Do you get the information you need in time?
T2. Does the system provide up-to-date information?

Measured on Five point Likert-type scale: 1 = almost never; 2 = some of the time; 3 = about half of the time; 4 = most of the time; and 5 = almost always.

William J. Doll and Gholamreza Torkzadeh, "*The Measurement of End-User Computing Satisfaction*", MIS Quarterly Vol. 12, No. 2 (Jun., 1988), pp. 259-274

## Benefits from improved usability

- Jakob Nielsen, 2003
  http://www.useit.com/alertbox/roi-first-study.html
- Data from 863 design projects
- Estimated productivity gains from redesigning an intranet interface to improve usability
  - for a company with 1,000 employees
    8 times bigger than the costs
  - for a company with 10,000 employees
    20 times bigger than the costs
  - for a company with 100,000 employees
    50 times bigger than the costs