Engineering software

Aim of session: To discuss strategies and techniques that can be used when developing some form of IT project

First: Why? What's wrong with software at the moment?

A number of questions

- Who has ever lost data or work?
- Who has suffered a 'crash'?
- Who has installed an update patch? Why?

Historical perspective (early 1960s)

Original problems solved using small programming teams

..small programming teams so..

- Small numbers of programmers
- Small numbers of communication lines
- Smaller problems specification
- Little management required
- Ad-hoc software development

Later:

Larger problems (enterprise size systems) tried to scale this approach

- 100’s or 1000’s of programmers
- No clear management approach

Outcome:

- Projects delayed / overbudget
- Full of bugs

Example

One IBM operating system (TSS/360) cost $millions to construct

- Every patch released contained more errors
- The OS was never fixed properly and was abandoned after 2 years

So think of a solution:

Which field of knowledge has had a great deal of success building things?

- Engineering
- 200 years of analysed management approaches
- Bridges typically stay up and tunnels generally don’t collapse

- Bridges typically stay up and tunnels generally don’t collapse

Tacoma Narrows Bridge (1940)

So apply engineering techniques to the construction of systems involving software

- Note the emphasis on *system* construction

This works really well:

Windows XP was released Nov/Dec 2001

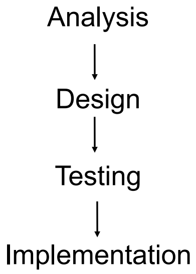
- By January 5 critical update patches had been released
- 2 of these were to fix problems introduced by the earlier critical update patches

What is well engineered software?

- Appropriate
- Reliable
- Efficient
- Maintainable

Many different suggested mechanisms for the development process

At an abstract level they all follow (at least)



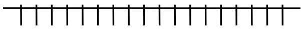
Analysis

Design

Testing

Implementation

They fall somewhere on this scale

Hard  Soft

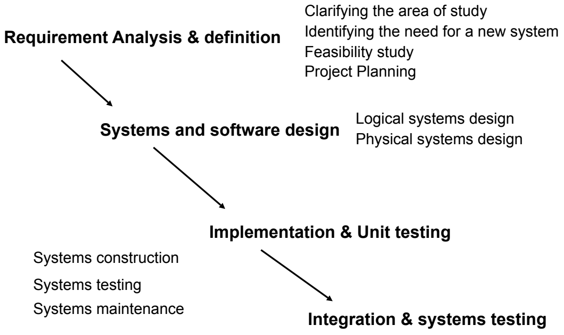
Strict	Less strict
Formal	Informal
Rigourous	Abstract
E.g.SSADM	E.g. Soft Systems

Example: SSADM

Structured Systems Analysis / Design methodology

- Strict methodology
- Large amount of paperwork
- Each stage precisely defined in various rulebooks
- A waterfall approach

SSADM stages



Requirement Analysis & definition

Clarifying the area of study
Identifying the need for a new system
Feasibility study
Project Planning

Systems and software design

Logical systems design
Physical systems design

Implementation & Unit testing

Systems construction
Systems testing
Systems maintenance

Integration & systems testing

Disadvantages of SSADM

- Very hard long drawn out process
- Large amount of documentation generated
- Slow, expensive

No one methodology is guaranteed to work

- SSADM has had some prominent failures
- London Ambulance Computer Aided despatch service
- London stock market system - Taurus

Looking at some of these phases in more detail

- The Analysis phase typically requires a requirements specification
- System requirements should be testable
- Yes/No or True/False responses

Examples

- The system will be fast

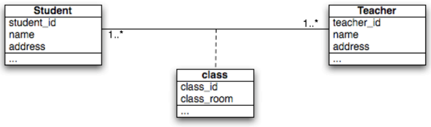
  The system must process 5000 records per second

- The system will have a user friendly interface

  The system should follow Apples GUI guidelines as sepcified in document … etc

The analysis phase can involve some form of diagramming notation

- UML (Unified modeling language)
- Diagrams of entities (objects) and the links (relationships between them)



student relation/ table

student_id	name	address	...
...	...	...	...

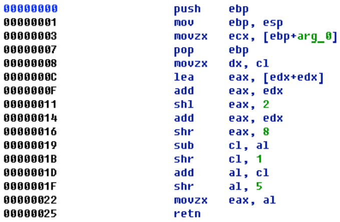
This process can also take place in the design phase

Design can be done in a variety of ways

- Exploratory - quickly build a system, try it, refine it, try it, refine it etc.
- Prototyping - Build a one off prototype, analyse it and then restart the process
- Formal transformation - construct a strict formal definition and then transform it into a working system (VDM)

Implementation

- Could be done in low level language (assembly or machine code)



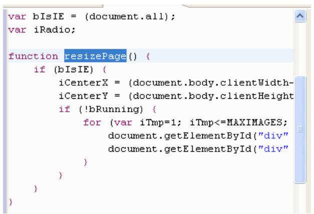
```
00000000                push    ebp
00000001                mov     ebp, esp
00000003                movzx   ecx, [ebp+arg_0]
00000007                pop     ebp
00000008                movzx   dx, cl
0000000C                lea     eax, [edx+edx]
0000000F                add     eax, edx
00000011                shl     eax, 2
00000014                add     eax, edx
00000016                shr     eax, 8
00000019                sub     cl, al
0000001B                shr     cl, 1
0000001D                add     al, cl
0000001F                shr     al, 5
00000022                movzx   eax, al
00000025                retn
```

Terse, harder to program

Very fast, small footprint (size)

Implementation

- Could be done in 3rd generation language such as Java, C++, C#, Fortran, Basic, Javascript, Java



```
var bIsIE = (document.all);
var iRadio;

function resizePage() {
    if (bIsIE) {
        iCenterX = (document.body.clientWidth-
        iCenterY = (document.body.clientHeight
        if (!bRunning) {
            for (var iTmp=1; iTmp<=MAXIMAGES;
                document.getElementById("div"
                document.getElementById("div"
            }
        }
    }
}
```

More abstract, higher conceptual level

May use some form of object hierarchy (assembly from reused components)

Implementation

- Could be done in 4th generation language such as SQL (Structured Query Language)

```
SELECT LastName
FROM Persons
```

More abstract again

Specify the problem without having to specify ‘how’ the problem should be solved

Case study: The web page as a system

- Often thrown together in adhoc style
- Classic problems
  - Out of date
  - Badly indexed
  - Not fit for purpose - marketing?
  - Badly designed
  - No regard for consequences - security?

Important:Reliability and maintainability