Waterfall Software Development Method (SDM)

**Definition**
- Define problem
- Analyse problem
- Plan the solution
- Implement the solution
- Error check

Dr. Hugh Melvin, Dept. of IT, NUI, G

**Why a structured approach?**

- Alternative- just jump in and do it!
  - leads to round-about solutions
  - undiscovered errors
  - takes longer
- The structured approach - plan the right way first
  - plans to avoid crisis
  - covers all eventualities
  - useful for team working
  - shorter in the end
- Computers need to be told EXACTLY what to do

Dr. Hugh Melvin, Dept. of IT, NUI, G

**Defining the problem**

- What exactly is required?
- Requirements gathering
- Generate and document a clear problem statement
  - System Requirements Specification / Document
- Need to talk to all stakeholders
- Most common cause of software system failure traced to poor requirements gathering

Dr. Hugh Melvin, Dept. of IT, NUI, G
**Analyse problem - WHAT?**

- I-P-O diagram
- What inputs are required?
- What outputs are required?
- Format of input / output
- Eg.
  - Program to convert miles to kilometres
  - Program to calculate the maximum of 4 temperatures

---

**Design algorithm - HOW?**

- **Algorithm**
  - an expansion of the processing steps in the I-P-O diagram
  - step by step set of instructions for solving a specific problem
- The algorithm is key to successful solution
  - take your time!
- **Use pseudocode or flowcharts**
  - tools to help describe the algorithm

*If you don’t know how to solve a problem by hand, you can’t do it by computer*

---

**A good algorithm must...**

- ... be sufficiently detailed to describe the transformations necessary to solve the problem
- ... always give the same result from the same initial situation
- ... give the correct result in all cases
- Use structured programming approach
Algorithm to record TV programmes

1. Insert a DVD
2. If unformatted..format DVD
3. If reqd, set record quality (SP HQ)
4. Repeat steps 4 - 7 for each program
5. Set date
6. Set start time
7. Set stop time
8. Set channel
9. Set DVD player to timed mode

Structured Programming

- Easy-to-follow logic
- Top-down design...divide and conquer
- Use 3 basic control structures to solve any problem

Control structures

1. **Sequence** is a group of instructions followed in order from the first through to the last

2. **Selection** (decision) is used to make logical decisions- to choose between alternate actions depending on certain conditions

3. **Repetition** (looping) allows a group of steps to be repeated several times, usually until some condition is satisfied
Describing your algorithm

- Number of methods
  » you choose the one with which you are most comfortable!!
- Pseudocode
  » English-like statements that are almost as specific as code
  » also called Structured English
- Flowcharts
  » graphic way to represent thinking
  » standard symbols and arrows connecting them

Pseudocode

- Accepting input
  » Read, Get
- Sending output
  » Print, Write, Put, Display
- Giving a value to something
  » Set, Initialise, use an assignment statement
- Arithmetic
  » +, -, *, /, ( ),
- Selection
  » If, Else, Endif
- Iteration
  » Do While, EndDO

Sample pseudocode

**Selection (decisions)**

```plaintext
IF Balance < Requested Amount
    Display "Insufficient Funds"
ELSE
    Display "Cash on the way"
ENDIF
```

**Iteration (loops)**

```plaintext
DO while numprogs < 5
    Accept starttime
    Accept endtime
    Add 1 to numprogs
ENDDO
```
Problem defined:
Input two subject results and calculate the total. If the total is greater than or
equal to 40, print message to indicate a pass, else print message to indicate
fail.

<table>
<thead>
<tr>
<th>Input</th>
<th>Process</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>res1</td>
<td>Prompt for res1, res2</td>
<td>Printed verdict</td>
</tr>
<tr>
<td>res2</td>
<td>Get res1, res2</td>
<td></td>
</tr>
<tr>
<td>total</td>
<td>total = res1 + res2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If total &gt;= 40</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Print &quot;Congrats, you passed&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Else</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Print &quot;Sorry, you failed&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End</td>
<td></td>
</tr>
<tr>
<td></td>
<td>End</td>
<td></td>
</tr>
</tbody>
</table>

Flowchart symbols
• Graphic way to represent thinking
• Standard symbols and arrows connecting them

Start/Stop
Start
Get it!
Input/output
Do it!
Processing
Is it?
Decision
Yes
No
Direction flow
start
Prompt user for res1, res2
Get res1, res2
total = res1 + res2
Is total >= 40?
print "Pass"
print "Fail"
end
### Pseudocode or flowchart?

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pseudocode</strong></td>
<td></td>
</tr>
<tr>
<td>Can be done easily on word processor</td>
<td>Not visual</td>
</tr>
<tr>
<td>Implements structured design elements well</td>
<td>No standard</td>
</tr>
<tr>
<td><strong>Flowcharting</strong></td>
<td></td>
</tr>
<tr>
<td>Standardised</td>
<td>Difficult to modify</td>
</tr>
<tr>
<td>visual</td>
<td>Need special software to do on computer</td>
</tr>
</tbody>
</table>

Dr. Hugh Melvin, Dept. of IT, NUI, G

### Write the solution in computer code

- Algorithm is in general terms - it may be applied to any computer language
- Before entering code into computer, desk-check it for logic errors or for syntax errors
- Use plenty of comments to explain code … documentation
- cf. Next section on programming languages

Dr. Hugh Melvin, Dept. of IT, NUI, G

### Testing and debugging

- Test with variety of data
- Test all limits for correctness
  - >= vs <
  - Eg. Fail if res < 40, Pass if res > 40 – what happens when res = 40?
  - Common logic error
  - ATM, machine, safety-critical systems: Testing critical
- Debugging: locating and correcting errors
  - Syntax error eg. IF (a < (b / c)) .. Missing closing bracket
  - Run-time error eg. Divide by zero
  - Logic error eg. Flaw in algorithm
- Programming packages usually have debugger software to help programmer

Dr. Hugh Melvin, Dept. of IT, NUI, G
Waterfall Approach

- Define problem
- Plan the solution
- Implement the solution

- Systems engineering
- Analysis
- Design
- Code
- Testing
- Maintenance

Dr. Hugh Melvin, Dept. of IT, NUI, G

Waterfall Approach

- Sequential
- Each phase completed before moving to next
- Requires a precisely defined requirements spec and analysis before any development
- Little or No flexibility
- Doesn't facilitate change process
  - Can result in spectacular failures
  - Unrealistic?

Dr. Hugh Melvin, Dept. of IT, NUI, G

Spiral Approach
Spiral Approach

- Iterative approach
- Allows for greater flexibility
  » Requirements may be added
  » Requirements may be changed
- Product is increasingly refined with each cycle of the spiral
- Problems can be identified & rectified earlier in cycle

Dr. Hugh Melvin, Dept. of IT, NUI, G

Extreme Programming Approach (XP)

- Rapid iterations of product development
- Full functionality is achieved over time, while partial functionality is achieved quickly
- An acceptance that change is inevitable
- More likely to deliver a useful system
- Small & Focused Development Teams
  » Pair Programming concept
- Otherwise not really that extreme
  » Still requires Requirements gathering/ Planning/ Tracking/ Quality Assurance
- Scrum programming is latest evolution

Dr. Hugh Melvin, Dept. of IT, NUI, G