A2 MODULE 5 (ICT5) TOPIC 14.9 SOFTWARE RELIABILITY

- *Describe methods of ensuring that software is reliable -å testing, ß testing, agreements between software houses and purchaser for testing.*
- *Understand the need for maintenance release(s).*
- *Understand the reasons why fully tested software may fail to operate successfully when implemented as part of an information technology system*

Methods of ensuring that software is reliable - chapter 62

Remember the following stages used in ICT 3 and ICT6:
1. **Unit Testing** - where each individual component is tested.
2. **Module Testing** - a module is a collection of dependant components or sub-routines.
3. **Subsystem Testing** - collections of modules are integrated into modules.
4. **System Testing** - Subsystems are integrated to make up the entire system. This stage of testing will examine whether or not the system meets its requirements specification.
5. **Acceptance Testing** - This is the final stage in the testing process before the system is accepted for operational use.

These stages will have to be repeated when modifications are made.

Make or buy? P.335

At the start of the systems life cycle, decisions have to be made about how to acquire the software that is needed. The options are:

- Software can be written by the end user
- A specialist department could design, write, test and evaluate the software
- External consultants could be called in to write and test the software
- An off-the-shelf package could be bought
- Software could be leased, with an annual fee payable for use.

End-user written software p.335

Computer-literate end-users can produce their own applications, using software such as Excel or Access. Their applications will be designed to do a specific job.
The advantages of this approach are that the end-user will know his requirements precisely and he should get exactly what he wants without waiting.
Disadvantages are that the end-user could leave and then wouldn't be available to provide technical support. His application may be incompatible with other software used by the organisation.
This approach is only suitable for minor projects with a limited life-span.

Writing software in-house

Advantages are that information can be kept within the organisation.
Disadvantages are that this approach requires people with certain skills who might need to be recruited. An external consultancy may have more experience of specialist skills

External consultants p.336

The job could be put out to tender. Cost is a consideration but a software house that has a proven track record of successful projects might be worth the extra money.

Buying a package p.336

The advantages of buying an off-the-shelf package are:

- Cheaper

- Immediately available
- Documentation is usually available
- Training courses may be available
- Technical support is usually available
- The organisation can look at reviews of the product and/or talk to other users
- Upgrades are usually available on a regular basis
- The disadvantages are:
- The package may not do exactly the job that is required.
- It may be incompatible with hardware or software that is already being used.

Leasing Software p.336

Instead of buying software, it can be leased for an annual fee. The cost is less in the short-term but in the long run this is a more expensive option.

Modifying existing packages p.336

Software can be bought and then modified. This is a dangerous option because it means that the manufacturer will no longer provide support. The modifications can cause errors in other parts of the software, which would then require more modifications.

Internet.com

PC Webopedia definitions and links for testing - white box, black box, alpha and beta.

The KJSSome highly recommended class notes entitled "Software Acquisition and Testing."

Software Testing Institute

http://www.softwaretestinginstitute.com/

Homepage of the STI. Loads of articles and links to material concerned with software testing.

http://www.aptest.com/resources.html

Free evaluation copies of software testing tools. These provide some useful insights.

http://www.softwareqatest.com/

Truly hideous looking site! Don't be put off as this site contains some excellent material.

A2 MODULE 5 (ICT5) TOPIC 14.9 SOFTWARE RELIABILITY

Problems in using software - BUGS

- Most choices in a program route include a series of steps that contain further choices and steps creating many complex paths. A program of several hundred lines can have tens of such choices and paths. Important programs within an organisation will have thousands or even millions of lines as well as hundreds or thousands of decision points.
- With so many lines of code and decision points a large number of errors will creep in.
- The consequences of these errors in code (or **BUGS** - remember Grace Hopper) can be expensive and in certain cases fatal to life or to the business.

Why do errors occur?

- **Typing errors** - almost certain to happen many times in dealing with tens of thousands of lines of code.
- **Lack of clarity in program specification**
- **Misunderstanding by the end-users** concerning what actions are needed under every possible combination of circumstances.
- **Featuritis** - glue piles of features without thinking about integration and how separate parts interact.
- Programmers take **undocumented shortcuts** that sometimes are not supported in future releases. Similarly programmers don't follow standard interface guidelines. Improper use of a complex programming technique to code a tricky passage
- **Inadequate testing** - leaving testing until coding is almost finished makes it difficult to turn back original design considerations.
- **Poor Documentation** leaves programmers who come back to edit programs in distress.
- **Pressure of time** usually commercial
- **Cost** - each line of the space shuttle's flight control software costs NASA about $1000 or 10 times more than for typical commercial software. Would you buy a word-processor or a spreadsheet for $5000 no matter how bug free it was? Or would you pay 90% less and live with the bugs?

Famous examples of bugs.

1. In 1945 **Grace Hopper** who spearheaded the standardisation of COBOL was asked to investigate why the computer she was working on had stopped. After looking inside the computer, she found a bug. It was a moth that had been beaten to death by an electrical relay signal. The moth was removed by tweezers (debugged).
2. Summer of 1991 - telephone breakdowns in California and along Eastern seaboard - errors in signalling software. - immediately before breakdown a bug was introduced by changing 3 lines of code in the several million line signalling program. After this tiny change no one thought it necessary to retest the program. Commercial pressure - give new feature NOW.
3. In 1986 two cancer patients in East Texas received fatal doses of radiation from the **Therac-25** computer controlled radiation therapy machine. There were several errors including the failure of programmer to detect mis-coordination between concurrent tasks. FAMOUS CASE

Solutions

- **The only way to make certain no errors exist is to test every line of code and every option. (See the diagram on P.337)**

However studies indicate that in a large application (e.g. payroll, hotel reservation system) would require thousands of years even on the most powerful computers available. This is NOT a realistic option, therefore: -

Sweat over the design specification - managers and customers often find it difficult to specify how a proposed program is supposed to perform.
Cleanly divides up tasks - makes it easier to co-ordinate and integrate. Structured programming - modularised. When bugs appear they are easily isolated and corrected. If necessary whole sections can be changed without affecting the basic logic.
Avoid shortcuts
Use comments liberally to describe code
Test extensively - BOTH the individual components and the interworkings of the entire system. But note - 400 people spent five years working on the Boeing 777 flight-control software.
Independently **validate** the product
Include **backup systems**
Technology-change and process-**change** within the organisation **need to be managed**

Syntax errors.

A compiler will report badly formed statements after it has tried to compile the source code; an interpreter will report illegal statements as it attempts to execute them

Logic errors - test data is created to try to trap errors.

- **Normal data** - the most general data for which the program was designed to handle.
- **Extreme values** - test behaviour of the system at the upper and lower limits of acceptability. Zero or null values important here.
- **Exceptional data** - illegal data - particularly where non-programmers use systems.

Methods used to test Software p. 337

Testing (alpha testing)

- Commercial software is traditionally developed to an incomplete state with some questions of design left unresolved. It is then issued to a restricted audience of testers usually within the developers' own company.
- Testing in-house i.e. with test data provided by software house, this is needed to test the implementation against design specification

. Testing (beta testing)

- Beta testing is done when software is being prepared for release. When changes on alpha test have been made the software is then issued to a number of privileged customers in exchange for their constructive comments. e.g. authors of how-to manuals, computer magazine reviewers.
- Testing is needed to detect errors not identified at the alpha stage, this is top-down testing i.e. real users become testers
- Another method might be to use off-site software house with `live data'
- Several beta versions are often released to iron out problems

Agreements

May exist between software houses and the purchaser for testing, in return purchaser is involved in development of the product, is given discounted prices and priority for upgrades or bug-fixes.

Maintenance Releases p. 338/9

The maintenance phase of a system begins once a system goes into production and lasts for as long as the system continues to be used. The maintenance process is triggered by requests from users who may report bugs or request new features. The cost and impact of each change will be assessed before a decision to go ahead is authorised. Minor changes in software packages are usually released with version numbers such as 3.1, 2.3, 3.3 etc. Major releases have new version numbers e.g. 4.0, 5.0. A number of reasons may trigger maintenance

- Previously undetected errors may be discovered in the software
- The original requirements may change to meet the changing needs of the user.
- Hardware developments may give scope for advances in software.
- New legislation may be introduced, which impacts on software (e.g. a new tax will affect accounting software)

Lehman and Belady 1985 p.339

- **The law of continuing change**
  A program that is used in a real-world environment necessarily must change or become progressively less useful in that environment.
- **The law of increasing complexity**
  As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure.
- **The law of large program evolution**
  Program evolution is a self-regulating process. System attributes such as size, time between releases and the number of reported errors are approximately invariant for each system release.
- **The law of organisational stability**
  Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.
- **The law of conservation of familiarity**
  Over the lifetime of a system, the incremental change in each release is approximately constant.

Types of Maintenance Release P.338	
Perfective	The system can be made better without changing its functionality e.g. improving performance speed, memory usage, etc.
Adaptive	The company may change the way it works e.g. moving from stand-alones to networks.
Corrective	Fixing bugs which only come to light after release e.g. the millenium bug.
HOW?	
mail-shot to all licensed users	despatch of update disk/floppy
computer bulletin board with detail of patches, fixed, known errors, etc.	

EXAMINATION QUESTIONS

Before releasing a new package the Software Company carries out alpha and beta testing
(a) What are these two types of testing and why are they both needed? (6)
(b) Explain why, once the package has been released there may be a need for maintenance releases and how might these be dealt with. (6)

(a) Alpha testing: Testing in-house, with test data provided by software house, needed to test implementation against design specification; 3x1=3
Beta testing off-site software house, with `live data', by real users, needed to detect errors not identified at alpha stage; 3x1=3
(b)Need for maintenance:
Perfective - improving performance speed, memory usage, etc. Corrective: fixing bugs, which only come to light after release.
Dealt with by mail-shot to all licensed users, despatch of update disk/floppy, and computer bulletin board with detail of patches, fixed, known errors, etc.

1997.5 (6 marks)
A software company is preparing to release a new application program. Describe the two types of testing carried out before the final release of the software. Explain why both are needed.

- *Each type must include justification to warrant 3 marks.*
- ***alpha testing** (1) - testing in-house (1) with data provided by software house (1) needed to test implementation against design spec.(1)*
- ***beta testing** (1), off-site/real user testing (1), using live data (1), needed to detect errors not detected at alpha stage (1) Beta testing involves a wider audience (1) & different environments. (1)*
- *Candidates may take an alpha/beta/reasons for both approach - use a 2+2+2 model*

1999.2 (5 marks)
A software company has notified customers of a maintenance release for its accounting package. The notification states that a programme of alpha and beta testing will be carried out to ensure that the maintenance release is reliable.
(a) State **three** reasons why a maintenance release might be necessary. (3)
(b) What is meant by the terms:
(i) alpha testing?
(ii) beta testing?(2)

*(a) **ADAPTIVE MAINTENANCE: EG***
To deal with y2k or EMU changes (1)
To deal with external issues such as tax law, budgetary, tax rate, etc (1).
To deal with hardware or software developments, new processors, new operating systems, etc (1)

PERFECTIVE MAINTENANCE: EG

To enhance functionality/ introduce new features of the package (1)
To decrease processing time (1)
To improve HCI (1)
CORRECTIVE MAINTENANCE EG:
To fix bugs/logic errors, coding errors, etc - NOT 'problems' must be more specific (1) Etc
Any 3x1=3

(b) alpha testing carried out by software house (1)
beta testing carried out by selection of software users (1)

June 2002.1
Software houses go through a long testing programme before releasing a product.
Despite this, problems can still occur with that product.
Give **three** reasons why testing may not be completely successful. (3 marks)

- *requirement to keep development cost to defined limits (1)*
- *requirement to keep development time to deadlines (1)*
- *in order to gain/maintain edge over competition - get product to market first (1)*
- *user has used product in a way that no-one has previously done (1)*
- *new hardware/ software is released the company was not aware of (1)*
- *inadequate test plan / data (1)*
- *etc.*

Spring 2003.7
A software company is producing a software package to perform an initial assessment of students entering colleges.
a. There is a fixed deadline for the release of this package.
Describe two effects that this might have on the final product. (4 marks)
b. The production of this package is a complex task. For this reason the company has decided to allocate sub-tasks to separate development teams.
Describe two benefits of this approach. (4 marks)
c. Two weeks after the release of the package, several colleges report identical problems with the software.
Describe what the company should do in this situation. (2 marks)

a.
- *Testing may not be fully carried out (1) so e.g. only the major functions of the software are checked/ only checked against a small set of hardware (1)*
- *Functionality may be left out (1) as it is deemed superfluous to requirements (1)*
- *Programming is not fully documented (1) so it is difficult to improve/ correct the software (1)*

2 x (2,1,0) marks
b.
- *This should reduced the development time (1) as parts can be worked on simultaneously (1)*
- *Personnel with particular expertise can be given parts of the system (1) so that the parts of the system are as efficient as possible (1)*
- *By using a modular approach (1) the system should be easier to test/ modify/ maintain (1)*

2 x (2,1,0) marks
c.
- *Produce a maintenance release/ software patch/ update (1) so that the software can operate with the operating system/ hardware that the colleges have (1)*
- *Provide a website/ document (1) so that technical staff have instructions on how to address the issue (1)*

1 x (2,1,0) marks

June 2003.1

Differentiate between *alpha-testing* and *beta-testing.* *(4 marks)*