COM6120/3210 - Software Measurement and Testing
Introduction to Software Testing

Structure of Course

This module will cover two main topics: software testing and software measurement. The software testing part of it will begin with a review of the basic concepts that have been covered at lower levels, and will then focus on different kinds of testing methods (functional, structural and machine-based). Then the measurement part will cover the basic principles of measurement and the kinds of measurements that can be made, and this will lead in to a discussion of how measurement can be used generally in the management of software engineering, which will include some discussion of the specific issue of how measurement can be used in managing the testing process.

Reading List

As with the background material on testing, there are two main texts for this part of the module:

G J Myers, *The Art of Software Testing*, Wiley, 1979, and
M Roper, *Software Testing*, McGraw-Hill (International Software Quality Assurance Series), 1994.

Unfortunately, neither of them are ideal texts: Myers is now rather out of date (although this is more a problem of it not covering newer techniques, rather than there being errors in what it says about the techniques that it does cover), while the key sections of Roper's book are really very tedious to read. Insofar as this module follows any one text closely, it will be Roper, and if you intend to purchase a book this would be the one to buy, particularly since it is likely to be useful as a reference for some considerable time into the future.

In addition to these, there is a useful texbook that has appeared fairly recently:
L Tamres, *Introducing Software Testing*, Addison-Wesley, 2002.
This approaches testing from the perspective of the whole testing process, rather than concentrating on the variety of methods that can be used for doing it. It is therefore a useful supplement to the above texts.

As well as these, there are also a number of texts which are either older or more advanced, but which are worth looking at as secondary sources (although some of them are fairly hard going!):

R A DeMillo, W M McCracken, R J Martin & J F Passafiume, *Software Testing and Evaluation*, Benjamin/ Cummings, 1987;
W C Hetzel, *The Complete Guide to Software Testing* (2nd ed.), Wiley (QED Information Sciences), 1988;
W E Howden, *Functional Program Testing and Analysis*, McGraw-Hill, 1987; and
M A Ould & C Unwin, *Testing in Software Development*, Cambridge University Press (BCS Monographs in Informatics), 1986.

Assumed Background

The background knowledge of testing which will be assumed for this module is as follows, where more details of these topics can be found in a separate set of notes.

- The motivation for testing: the unavoidability of errors, due to the ways in which different kinds of mistakes are caused by limitations of human memory (short term, medium term and long term).
- How mistakes occur in software development processes, and the distinctions between errors, faults, latent faults and failures, as these are defined in the relevant standards (IEEE standard 729 of 1983, and BS7925 Part 1 of 1998).
- The definition of testing that will be used (which is an update of the one given by Myers): "Testing concerns the selection of inputs for a system with the intention of causing failures in the system, and thereby revealing the presence of faults."
- The definitions of key concepts in the testing process, as taken from the standards: test case, test set, test script, test execution, test analysis, and test suite.
- The way in which testing as an activity has developed, through different emphases on: debugging, demonstration, destruction, evaluation, and prevention, to the view that it must be carried on throughout the lifecycle rather than just at certain stages in it.
- The need for testing methods, and the need for testing methods to produce test sets that are: of realistic and controllable size, better when larger, realistic in the cases that they cover, representative, and possibly adequate according to some criterion. Also an appreciation of the relationships between these properties, and the difficulty of formalising them.
- The basic distinction between some common kinds of testing methods: random testing, specification-based (or functional) testing, implementation-based (or structural) testing, and hybrid methods.

- The key concepts of specification-based testing methods: equivalence partitioning, boundary value analysis, and the use of these in identifying equivalence classes.
- The different stages at which testing is used in the software development process, and the corresponding kinds of testing: unit testing, integration testing, system testing (both alpha and beta forms), exploratory testing, non-functional testing, stress testing and acceptance testing.

System Models and Testing Methods

As described at the start of this handout, the rest of this part of the course will be concerned primarily with examining different testing methods in detail. Before doing so, though it is worth examining the role of different kinds of system model in testing methods.

Essentially, there are four main kinds of models that are created during the course of developing a software system.

- A requirements model, which describes how the system is supposed to fit into its environment (in other words, the kinds of data that it has to be able to handle, the functions that it has to be able to perform to handle this data, and the associated performance and quality measures).
- A specification, which gives a more precise definition of the data and functions, and possibly also of the performance and quality measures that relate to them.
- A design model, which describes the internal structure of the system and how the functions will actually operate, and which ideally should provide enough information to determine whether the performance and quality requirements will be met (but with the present state of knowledge this ideal is not always achievable, never mind actually being achieved).
- An implementation model, which may either be the code itself, or some abstraction from its structure.

In principle, testing can be concerned with the relationships between any two of these models, although in practice it is ultimately concerned with ensuring that the relationship between the implementation and the requirements is correct. To do this in one step, though, may often require a huge amount of work, and sometimes it is more efficient to test the individual stages in the relationship separately, and then put them together. Thus, it may involve less work to structure the testing into two stages, such as:

- First test the specification, to ensure that it correctly captures all the requirements; and then
- test the implementation against the specification, to ensure that all specified features are correctly implemented.

Other combinations of stages are also possible.

The consequence of this is that, for any stage in the testing, there are two models involved, and in principle either of them could be used as a basis for the operation of a test method. Thus, ignoring random testing, the kinds of test method that have been identified previously can be seen simply as special cases of this.

- Functional testing bases the test method on either the requirements themselves, or the specification (and in practice usually both, because while test cases can be identified from the requirements, a test case also has to define the expected output, and usually this is given by the specification).
- Structural testing bases the test method on the implementation model (although again, while this can identify the test cases, determining the expected output will usually involve going back to the specification).

Of course, if two of the models are very similar in structure, then it does not matter so much which one is used as the basis for the test method, since one should get similar test sets either way. In particular, this occurs frequently for systems whose behaviour can be modelled naturally in terms of state machines (or statecharts), and the use of these models is becoming increasingly common, which is why test methods that are based on machine models will also be covered within this course.