APPENDIX 2: DISCUSSION TOPICS

The following is a collection of topics that lecturers may wish to put forward to the students in their lecture groups where time permits. The topics are largely drawn from the series of assignment topics since 1996.

Semester 2 1996: Software Quality

Methods, tools and procedures are known as the concepts of Software Engineering, and all work together towards a single goal, i.e. **achieving good software quality**.

(a) Several "checklists" have been developed for software quality. For example, McCall's categorization of Software Quality Factors is one such popular checklist. Discuss any **one** of such checklists, explaining each factor in your own words. A minimum of **five** factors should be covered. [15 marks]

(b) There is also a degree of interdependence between **each** factor. Show **graphically** this interdependence by drawing a graph of nodes, where each node represents a certain software quality factor, and an arrow from node A to node B will indicate that quality factor A contributes towards achieving quality factor B. Provide some explanation for each relationship that you indicate. [10 marks]

(c) Referring to (b), what does your graph indicate to you about the relative importance of the software qualities, i.e. are there any factors that seem more **important** that others? Provide some explanation. Illustrate your answer with a table showing the relative importance of each factor. [5 marks]

(d) Why is there always some **tension** between a software engineering group and an independent software quality assurance group? Explain how this tension comes about, and also indicate with explanation, whether in your opinion, this tension is healthy and can provide for greater software quality. [5 marks]

(e) Explain, in your opinion, whether the implementation of good software quality is being practiced in the real world, i.e. are real-life companies taking the extra effort to ensure that their software products are of good quality? Justify your answer by drawing upon a real life example of a software company. [5 marks]

Semester 1 1997: Software Quality Assurance

***Objective**: To expose students attempting the Software Engineering unit an initial appreciation of the issues affecting Software Quality Assurance in the real world.*

a) There are several "Major Activities" that can be conducted in the assurance of Software Quality. Describe any ***four*** of them in about 1 to 1.5 pages. [10 marks]

b) You have just been given the responsibility of improving the quality of software in your organization. Discuss some of the measures you will take. You should discuss to some detail at least 4 distinct measures or activities. If you wish to use the activities described in part (a), your answer here is still required to be real-life implementation measures of the activities in part (a) that you are proposing. You may

wish to form your answer into a formal proposal containing a series of measures to management. Your answer should be about 2-3 pages. [20 marks]

c) The writer of the following article, Steven[1], has about 20 years experience as a software engineer, primarily in the area known as "embedded software", which means products in which a microprocessor is embedded in custom hardware and shares execution of functionality of the product in such a way that it is not obvious to a user what is done by hardware and what is done by software. However, there are many aspects of his work which are common to all software development.

Recently, he posted an article onto the Internet Newsgroup in response to a thread of arguments on whose responsibility should Software Quality Assurance be attributed to: the Software Publisher (the organization who prepares the software for distribution) or the Software Developers (the programmers who actually code the program). The basic scenario in discussion was that one Software Publishing house, GET2[1], had published a software program, a game called "WarShip[1]", that had so many bugs it was simply un-workable. One of the reasons that readers of the Newsgroup had earlier attributed was that GET2 knowingly did this was because they were tired of the head software developer, M. John[1]'s, constant postponement of deadlines. As a result, GET2 simply released the latest (and apparently uncompleted) version of the program that John had given to them. Not only that, these same readers accuse John of having a share of the blame, since he was the chief person in charge of the program's code, of which it was so filled with errors that most people simply could not even start the program.

This is an excerpt of his article, which discusses about Software Quality Assurance.

> "We do not know how to write software which has no bugs. No-one knows how.
>
> The operating software for the Space Shuttle is probably the best crafted code ever written. Unbelievable amounts of effort went into design and checking on it. The quality levels and procedures used equal or exceed that for any other project ever done before or since.
>
> And... the first launch of the Shuttle was scrubbed because of a software bug, with only a few seconds to go before the engines lit.
>
> It has been said, with much truth: "If carpenters built buildings the way that programmers write software, the first woodpecker to come along would destroy civilization."
>
> The problem, simply, is that a typical program is too large to fully comprehend. No-one can understand it all; it is always necessary to deal with levels of abstraction, and abstraction ***always*** leaves out details.
>
> As an engineering work, software differs qualitatively from every other kind of engineering, because we programmers have our work measured against a mathematical level of perfection. No engineer is perfect, but other engineers can live with the fact that their work permits tolerance.

[1]Names have been changed to protect the associated parties.

[1] Names have been changed to protect the associated parties.

Computers have no tolerance, we programmers gotta be perfect. ("The Psychology of Computer Programming" has a beautiful aphorism: "Programmers call their mistakes 'bugs' because no-one could admit to making as many mistakes as programmers make.")

To that end, software Quality Assurance (QA) must be different than QA for any other field. For any other field, QA is a statistical process to determine the extent to which problems exist. For software, on the other hand, we already ***know*** that there are problems, because there ***always*** are problems. Even the best programmers will let a few through. So QA is responsible not for sampling, but rather for problem identification.

Rather than determining whether there ***are*** problems, software QA attempts to determine the severity of the problems which we're sure are there, and tries to specifically identify as many as possible before release so the original programmers can correct them.

There are various processes known which permit problems to be located; each is imperfect, a sieve with major holes in it. After applying several such sieves (top-down design, specification walk-throughs, code review, unit test, system test under hostile load, QA), you hope that only a few problems remain and that none of them are large. (Exhaustive brute force testing is not an option. For large programs it can be proved that if every sub-atomic particle in the universe was a software tester, and if each one tested a path through the program at atomic vibration rates, they still wouldn't be finished when the universe winds down. The number of tests required for exhaustive testing ain't "infinity", but for all practical purposes it may as well be.)

The danger is when this filtration process gets short-cutted, or when it doesn't work in the first place. If management, for instance, pulls the plug and says "It ships now or it don't ship at all" (which is apparently what happened with Warship), then the process of finding and correcting problems is indeed short-circuited, and the result is predictable to any experienced programmer.

So where does the fault lie? With the management? To answer that question you have to know more. If management does that to a normal team who would have finished on their own shortly thereafter, then sole blame certainly lies with management. But with someone like John, who simply would never have finished the game without a gun at his head, I blame him exclusively.

Of the three choices: Ship a good product, ship crap, stop development and write off your losses; clearly the first is the best one. But John is responsible for removing that choice from GET2's universe of discourse. So they let John decide between the other two choices, and he decided (not surprisingly) to ship what he had, which turned out to be crap."

Based on the above article, discuss whether it is more of the Software Publisher or Software Developer's responsibility to ensure bug-free software programs. Justify your reasoning with at least three to four points. Your answer should be about one page. [10 marks]

Semester 2 1997: Programming Languages

The final objective in software development is to translate design representations into a form that can be understood by the computer, i.e. in programming language form. Thus, the role of the software engineer extends to the consideration and subsequent choice of programming languages to implement some program design.

Question (40 marks)

a) There are a number of factors that one would take into consideration when choosing a programming language to develop some software program. For example, one such factor would be the application area that the programming language is best suited for. Describe firstly any other ***four*** such factors that should be taken into consideration. [10 marks]

However, given practical considerations, it often becomes necessary to ***prioritize*** these factors, given resource limitations. In other words, some of these factors would need to take greater significance over others. Based on the factors you have chosen earlier, and any others you have in mind, prioritize the factors, and ***justify*** your prioritization. [10 marks]

b) Different programming languages are suited to develop different types of software. In fact, it has become commonplace for a software program to be developed using a variety of different programming languages.

Choose ***one*** of the application areas below, and prepare a short report of approximately 1000 words on how the programming language(s) can be used to develop the particular application.

i) A Real Time System;
ii) An Operating System;
iii) An Expert System;
iv) A Flight Simulation game;

In your report, you should remember to include the following:

- A brief description of the application area, and a program falling into this area.
- Example(s) of the programming language you've chosen.
- Why and how this programming language is appropriate for the development of this application. [20 marks]

Semester 3 1997: Modern Software Design and Development Concepts

The objective of this assignment is to introduce important concepts of software development to students. The last part of the assignment also discusses the importance of design.

Part (1) 30 marks

The following are explanations adapted from Roger Pressman in *Software Engineering: A Practitioner's Approach* of three software development concepts: Software Reusability, Software Modularity, and Software Reengineering.

> "*Software Reusability* is an important characteristic of a high-quality software component. That is, the component should be designed and implemented so that it can be reused in many different programs."
>
> *"Software Modularity* indicates the degree to which software can be divided into separately named and addressable components, called modules, that are integrated to satisfy problem requirements."
>
> *"Software Reengineering* recovers design information from existing software, but uses this information to alter or reconstitute the existing system in an effort to improve its overall quality. In most cases, re-engineered software implements the function of the existing system. But at the same time, the software developer also adds new functions and/or improves overall performance."

Choose any ***two*** of the three concepts above, and prepare a short report discussing the two concepts. Some examples of points of discussion you may want to include in are: the importance of the concept, its characteristics, an example of how it is dealt with (e.g. a case-study). The total number of words for *each* concept should be about 750 – 800 words. [2 * 15 marks, Total 30 marks]

Part (2) 10 marks

One software developer once commented the following: that he believes in nailing down 99.5 percent of the software design, before a single line of code is ever written. In a practical sense, more emphasis is often given to the design of software, rather than its implementation. For example, typically at least 25 weeks of a 40-week period given to a mid-size software development project would be devoted on design alone.

Based on the above sentence, prepare a short report of about 500 words on the *importance of design*. [10 marks]

Semester 1 1998: Modern Software Development Life-Cycles

The objective of this assignment is to introduce important concepts of software development to students, namely the use of life-cycles and methodologies in software development. The assignment consists of two parts, which are to be answered.

Part (1) 30 marks

(a) There are many different types of life-cycles that serve as possible models for software development, for example the prototyping life-cycle or the classical life-cycle as studied in the syllabus. Describe a life-cycle for software development of your choice. Your answer should include discussion in the following areas:

- Description of the life-cycle, with clear explanation of the different steps of development;

- Diagrams or illustrations of the life-cycle;
- Possible application areas using this life-cycle;
- Possible advantages and disadvantages of this life-cycle;
- Comparison of this life-cycle with the life-cycles found in the study-guide;

Although you may choose one of the life-cycles covered in the study guide, you are ***strongly*** encouraged to use a life-cycle found in other software development publications, or a methodology specifically developed and used in your organisation. Examples of such methodologies include the Soft Systems methodology, Information Engineering methodology, or Object-Oriented development methodologies.

[20 marks]

(b) It has been widely commented that software development should ***not*** end after delivery of the completed software product to the customer. Do you agree with this statement? In your answer, justify your answer, and comment on how post-development activities will enhance the value of the software product to the customer.

[10 marks]

Part (2) 10 marks

Undertake a ***real-life*** example from software development journals, magazines and reference textbooks in which the software development project was completed through the use of ***prototyping***. Describe this case-study in your own words. Your answer should include discussion in the following areas:

- Description of the software, its nature and application area;
- A brief write-up on the prototyping methodology of development;
- Why the prototyping methodology is particularly suitable for the development of this software;
- How the software was actually developed using this methodology.

Prepare a short report of about 500 words in your case-study. You should not be using the same material in your answer for part (1). You should also ensure that the source reference for this case-study is indicated in your answer.

[10 marks]

Semester 2 1998: Software Maintenance

The software maintenance activity is now recognised as an essential part of any software development life-cycle. The objective of this assignment is to introduce some understanding of how the development of software programs can have serious repercussions that only surface in the future after software implementation. In addition, a study would be made of the current perception of software maintenance in general software development.

Part (1) 30 marks

One of the most potentially serious problems threatening computer systems all over the world is the issue known as the *Year 2000 problem*, or the millennium bug. You are to undertake a study of this problem, and present a short report covering the following aspects of this problem:

- The nature of the problem;

- How the problem came about;
- The effects and consequences of not solving the problem;
- The technical solutions to solving the problem.

You should give equal weight to each of the sections. Your report for this part should be approximately 1400 to 1600 words.

Part (2) 10 marks

Software Maintenance has been defined in many ways. Pressman (Software Engineering: A Practitioner's Approach) defines it as activities applied to an existing system to keep it working in changing environments. However, in many modern IT organisations today, software maintenance jobs are frequently relegated and passed on to more junior programmers and developers in a software development department.

In a short report of approximately 450 to 550 words, describe why is maintenance frequently viewed as a "negative" activity and a "waste of time and resources" by management. In your answer, also describe and justify whether such a negative perspective is correct or should not be changed.

Semester 3 1998: Software Testing Techniques

The software testing activity is now recognized as an essential part of any software development life cycle. The objective of this assignment is to introduce some understanding and application ability of simple testing techniques. The assignment is broken into three sections:

Section A: Basis Path Testing	[10 marks]
Section B: Equivalence Partitioning and Boundary Value Analysis	[20 marks]
Section C: An objective view of software testing	[10 marks]

A total of forty marks is allocated for this assignment, and this forty marks will be later converted into a percentile figure reflected on your assignment cover sheet. You are required to read up ahead of the syllabus, particularly for sections A and B. At all stages of your working, particularly for sections A and B, be descriptive in your explanation of steps.

Section A: Basis Path Testing

```
main()    /* Start_of_program */
{
    If x > 10 then goto_subroutine A();
    else goto_subroutine B();
}

A()
{
    If y < 5 then x = x + 5;
    else then x = x - 5;
```

```
        goto_subroutine C();
    }

    B()
    {
        If z < 7, then
            {
                    y = y - 2;
                    z = z + 1;
                    goto_subroutine C();
            }
        else, then
            {
                     x = x + 1;
                    goto_subroutine D();
            }

    C()
    {
        goto_subroutine main();
    }

    D()
    {
        /* End_of_program */
    }
```

a) Based on the above section of pseudo-code, draw a ***flow graph*** representing program flow. [4 marks]

b) Based on your flow graph, calculate the value of ***V(G)*** using all of the three methods of calculation. Indicate all working necessary. [3 marks]

c) Based on your answer of V(G), indicate the ***basis set of test-paths***. Ensure that the order of test paths is indicated correctly. [3 marks]

Section B: Equivalence Partitioning and Boundary Value Analysis

a) Two additional testing techniques that are widely used today are the ***equivalence partitioning*** and ***boundary value analysis*** techniques. In your own words, discuss these two techniques. Provide also some of your own examples in the explanation of these techniques. [5 marks * 2 = 10 marks]

b) Make use of the following additional example. Consider the following data variable:

- Name of data variable: Age
- Description of data variable: the length of time that one has existed
- Data variable type: Integer
- Acceptable range: 25 to 60

For ***both*** equivalence partitioning and boundary value analysis, indicate the relevant details and all working during testing for both techniques. Justify your choice of test

cases. Indicate also examples of testing cases used in both techniques, and the expected results of each of these testing cases as well. [5 marks * 2 = 10 marks]

Section C: An Objective View of Software Testing

Software testing has frequently been considered to be the panacea to solve the problem of buggy software. However, software today is still often released and sold to the general public in buggy states, and fraught with errors. Based on your understanding of software testing, describe in about 700 words why this is so. In your answer, you should indicate the following:

- Current status of software testing: e.g. how much attention and time is spent on software testing.
- Limitations of software testing
- ***Why*** software is still being released in a buggy state [10 mark]

Semester 1 1999: Software Development and Methodologies

An understanding of how software has impacted our society would be important for any real appreciation of software development to take place. In this assignment, you are to study the effects of software on society, and also undertake a comparative study of different methodologies used in modern software development.

Section A (15 marks)

Many authors have referred to our age as the "Information Age". Making use of examples, discuss ***both*** the positive and negative effects on the impact of software on our society, e.g. whether and in what ways our lives have been improved or become worse through the use of software. Your discussion should be from 700 to 900 words.

Section B (25 marks)

The following are examples of software development methodologies:

- Linear Sequential models: e.g. the classic life-cycle;
- Prototyping models
- Rapid Application models
- Evolutionary Software process models.

You are to undertake a study of at least ***three*** development methodologies. You may wish to use examples of methodologies from the list above, or look for other types publicized in other literature. Upon your research, write a short report of approximately 1300 to 1600 words discussing both of the following:

a) A comparative analysis of the three development methodology you have selected;
b) When and under what circumstances each methodology would be most effective.

In order to facilitate your discussion, you should choose methodologies that are differing in many aspects, not methodologies that are similar. You should also note that a discussion of the steps in the methodology itself, though relevant in your report, should not be the main focus of your report.

Semester 2 1999: Software Metrics and Software Quality

A software metric is any type of measurement that can be related to a software system, the process of development or related documentation.

a) The use of Software Metrics is a component of software quality assurance. By using such measures, we better understand the attributes of the software that we create. And even more importantly, measurements can be used to assess the quality of the engineered products or systems that we build. Discuss (in about 700 to 900 words) in your own words the following aspects of software metrics: [15 marks]

- An overview of software metrics;
- How metrics actually aid us in understanding software, the advantages or limitations if any; its relationship to software quality.
- Whether metrics can be outdated;

b) An example of a fairly widely used but yet simple software metric is the Software Maturity Index (SMI) suggested by IEEE Std. 982.1-1988[2]. Make use of this metric in the following scenario:

A legacy system has 940 modules. The latest released required that 90 of these modules be changed. In addition, 40 new modules were added and 12 old modules were removed.

Now, compute the software maturity index for the system, clearly showing all steps of your working. In addition, provide a short commentary on the answer you have derived of approximately 300 words. [10 marks]

c) Another of the software technical metrics besides the Software Maturity Index used in software development concerns McCall' s Quality Factors, which discusses three important aspects of software quality. Write a short report of approximately 700 to 900 words on ***another*** software technical metric that you understand . In your short report, ensure that you discuss the following aspects:

A brief overview of the technical metric in your own words;
A commentary on its relevance, usability or practicality of the technical metric.

Ensure that you provide equal weight to both aspects in your discussion.
[15 marks]

Semester 3 1999: Programming Languages

This assignment concerns a structured study into various aspects and points of differences between various programming languages. The emphasis on this assignment is not on whether you can reproduce material already found in published material, but on whether you ***understand*** clearly these aspects and differences.

Question (a) (20 marks)

[2] A discussion for the Software Maturity Index metric is provided for in Software Engineering: A Practitioner' s Approach (Roger Pressman), 4th edition.

The following are statements frequently made when comparing 3rd and 4th generation languages:

- "4th generation languages lead to increased productivity, when compared to 3rd generation languages.
- "4th generation languages allow for greater user participation."
- "3rd generation languages are more machine efficient than 4th generation languages."
- "3rd generation languages are easier to test and maintain than 4th generation languages."

Based on your understanding, write a short report (of 1000 to 1200 words) comparing between the differences between 3rd and 4th generation languages. Ensure that your report clearly discusses the validity of ***each*** one of the four statements above.

Question (b) (10 marks)

Is there a movement from procedural languages to non-procedural languages in today's application context? Are procedural languages still as widely used as they are before? Discuss in 500 to 600 words, making use of examples where relevant.

Question (c) (10 marks)

A compiler translates all source code in a program to machine language instructions before any of the coded instructions are executed. For any one programming language, one is likely to find different compilers made by software houses. One factor of comparison, for example, is the quality of the output program produced by the compiler. Write a short report of 500 to 600 words, and discuss some factors in which compilers for programming languages can be compared. You should discuss at least three distinct factors.

Semester 1 2000: Software Requirement Analysis

"A complete understanding of software requirements is essential to the success of a software development effort. No matter how well designed or well coded, a poorly analyzed and specified program will disappoint the user and bring grief to the developer." [Pressman, 1997]

This assignment is concerned with various aspects to bring about a greater understanding of what exactly is Requirement Analysis all about, and how the specification plays a role in this activity.

The assignment consists of three parts, for a total of 40 marks.

Question (a) (15 marks)

We know from the study material that software requirements analysis is unquestionably the most communication intensive step in the software engineering process, given that it' s conducted between the developer and user, and also bases a lot on the quality of information gathered in the fact-finding phase. However, the communication path frequently breaks down between the developer and user. In fact, in some instances, workers in an organization may even fear the implementation of such automated systems within an organization, a possible result of poor communication.

Why does the communication path break down? Discuss in about 700 to 800 words, elaborating on at least three main reasons. [15 marks]

Question (b) (15 marks)

The term "requirements specification" is frequently used during this activity. Based on your understanding, firstly; clearly distinguish between both words "requirements" and "specifications."

Secondly, comment on why the requirements specification is a key component of requirement analysis. You may want to draw upon the possible repercussions or consequences in the event that the requirements specification is not correctly or accurately prepared. Discuss in about 700 to 800 words. [15 marks]

Question (c) (10 marks)

Assume that you've been tasked to prepare a training plan for a system analyst. This training is to be completed within the span of a four working days. Based on your understanding of what the responsibilities of an analyst are, work out this training plan. You should indicate the components of each day's training to some detail, clearly indicating the objective of that training component, and how it will assist the analyst in discharging his responsibility. [10 marks]

Semester 2 2000: Computer-Aided Software Engineering

"CASE can be defined as the disciplined and structured engineering approach to software and systems development. It emphasises structured methods, with defined and standardised procedures."

This assignment is concerned with various aspects to bring about a greater understanding of Computer-Aided Software Engineering tools. The objective of your report would be to provoke discussion on the implications of CASE, and more importantly whether, and how should CASE be implemented.

In your report of approximately 2000 to 2500 words, ensure you discuss the following aspects of CASE:

- What is CASE, and its relation to software development.
- Expected improvements of CASE;
- Potential risk areas, and an analysis of these risks;
- Limitations of CASE;

- An example of a CASE tool, and a short appraisal of this tool in each one of the four above aspects (e.g.. limitations, expected improvements)

You should take note that there it is insufficient to draw information for the entire assignment from a single source. There are enough varying perspectives on the use of CASE itself in software development, and you are strongly encouraged to read as widely as possible to address the requirements of the assignment.

Semester 3 2000: Software Maintenance

The software maintenance activity is now recognized as an essential part of any software development life-cycle. The objective of this assignment is to introduce some understanding of some real world implications of maintenance software programs. In addition, a study would be made of the current perception of software maintenance in general software development.

40 marks (100%)

Software Maintenance has been defined in many ways. Pressman (Software Engineering: A Practitioner's Approach) defines it as activities applied to an existing system to keep it working in changing environments. There are many issues with regards to the treatment of software maintenance in modern IT organizations today however.

Write a structured report of about 2000 to 2500 words on Software Maintenance Issues in the Real World. In your report, address the following issues:

1) Software maintenance is frequently viewed as a "negative" activity and a "waste of time and resources" by management. Describe and justify whether such a negative perspective is correct or should not be changed.

2) Software maintenance is often relegated and passed on to more junior programmers and developers in a software development department. What are your opinions on this?

3) What are some of the criteria that must be in place in order for successful maintenance to be instituted in organizations?

4) Lastly, what are other problems in relation to the conduct of software maintenance?

You should generally give equal weight in your report to each one of the four issues above. You should also take note that there it is insufficient to draw information for the entire assignment from a single source. There are enough varying perspectives on software maintenance in the context of software development, and you are strongly encouraged to read as widely as possible to address the requirements of the assignment.