Xday, XX XXX 2014.

9.30 am - 11.15am ***(check this!)***

University of Glasgow

DEGREES OF BEng, BSc, MA, MA (SOCIAL SCIENCES).

COMPUTING SCIENCE - SINGLE AND COMBINED HONOURS
ELECTRONIC AND SOFTWARE ENGINEERING - HONOURS
SOFTWARE ENGINEERING - HONOURS

SAFETY-CRITICAL SYSTEMS DEVELOPMENT

Answer 3 of the 4 questions.

1.

a) Briefly explain the relevance of verification and validation to the maintenance of safety critical software.

[6 marks]

[seen/unseen problem]

Verification tests the "truth" of particular requirements, in contrast validation establishes the worth or value of those requirements (1 mark). In other words, tests help to verify that an implementation or design meets a set of requirements (1 mark). Validation helps to ensure that the requirements are appropriate in the first place – we can verify that a system meets certain constraints but safety will still be compromised if we use a trivial or inappropriate set of requirements (1 mark). Verification and validation are applied throughout the design lifecycle (1 mark). They are particularly important during maintenance because changes to a system can invalidate the tests that were performed on an initial implementation – changes create the need for re-verification (1 mark). Equally maintenance may be in response to new requirements and these will also need to be validated (1 mark). Verification and validation can be more difficult at this stage in the lifecycle when the original development teams may not be on-hand to support this process (1 mark).

b) The UK Railways & Other Guided Transport Systems (Safety) Regulations 2006: Guide To The Application Of Safety Verification published by Her Majesty's Railway Inspectorate states that:

> "Safety Integrity Levels (SIL) are measures of the safety of a given process. Specifically, to what extent can the end user expect the process in question to perform safely, and in the case of a failure, fail in a safe manner? Usually applies to electronic systems particularly software and firmware architecture". (UK HMRI, 2007)

[6 marks]

Briefly explain how you would both validate the assignment of a SIL to a specific process AND how you might verify that the SIL has been achieved?

[unseen problem]

As mentioned, SILs are allocated to mitigations in proportion to the degree of risk that is reduced by those processes. Development resources are then allocation in proportion to the integrity level (1 mark); higher levels of implied risk reduction attract additional development resources (1 mark). Many companies use independent verification and validation teams (IV&V) (1 mark). These groups are to be kept at a distance from the development teams so that they can critically assess their work (1 mark). In some cases, they may even be outside of the company doing the design and development (1 mark). Where possible, regulatory assistance may also be used to validate SIL allocations. Incident reports within a safety management system might indicate areas where a SIL needs to be reassessed. Comparisons may be made across an industry etc (1 mark). Internal audits can also be conducted to verify that appropriate development tasks are being conducted for safety processes at a particular safety integrity level (1 mark).

c) The UK 2006 Regulations, cited above, expect that safety verification will be performed on behalf of a train operator by a 'competent person', defined as:

> "a person with sufficient practical and theoretical knowledge as well as experience of the particular task, plant, machine, procedure, equipment (etc) involved to enable them to thoroughly examine and identify any defects or weaknesses during examinations, and to assess their importance in relation to the safety, function and continued use of the plant,

machine, procedure, (etc again) and to be aware of their own particular limitations with regard to the task to be undertaken." (UK HMRI, 2007)

What problems arise when trying to identify a 'competent person' to be responsible for the verification of safety-critical software?

[8 marks]

[seen/unseen problem]

The 2006 regulations refer to attributes and not qualifications of a competent person (1 mark). Hence, there is considerable scope for subjective interpretation as to whether or not any individual can be classed as competent to conduct verification under this definition (1 mark). It can also be difficult for any individual to satisfy all of the criteria – for instance, there are relatively few software engineers who also have the requisite knowledge of the application domain; "particular task, plant, machine, procedure, equipment (etc) involved" (1 mark). In consequence, independent verification and validation consultancies tend to attract relatively high fees – especially considering the need to understand the requirements of software development standards (1 mark).

The caveat that a competent person must understand their own limitations is also a strong requirement given commercial pressures in many safety-critical industries (1 mark).

The importance of IV&V increases concerns over competency. If an individual is responsible for verification and validation then they can determine whether or not an implementation meets the necessary safety requirements (1 mark). If their competence is subsequently questioned then it may be necessary to re-examine the verification and validation of all code that was developed under the processes that they monitored. The financial consequences of this could be ruinous (1 mark).

In the future, it may be necessary to increase the use of professional examinations to determine competency. These schemes do exist at present but there is no international agreement on suitable qualifications (1 mark). This makes it difficult for a company in one country to be sure that software developed by another company in another country is developed to an appropriate level (1 mark).

2.

a) Traceability is a core concept in DO-178C, Software Considerations in Airborne Systems and Equipment Certification. It must be possible to trace from high to low level requirements. It must also be possible to trace requirements to code. Traceability applies more widely across the lifecycle. It is important to show that all major software components have been tested; this implies links between test results and source code. The amount of time and money invested in traceability is linked to the software level which is determined by the effects of a failure on the system.

Briefly summarise the problems of demonstrating traceability across complex safety-critical software.

[5 marks]

[seen/unseen problem]

Traceability relies upon a refinement relation that exists between a higher-level abstraction and the lower level representation. In some cases this relationship might be straightforward and amenable to formal reasoning – for example, if a model based approach is used then code can be automatically derived from a specification that meets particular requirements (1 mark). In other cases, the relationship is more complex – for example showing the relationship between a natural language statement and a more formal representation (1 mark). This may rely upon informal reasoning that can be subjective and open to mis-interpretation, what one analyst believes to be a complete and traceable refinement might not convince another analyst (1 mark). These problems are compounded by issues of scale – the more complex a system then the more difficult it can be to establish and maintain the correspondence between levels of representation (1 mark). Further problems arise under modification and maintenance where traceability may be jeopardized by changes (1 mark). The risk based approach mentioned in the final sentence of the question provides means of identifying key relationships that must be traceable – whereas others may be less critical to the safety of the system (1 mark).

b) DO-178C supports the use of model-based development; where abstractions including state-diagrams can be used to construct a high-level design that is then automatically transformed into executable code. This enables traceability between the high-level requirements expressed in the abstract representation and the implementation that can be automatically generated by appropriate tool support, for instance using Simulink's Code Inspector.

Identify five potential risks that might arise during the use of model-based development for safety-critical software engineering?

[5 marks]

[unseen problem]

There are many different solutions to this question – the following list provides a summary:

1. *Errors in the translation process.* The tools used to transform a higher-level model into a lower level 'implementation' may themselves contain errors and introduce failure modes.
2. *Lack of understanding in the refinement process.* Very often analysts become familiar with the higher level abstractions and over time lose the ability to work n the lower level representations, hence they lose the ability to identify problems such as that mentioned in part 1.
3. *Lack of experience and expertise across large projects.* In addition to the technical concerns, it can be difficult to manage the model based development of safety-critical systems because not enough engineers/managers are familiar with the approach.

4. *Higher-level requirements failure.* The use of model-based development does not eliminate the need to correctly identify appropriate requirements in the first place. If this is not done then model-based development will ensure the implementation of a faulty system.
5. *Bias towards the modeling language.* Higher-level abstractions offer particular support to particular types of systems. For instance, state diagrams provide powerful abstractions for analyzing sequences of interaction with complex, safety-critical systems. They are arguably less good at representing and reasoning about distributed, concurrent interaction across multiple components.
6. *Premature commitment to an implementation platform.* Many model-based development environments will offer a limited number of target platforms – for instance, deriving implementations in a subset of ADA or even C. The choice of a model based environment might then force designers to use a language that is supported by the tool rather than the one that might otherwise provide greatest support for a project.

As mentioned, other solutions are possible – for instance the issues in translating requirements from natural language into a formal representation and then doing the reverse – for instance, when a regulator is unfamiliar with a particular approach etc. (One mark for each risk up to the total of 5).

c) At the very highest levels of integrity, DO-178C requires Modified condition/decision coverage (MDCD) testing. This assumes that you must test each decision in a program (Boolean expression) to examine every outcome. Each condition must also be shown to independently affect the outcome of the decision.

Briefly explain why MCDC testing is both costly and time consuming.

[10 marks]

[seen/unseen problem]

MC/DC testing is part of a hierarchy of increasing rigor in testing – at each level the complexity and costs will increase (2 marks). In statement coverage testing, every executable statement in the program is invoked at least once during software testing. This shows that all code statements are reachable. However, it is a weak criterion because a Boolean test involving an OR condition may not consider every possible case in which the condition evaluates to TRUE – in other words, under statement coverage it would be sufficient to test A as true to execute C – without necessarily considering what would happen if A were false but B were true (2 marks).

```
if A or B then C
```

Decision coverage extends statement coverage to consider what would happen if the elements in a Boolean expression were false as well as true. This still does not ensure complete coverage in more complex Booleans. For the decision (A or B), test cases (TF) and (FF) will toggle the decision outcome between true and false. However, the effect of B is not tested; that is, those test cases cannot distinguish between the decision (A or B) and the decision A (2 marks). Condition coverage requires that each condition in a decision take on all possible outcomes at least once. In this case, for the decision (A or B) test cases (TF) and (FT) meet the coverage criterion, but do not cause the decision to take on all possible outcomes. Condition/decision coverage requires that there be sufficient test cases to toggle the decision outcome between true and false and to toggle each condition value between true and false. Using the example (A or B), test cases (TT) and (FF) would meet the coverage requirement. However, these two tests do not distinguish the correct expression (A or B) from the expression A or from the expression B or from the expression (A and B) (2 marks). MC/DC testing requires that each condition is shown to independently affect the outcome of the decision. For (A or B), test cases (TF), (FT), and (FF) provide MC/DC. For

decisions with a large number of inputs, MC/DC requires considerably more test cases than any of the coverage measures discussed above (2 marks). For additional detailed see http://shemesh.larc.nasa.gov/fm/papers/Hayhurst-2001-tm210876-MCDC.pdf

3.

a) Why is it important to avoid a culture of 'blame' in the aftermath of a major accident or incident?
[5 marks]

[unseen problem]

A number of authors, including Hollnagel and Dekker, have criticized the blame culture that can arise in the aftermath of an incident or accident (1 mark). They argue that a focus on liability can prevent stakeholders from participating in the recovery, including the subsequent investigation that is intended to avoid future recurrence by identifying the causes that led to a failure (1 mark). Individuals may be less likely to provide an honest and unbiased account of an adverse event if they know that they may be subsequently blamed for the loss (1 mark). There are further concerns related to crisis management – there is typically a high level of stress in the aftermath of an incident – hence individuals who feel responsible for a failure cannot always be relied on to resume the operation or management of a safety-critical system as they may be more prone to make subsequent errors (1 mark). This is an area of considerable debate – in the course we have discussed recent changes in Corporate Manslaughter changes –(1 mark) and the notion of proportionate blame, given that the relatives of the victims may argue strongly against a 'no blame' approach (1 mark).

b) The UK HSE defines Human Error to be an action or decision that was not intended. Identify five different causes of intended violations that commonly occur during the operation of safety-critical systems.
[5 marks]

[unseen problem]
There is a slight trick in this question – the opening sentence focuses on human error but the question asks about violations, which are *intended* actions in contravention of rules and procedures. A number of alternate answers are possible – the following list is a partial summary:

1. *Procedures did not apply.* A violation may occur when staff find themselves in a position where no procedures seem to apply; for example through lack of training or inadequate expertise. This confusion may lead operators to violating those rules that should hold in a given context.
2. *Procedures did apply but were perceived to be wrong.* Staff may deliberately choose to violate a rule or procedure if those rules or procedures are themselves perceived to jeopardize the safety of an application process. This happened in the Davis-Besse reactor incident;
3. *Procedures were contradictory.* In some cases, there may be two different rules or procedures that seem to apply in a given situation and the operator may be forced to violate one of them whatever he or she does.
4. *Production pressures.* Management pressure may lead operators to violate procedures in order to "get the job done";
5. *Staff lacked resources to implement procedures.* Procedures may require that staff have access to equipment, other operators, consumables, time etc that is not available when needed. In such situations, engineers will often find "workarounds" that violate normal operating requirements;
6. *Forgetfulness.* Operators may fail to follow an SOP that they previously knew through forgetfulness or distraction – possibly linked to the need for retraining;
7. *Informal operating procedures (deliberate neglect).* Over time, without adequate motivation and training, staff may rely on informal working practices that cut corners rather than follow particular rules or procedures.

For all of these reasons, procedures and rules are usually seen to be a relatively weak guarantee of system safety. (One mark for each cause of a violation up to a total of 5).

c) The HSE guidance on 'Reducing error and influencing behaviour' (2007) identifies fatigue as a key factor in human error leading to major accidents and incidents.

Write a brief technical report explaining to senior management the risks that fatigue creates for the users of safety-critical software. Expand your answer to recommend a number of measures to combat the effects of fatigue and identify Key Performance Indicators (KPIs) that might be used to assess the effectiveness of these measures.

[10 marks]

[unseen problem]

Fatigue plays a role in all of the major forms of human error – slips, lapses and mistakes (1 mark). It can also influence the violations described in part b) of this question. The influence of fatigue is compounded by the difficulty in ensuring that it does not affect safety. Many tasks inevitably involve shift work and the consequence concerns to mitigate the effects of circadian rhythms (1 mark). We have also discussed concerns over self-monitoring (1 mark) – including the US military studies that show our ability to predict periods of sleep decreases as we become more tired (1 mark).

In the course, we have discussed the influence of mitigations – including sleep discipline (1 mark), nutrition (protein and carbohydrate in improving wakefulness) (1 mark), the controlled use of "NASA naps" (1 mark). We have discussed the impact of sugar and of caffeine as well as amphetamines (1 mark). All of these mitigations carry associated risks. Epydrine has been associated with heart problems (1 mark). Caffeine can reduce fatigue but only for short periods of time, it can also cumulatively influence sleep patterns when an operator or engineer does find time to rest. This can, in turn, create further knock-on effects increasing fatigue over subsequent shifts (1 mark).

A number of KPIs can be used to assess the risks created by fatigue. One approach is to introduce secondary tasks –(1 mark) including queued response tasks where operators have to press a button as soon as possible after an alarm to check that they are awake (1 mark). – This technique is commonly used to slow trains if the driver does not respond within seconds of an alert.

Other techniques include the use of activity monitors (1 mark). These are becoming increasingly common with devices being transferred from health and exercise applications to provide data on control room behavior. In some cases, these devices have been integrated with stimulus response systems, described above. The KPIs in these cases would include the average response time to an alert (1 mark).

Direct monitoring techniques will measure physiological attributed of the staff include heart-rate and visual fixations (1 mark). These are relatively rare outside of military/space systems where there are less concerns over intrusion (1 mark). However, there continue to be attempts to introduce video cameras into cockpits so that management can monitor crew behavior including the identification of fatigue issues. These systems tend to rely on subjective analysis hence are less good for deriving KPIs that the direct physiological monitors (1 mark).

Finally, the KPIs may be linked to the automated analysis of errors (1 mark). The utility of fatigue management systems might be measured by monitoring the number of near miss incidents that occur at times of key concern – such as the dawn and dusk periods linked to circadian patterns (1 mark). An example of these incidents might be Signals Passed at Danger in rail applications – most of these do not result in an accident but they remain a safety concern linked to attention and fatigue (1 mark).

4.

The Boeing 777 uses a Fly-By-Wire flight control system. High levels of reliability are achieved using triple redundancy across all major hardware including the computing, power, hydraulic communication systems. The Primary Flight Computer (PFC) uses triple modular redundancy and N-version diversity; there are three channels; each contains three dissimilar computation lanes.

Write a technical report explaining the technical strengths and weaknesses of this approach. Explain how an appropriate safety-management system can ensure potential weaknesses are identified and resolved in the design of future aircraft, including updates to the 777.

[20 marks]

[structured essay/unseen problem]

This is an open-ended essay question. The description of the 777 focuses on two issues: redundancy and diversity.

Redundancy is primarily used to mitigate hardware fault distributions; we have shown the bath-tub curves in the course and analyzed both burn-in and wear-out times (2 marks). In hardware systems, these are fewer concerns over common-cause failures (1 mark). Hence, the use of triple redundancy makes it less likely that a hardware failure will affect two of the three components. I expect students will provide a diagram illustrating triple modular redundancy with multi-level extensions showing multiple redundant voting elements (for up to 4 marks).

There are important limitations with the application of redundancy for software applications because common cause failures are a far greater concern (1 mark). If a piece of code contains a bug and that code is executed then under deterministic models of execution it will fail (1 mark). Providing redundant backup implementations of the same code provides no benefits. The duplicate will fail in the same way unless its execution is affected by the failure of the primary system (1 mark). In consequence, as mentioned in the question design diversity is use through N-version programming (1 mark). It is important to mention that the combination of diversity and redundancy may also extend to hardware – especially as the gap between software and hardware closes through the use of FPGAs – although this is not so much an issue for the B777, it is important in other application areas, including the nuclear industry (2 marks).

A host of further issues complicate the application both of redundancy and of diversity. Redundant systems inevitable increase the complexity of an application – especially when cold stand-by systems need to be brought on-line or where a hot stand-by needs to be integrated through adaptive voting systems (2 marks). The additional hardware increases the weight (1 mark) and power consumption (1 mark) and cooling requirements (1 mark) of on-board systems that are concerns for commercial aircraft. In addition, there are significant costs for design, certification, maintenance etc (1 mark). Diversity raises further questions, especially for software. The use of common libraries means that two or more diverse implementations may be using the same flawed components. In other cases, faults can cluster in complex areas of the coding even though two or more sets of coders are involved (1 mark). Some companies have argued that these issues undermine the case for redundancy – it may mean taking a budget and dividing into N leaving insufficient resources for each implementation. In consequence, it may be better to spend all the resources on developing and testing one "good version" of the code (1 mark). These arguments remain controversial bit it is important to recognize the costs of N version programming do not end at delivery – staff must maintain and configure all of the diverse code during the remainder of the lifecycle, hence as mentioned above these techniques may be isolated within the primary flight systems (1 mark).