TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS – 2018

TRIMESTER 1

SWEN 326

SAFETY-CRITICAL SYSTEMS

Time Allowed: TWO HOURS

CLOSED BOOK

Permitted materials: No calculators permitted.
Non-electronic Foreign language to English dictionaries are allowed.

Instructions: Answer all questions

Question	Topic	Marks
1.	Risk, Hazards and Failure	30
2.	Design Validation	30
3.	Software Testing	30
4.	Static Analysis	30
	Total	120

1. Risk, Hazards and Failure **(30 marks)**

   (a) Consider the following description of a system for controlling *lift doors*.

   "The lift doors are operating by a *software controller*. After the doors have been *open* a certain amount of time, they automatically close. If a *sensor* detects an *obstruction* as the doors are closing, they reopen. Under no circumstance should the doors close on an obstruction."

   i. **(2 marks)** Following the terminology of IEC61508, identify the *Equipment Under Control* for the door system.

   ii. **(2 marks)** Identify an important *hazard* for the door system.

   iii. **(4 marks)** Briefly, discuss how the *risk* of the above hazard occurring might be estimated.

   iv. **(2 marks)** Briefly, discuss how the above hazard is *mitigated* in the system.

v. **(4 marks)** Under IEC61508 there can never be *zero risk*. Briefly, discuss what this means with respect to the door system.

vi. **(2 marks)** When calculating risk, one will often consider only a *single component failure*. Briefly, discuss why this approach is sensible in many cases.

vii. **(2 marks)** Briefly, outline one simple approach for mitigating against a single component failure.

viii. **(4 marks)** Briefly, discuss why it is difficult to estimate the likelihood of a software sytem failing.

(b) The *"Power-of-Ten"* rules provide a simple set of coding guidelines for developing safety-critical software.

i. **(4 marks)** Rule 4 states that *"No function should be longer than what can be printed on a single sheet of paper"*. Briefly, discuss the motivation behind this rule.

ii. **(4 marks)** Rule 10 states that *"all compiler warnings must be enabled"*. Briefly, discuss the advantages and disadvantages of this rule.

SPARE PAGE FOR EXTRA ANSWERS

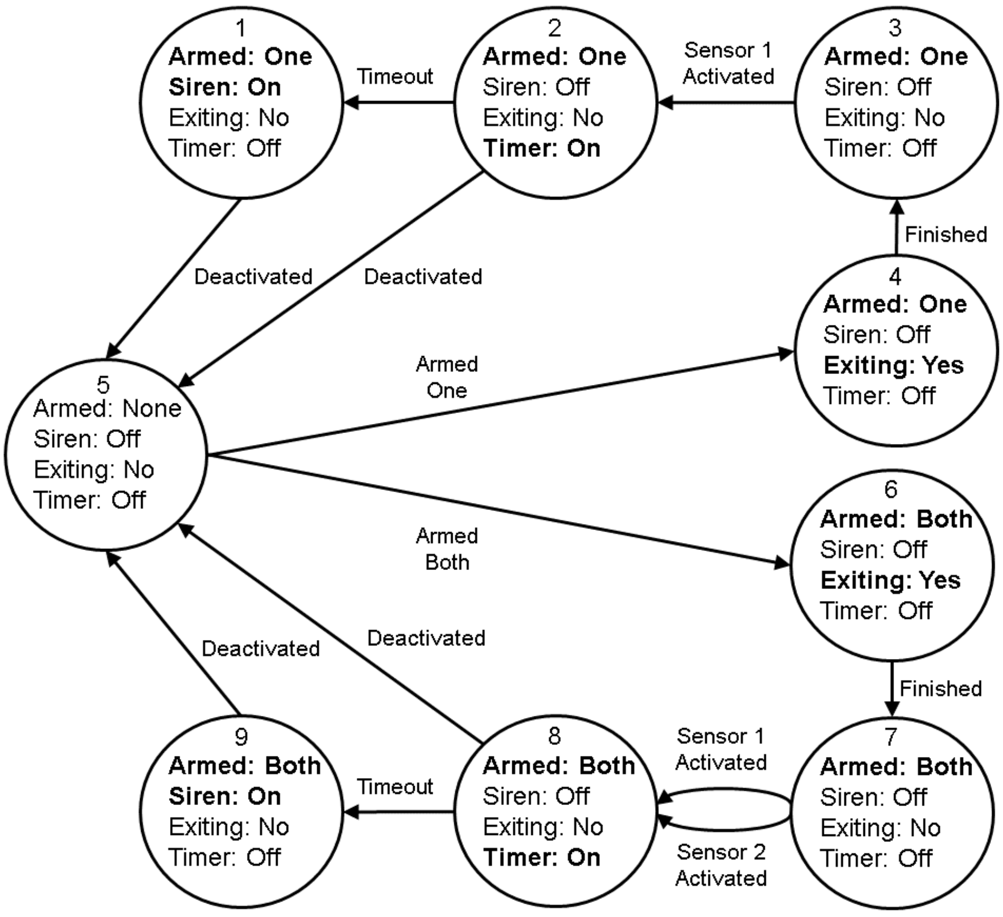
Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

2. Design Validation **(30 marks)**

Consider the following description of a simple *alarm system* with timers:

> "The alarm system protects houses against break-ins. The system has two *sensors* which detect movement in the home. The owner can *arm* one or both of the sensors. When no-one is home, both sensors should be armed. At night, the downstairs sensor should be armed so the owner can move around upstairs. When one or more sensors is armed, the system begins an *exit timer*. Whilst the exit timer is active, the *siren* will not sound. Once exiting is complete, the system is fully armed. When an armed sensor detects movement, an *alarm timer* begins. This gives time for the owner to deactivate the system when returning home. However, if the alarm is not deactivated before time runs out, the siren will sound and continue until the system is deactivated."

A *state machine diagram* for the alarm system has been provided:



(Question 2 continued on next page)

(Question 2 continued)

(a) For each of the following statements, indicate whether it is a true or false statement based on the state machine diagram.

i. **(2 marks)** Sensor 2 may be armed whilst sensor 1 is not.

ii. **(2 marks)** The siren may sound when the exit timer is active.

iii. **(2 marks)** When the siren is deactivated, the system remains armed.

iv. **(2 marks)** A *timeout* does not always result in the siren being activated.

(Question 2 continued on next page)

(Question 2 continued)

(b) Consider the following *incomplete* Alloy model of the alarm system:

```
enum Bool {True, False}
enum Sensor {None, Both}

sig AlarmState {
 armed : Sensor, siren : Bool, exiting : Bool, timer : Bool
}

pred init(s : AlarmState) {
 s.armed = None and s.siren = False and s.exiting = False and s.timer = False
}

pred armSystem(s,s' : AlarmState) {
 s.armed = None
 s'.armed = Both and s'.siren = False and s'.timer = False and s'.exiting = True
}

pred finished(s,s' : AlarmState) {
 s.exiting = True
 s'.armed = Both and s'.siren = False and s'.timer = False and s'.exiting = False
}

pred sensorActivated(s,s' : AlarmState) {
 s.armed = Both and s.siren = False and s.timer = False and s.exiting = False
 s'.armed = Both and s'.siren = False and s'.timer = True and s'.exiting = False
}

pred timeOut(s,s' : AlarmState) {
 s.timer = True
 s'.armed = s.armed and s'.siren = True and s'.timer = False and s'.exiting = False
}

pred deactivated(s,s' : AlarmState) {
 s.armed = Both
 s'.armed = None and s'.siren = False and s'.timer = False and s'.exiting = False
}

pred transition(s1, s2: AlarmState) {
  armSystem[s1,s2] or finished[s1,s2] or sensorActivated[s1,s2]
                   or timeOut[s1,s2] or deactivated[s1,s2]
}

sig ExecutionTrace { states: seq AlarmState }{
  init[states[0]] and all i: states.inds | i > 0 implies transition[states[i−1], states[i]]
}
```

(Question 2 continued on next page)

(Question 2 continued)

i. **(2 marks)** Give an instance of `AlarmState` which corresponds to state eight from the diagram on page 6.

ii. **(2 marks)** Give an instance of `ExecutionTrace` which corresponds to the execution trace “5 → 6 → 7” from the diagram on page 6.

iii. **(2 marks)** What key functionality is provided in the state machine diagram but not by the Alloy model?

(Question 2 continued on next page)

(Question 2 continued)

iv. **(2 marks)** Can the system described by the Alloy model be deactivated whilst exiting is in progress?

v. **(2 marks)** How many different instances of `ExecutionTrace` are possible with exactly *three* states?

vi. **(2 marks)** How many different instances of `ExecutionTrace` are possible with exactly *four* states?

vii. **(4 marks)** Suppose we add "`exiting = True implies armed = Both`" as an invariant on `AlarmState`. Briefly, discuss whether or not this changes the number of valid instances of the model.

(c) Consider the following implementation for the Whiley function `zeroOut()`:

```
function zeroOut(int[] items, int start) -> (int [] r)
ensures |r| == |items|
ensures all { k in start..|r| | r[k] == 0 }
ensures all { k in 0..start | r[k] == items[k] }:
   //
   int i = start
   int[] oitems = items
   //
   while i < |items|:
      items[i] = 0
      i = i + 1
   //
   return items
```

i. **(2 marks)** Provide a *precondition* for `zeroOut()` which will allow it to verify:

ii. **(4 marks)** Provide a *loop invariant* for `zeroOut()` which will allow it to verify.

3. Software Testing **(30 marks)**

(a) **(5 marks)** Briefly, discuss the advantages and disadvantages of *testing* as a mechanism for ensuring correctness of safety-critical software.

(b) Consider the following Java class which compiles without error:

```java
public class Util {
   /**
    * Return the maximum element of an array.
    */
   public static int max(int[] items) {
      //
      int m = items[0];
      //
      for(int i=1;i!=items.length;i=i+1) {
         if(m < items[i]) {
            m = items[i];
         }
      }
      //
      return m;
   }
}
```

i. **(2 marks)** Briefly, discuss why "`new int[0]`" should not be considered a valid test input.

ii. **(6 marks)** Draw the *control-flow graph* for the `max(int[])` method.

iii. **(3 marks)** Give test inputs which achieve 100% *branch coverage* of `max(int[])`.

iv. **(3 marks)** Briefly, discuss why the *Modified Condition/Decision Coverage (MC/DC)* criteria does not improve upon branch coverage in this case.

v. **(3 marks)** Give test inputs which achieve 100% *prime path coverage* of `max(int[])`.

vi. **(3 marks)** The `max(int[])` method does not contain any *unreaslisable paths*. Briefly, discuss what this means.

(c) **(5 marks)** *Fuzz testing* uses random input data to automatically generate test cases. Briefly, discuss why fuzz testing of methods in Java is challenging.

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

4. Static Analysis **(30 marks)**

(a) This question is concerned with *static analysis*.

i. **(3 marks)** Briefly, discuss what is meant by the term static analysis.

ii. **(3 marks)** Briefly, discuss how the use of static analysis can help to find errors in software.

iii. **(3 marks)** Briefly, discuss what *conservatism* means in the context of static analysis.

(b) This question is concerned with *non-null* analysis.

i. **(2 marks)** Briefly, discuss what the `@NonNull` annotation means.

ii. **(2 marks)** Briefly, discuss whether or not `@NonNull` is a *subtype* of `@Nullable`.

iii. **(6 marks)** For each *parameter*, *return* and *field* in the following program, insert `@NonNull` or `@Nullable` annotations (where appropriate) by writing in the box.

```
public class Property {

  private String name; // Every property has a name

  private Player owner; // Some properties have owners

  private boolean mortgaged; // Some properties are mortgaged

  public Property(String name) {
     this.name = name;
  }

  public String getName() { return name; }

  public Player getOwner() { return owner; }

  public void setOwner(Player p) { owner = p; }

  public boolean isMortgaged() { return mortgaged; }

  public void setMortgated(boolean m) {
     mortgaged = m;
  }
}
```

Student ID: ....

(c) This question is concerned with Java's *definite assignment* analysis.

i. **(2 marks)** Briefly, state the purpose of checking definite assignment.

Consider the following Java program.

```
int max(int x, int y) {
   int r;
   //
   if(x >= y) { r = x; }
   if(x < y) { r = y; }
   //
   return r;
}
```

ii. **(5 marks)** This program fails *definite assignment*. Briefly, discuss why this happens. Your answer should illustrate the program's *control-flow graph*.

iii. **(4 marks)** The above program is a *false positive* with respect to definite assignment analysis. Briefly, discuss what this means.

* * * * * * * * * * * * * * *

Student ID: . . . .

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Student ID: . . . .

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.