The University of Melbourne

SWEN90010: High Integrity Software Engineering

Assignment 4

Due Date: 11:59pm, Sunday 29 May, 2015

1 Introduction

This handout is the assignment 4 sheet. The assignment is worth 10% of your total mark and is done in pairs (the same pairs as assignment 2 and 3).

The aim of this assignment is to use SPARK to implement the Alloy model that you produced for assignment 3 and verify this using the SPARK toolset. The assignment evaluates your ability to apply safe-programming subset techniques to implement a high-integrity system, to apply design-by-contract and proof to verify your implementation, and to assess the correctness of your implementation with respect to the requirements.

The assignment should be done in pairs, with careful planning around which member of the pair does which task.

2 Your tasks

The assignment asks you to write your Alloy model from assignment 3 as preconditions and post-conditions in SPARK, implement these, and then compare your assignment 1 implementations to this new implementation.

1. Team member one: Derive a SPARK package interface (including preconditions and post-condition) for your Alloy model from assignment 3 (although you are permitted to fix issues with your Alloy model).

   As in assignment 3, part of a solution is given — the same operations provided as a sample in assignment 3 are modelled and implemented in SPARK, and the GPS tool can be used to prove that the implementation match the preconditions and postconditions.

   NOTE: If you feel that your Alloy model contains some idiosyncrasies that make it exceedingly difficult to model as SPARK preconditions and postconditions, please talk to Tim, and together you can negotiate a modification so that you do not spend too much time on small problems.

2. Team member two: Implement a SPARK package body for this package. This may be based on your implementation from assignment 1, but must conform to the SPARK contract from Task 1.

3. Together: Once the SPARK implementation and SPARK contracts are written, use the SPARK tools to verify whether your program conforms to its contract.

   HINT: You should first try this for a few simple operations, rather than trying to put all this together in one hit. Then, repeat incrementally for other operations.

Note: you should fix the faults found in your SPARK contracts and implementation, as some errors may be masking others. You may choose to ignore some warnings if you believe that the implementation is correct despite the warnings – the SPARK tools are not fool proof. However, this is suggested only if you are truly convinced – contracts and implementations that are inconsistent will be marked down.

4. Compare the integrity of the verified and tested SPARK implementation against the two solutions produced for assignment 1. Which is the higher integrity system *and why*? Argue a case for your answer. In your argument, you should consider the measures taken over the course of assignments 2, 3 and 4. Limit your argument to one A4 page.

5. Compare the designs (data structured and algorithms) of the SPARK implementation against the two solutions produced for assignment 1. Are there fundamental differences? If so, why? Limit your answer to one A4 page.

   For these last two tasks, what we are looking for is an understanding of the theory and practice of the subject. Thus, to keep your answers to one A4 page in each task and get good marks, you will need to think about the answers.

3 Criteria

Criterion	Description	Marks
SPARK Implementation [3 marks]		
Correctness & completeness	The SPARK implementation correctly and completely implements the Alloy model.	1 mark
SPARK	The SPARK implementation uses suitable SPARK features to ensure the integrity of the system.	1 mark
Clarity and code formatting	The implementation is clear and succinct. The implementation adheres to the code format rules in Appendix A.	1 mark
SPARK Contracts [3 marks]		
Correctness & completeness	The SPARK preconditions and postconditions are correct, complete, and clear.	2 marks
Consistency	The SPARK contracts are consistent with the Alloy specification from assignment 3.	1 mark
Discussion [4 marks]		
Completeness	All aspects related to the case are considered.	1 mark
Understanding	The argument links to relevant theory, and demonstrates an understanding of that theory, and application of that theory to engineering high-integrity systems.	3 marks
Total		10 marks

4 Submission

Submit the assignment using the submission link on the subject LMS. Go to the SWEN90010 LMS page, select *Assignments* from the subject menu, and then select *View/Complete* from

the *Assignment 4 submission* item. Following the instructions, upload a zip file containing the following:

1. A directory called `code/`, which contains all code for the system.
2. A PDF file containing your answers to tasks 4 and 5.

Only *one* student from the pair should submit the solution, and the submission should clearly identify both authors.

Late submissions Late submissions will attract a penalty of 1 mark for every day that they are late. If you have a reason that you require an extension, email Tim *well before the due date* to discuss this.

Please note that having assignments due around the same date for other subjects is not sufficient grounds to grant an extension. It is the responsibility of individual students to ensure that, if they have a cluster of assignments due at the same time, they start some of them early to avoid a bottleneck around the due date. Further, giving an extension in these circumstances simply crowds out assignments and exams after this date.

5 Academic Misconduct

The University misconduct policy applies to all assessment. Students are encouraged to discuss the assignment topic, but all submitted work must represent the individual's understanding.

The subject staff take plagiarism very seriously. In the past, we have successfully prosecuted several students that have breached the university policy. Often this results in receiving 0 marks for the assessment, and in some cases, has resulted in failure of the subject.

Appendix

A Code format rules

The layout of code has a strong influence on its readability. Readability is an important characteristic of high integrity software. As such, you are expected to have well-formatted code.

A code formatting style guide is available at `http://en.wikibooks.org/wiki/Ada_Style_Guide/Source_Code_Presentation`. You are free to adopt any guide you wish, or to use your own. However, the following your implementation must adhere to at least the following simple code format rules:

- Every Ada package must contain a comment at the top of the specification file indicating its purpose.
- Every function or procedure must contain a comment at the beginning explaining its behaviour. In particular, any assumptions should be clearly stated.
- Constants and variables must be documented.

- Variable names must be meaningful.
- Significant blocks of code must be commented.

  However, not every statement in a program needs to be commented. Just as you can write too few comments, it is possible to write too many comments.
- Program blocks appearing in if-statements, while-loops, etc. must be indented consistently. Tabs or spaces can be used, as long as it is done consistently.
- Lines must be no longer than 80 characters. You can use the Unix command "`wc -L *.ad*`" to check the maximum length line in your Ada source files.