The University of Melbourne
School of Computing and Information Systems
SWEN90004 Modelling Complex Software Systems

Assignment 1b, 2020

Released: Monday 30 March. Deadline: 11:59pm, Monday 20 April.

Objective

To use a higher-level modelling language to specify and reason about a concurrent system.

Background and context

Assignment 1 has two parts. The first part, 1a, was worth 10% of your final mark; this part, 1b, is also worth 10%. In the first part (which you should complete before attempting this part) you designed and implemented (in Java) a simulation of King Arthur and the Knights of the Round Table. Now the task is to model the system in FSP, use LTSA to check it, and to identify and mitigate any problems that you discover through modelling.

The tasks

1. **Model:** The first task is to model your implementation from Assignment 1a in FSP. That is, reverse engineer an FSP model from your Java implementation. Your model should contain comments that explain the design and its components. NB: If your implementation in Assignment 1a was incomplete, or too incorrect to form the basis of an FSP model, you may wish to construct your model based on the system description in the Assignment 1a specification.

2. **Check:** Specify what you believe are relevant safety and liveness properties for your FSP model. *Note: for liveness, it may be the case that very few properties are required.* Use LTSA to check these properties.

3. **Modify**: You will most likely have discovered, either while completing Assignment 1a, or while completing Tasks 1 and 2 of this Assignment, that there is a potential deadlock issue.)

   Propose at least one modification to the design of the system that mitigates this problem and update your FSP model.

   Be sure to create a copy of the file containing your original FSP model. You are required to submit **both** *the original model from Task 1 and the modified model from Task 3.*

   If you discover any additional problems with your original Java implementation, you should also correct these in your FSP model. You may *optionally* also update your Java implementation to match the modified model (for practice, not assessed).

4. **Discussion:** Discuss your original and modified models. Points you *may* wish to address include: What (if any) problems did you find in your original model as a result of using

LTSA? Were these problems also in the implementation? If so, why do you think you picked these up now and not before submitting Part 1a? What was the reasoning behind the changes you suggested in your modified model? Are there any trade-offs in system performance associated with your suggestion?

If you did not find problems with your original implementation, were you convinced when you submitted Part 1a that no problems existed? Why did you believe this? Do you still believe there are no problems?

Keep your discussion to no more than 500 words.

Procedure and assessment

The assignment should be completed by students individually. A late submission will attract a penalty of 1 mark for every calendar day it is late. If you have a reason that you require an extension, email Nic *well before the due date* to discuss this.

To tackle the assignment, first work through (and *understand*) the examples from lectures, and do the workshop exercises. FSP is not difficult—it is simpler than most programming languages, and much simpler than a language like Java. However, as with other languages, the way to master it is to use it, and to learn by doing. Trying to do the assignment straight up may mean you struggle. Work through some easier examples first.

Submit a single zip file via the LMS. The file should include

- A file called `model.lts` with your initial FSP model, including the safety and liveness properties from Task 2.
- Your modified/corrected FSP model `model_modified.lts`, including the safety and liveness properties from Task 2.
- A plain text file `discussion.txt`, containing the discussion of issues. Please ensure that this is a *plain text* file; ie, not a `doc`, `docx`, `rtf`, or other file type that requires specific software to read.

All model files and your text file should contain, near the top of the file, your name and student number.

We encourage the use of the LMS's discussion board for discussions about the project. However, all submitted work is to be your own individual work.

This project counts for 10 of the 50 marks allocated to project work in this subject. Marks will be awarded according to the following guidelines:

Criterion	Description	Marks
Clarity, abstraction	FSP models are at a suitable level of abstraction. All behaviours relevant to interaction are specified, and there is sufficient detail to implement the system from the model. Descriptions of all actions and processes are clear and concise.	2 marks
Completeness	The model is complete. All components have been modelled and all expected behaviour is present. Suitable safety and liveness properties have been described.	2 marks
Correctness	The original model accurately reflects the original implementation (or specification). The modified model behaves "correctly" (ie, is free of problems identified in the original model), does not violate any safety properties, and demonstrates all liveness properties.	3 marks
Formatting	The FSP source adheres to the code format rules from Part 1a where this makes sense, including the use of comments to document model components and properties.	1 marks
Discussion	The discussion demonstrates understanding of the subject material.	2 marks
Total		10 marks

Nic Geard
26 March 2019

Why backwards?

A valid question: Why are we modelling the system *after* implementing it? Should it not be done the other way? Well, yes and no. Many people use modelling to understand an existing code base (just look at the number of tools for reverse engineering UML models from code bases). Reverse engineering is a great way to understand problems with an existing system; for example, why a deadlock is occurring. It is true, however, that in most cases, it would be cheaper and easier to do the modelling first.

The other reason why the assignment is "backwards" is that trying to model a new system using a new type of notation, such as FSP, will often end in disaster. We hope that, having gone through the Java programming stage, you feel familiar with the system to be modelled and thus can concentrate on the use of FSP. The exercise should be one of applying abstraction—a skill that is of utmost importance in any engineering discipline.