The University of Melbourne

SWEN40004: Modelling Complex Software Systems

Assignment 1

First Semester, 2015

Due Date: 11:00am, Monday 20 April, 2015

1 Introduction

This handout specifies assignment 1, which is worth 25% of your final mark.

This project provides an opportunity for you to develop your skills in developing a computational models of a complex system, using the model to run experiments, and reporting both the design of the model and the results of the experiments. Specifically, your task is to replicate an existing grid-based NetLogo model in Java, perform experiments to verify that its behaviour matches that of the original model, and prepare a report on your findings.

It is expected that you will work on this Assignment in pairs.

2 Motivation

"Replication is a critical component of the scientific method and a core practice of scientists."

Wilensky & Rand (2007), *JASSS* 10(4):2

The idea that a scientific experiment should be reproducible in order to be credible has a long history in science, dating back to early Greek philosophers. Because computational models of complex systems are used as the basis of scientific claims about the behaviour of those systems, it is essential that they are subject to the same level of rigorous evaluation.

Replication of a computational model demonstrates that the results of the original model were not an exceptional occurrence, and can help to increase our confidence in the validity of its behaviour. Replicating a model in a different computer language to the original model can help ensure that model behaviour is independent of any implementation details specific to a particular programming language.

3 Process

1. Select a model from the NetLogo Model Library (see next section for a pre-approved list). You are welcome to choose another model to replicate, however please discuss this with Nic for approval first.
2. Explore the behaviour of the NetLogo model. How does it work? What are the behaviours that are captured by the model? Which outputs are measured? What are the limitations of the model?

3. Use your software engineering skills, along with what you have learned so far in the subject, to design and implement an identical model in Java.

4. Experiment with your new model. Can you recapture the same behaviours as the original NetLogo model? Why/why not?

5. Write a report detailing the motivation for the model, the design of your implementation, results of the experiments you ran to compare the behaviour of the two models, and a discussion of your findings. The marking schema below provides an indication of the content expected in your report.

You should start by implementing the simplest possible prototype (and simulation experiments) and ensure that it works well before proceeding with more complex designs. You must investigate the effects of model parameters on model behaviour. Appropriate analysis of the results from both models is expected.

Please note marks will **not** be allocated for the development of new libraries or GUI interfaces.

By the start of week 4, you should have chosen the topic, experimented with the NetLogo model, done some background reading on model development, started thinking about the design of your model, and developed a preliminary set of simulation experiments to be performed.

4 Possible models

These models are available from the NetLogo Model Library `http://ccl.northwestern.edu/netlogo/models/`:

- Ethnocentrism (Social Science)
- Rebellion (Social Science)
- Wealth Distribution (Social Science)
- Daisy World (Biology)

There are a range of other models in the Model Library, and if you would like to choose a model *of an equivalent level of complexity to the above models*, you are welcome to discuss with Nic.

5 Resources

- Netlogo: `http://ccl.northwestern.edu/netlogo/`
- Additional Netlogo tutorials: `http://cfpm.org/simulationcourse/`

There are some good papers in the *Journal of Artificial Societies and Social Simulation* on replication of Agent-Based Models. While these papers go into greater depth than you are expected to for this project, they may serve as a useful guide to designing experiments and describing your model design.

- Uri Wilensky & William Rand (2007) Making Models Match: Replicating an Agent-Based Model, *Journal of Artificial Societies and Social Simulation* 10(4):2. `http://jasss.soc.surrey.ac.uk/10/4/2.html`
- Wolfgang Radax & Bernhard Rengs (2010) Prospects and Pitfalls of Statistical Testing: Insights from Replicating the Demographic Prisoner's Dilemma. *Journal of Artificial Societies and Social Simulation*, 13(4)1. `http://jasss.soc.surrey.ac.uk/13/4/1.html`

6 Submission

6.1 Proposal

The proposal is to be submitted via LMS by 11:59pm, Monday 23 March. You will be expected to bring copies of your proposal to the workshop on 25/26 March where feedback will be given from the teaching staff as well as your peers.

The proposal is expected to be 1-3 pages (12pt font, reasonable margins) and contain:

- a descriptive overview of model you are replicating (eg, purpose, users);
- the design of the existing model (eg, states, update rules);
- the design of your model;
- the experiments you intend to run (optionally, some results from the NetLogo model or early results from your model).

The proposal is worth 0 marks, however failure to submit by the deadline will incur a 1 mark deduction in your final mark. The proposal will constitue a 'first draft' of your report for final assessment, and the interim submission is a valuable opportunity for you to get early feedback.

6.2 Final project

We will use the LMS for project submission. There are two separate submissions to be made:

1. a PDF copy of your report as a TurnItIn submission:

   The report format should follow Springers Lecture Notes in Computer Science (LNCS) conference proceedings format. See: `http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0`. The report must be no longer than 1500 words and 8 pages maximum (including graphs) – marks will be deducted for longer reports or for not following the formatting guidelines.

2. a zip file as a regular project submission via LMS containing:

   - all source code
   - any scripts required to run the experiments documented in your report
   - clear instructions describing how to run your model

Code compliant with Java SE 7 will be accepted, however Java SE 6 is preferred.

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 Nicole *well before the due date* to discuss this.

Note that late or no submission of a proposal in week 4 will also incur a 1 mark deduction as described previously.

7 Criteria

There are two components: the final report (15%) and the quality of your solution/code to an appropriately challenging complex systems problem (10%).

7.1 Report

Criterion	**Description**	**Marks**
Aim/objectives	You have clearly stated the aims and objectives of your study.	1 mark
Background	You have provided an appropriate review of background material on your chosen model. You have described why the system modelled is a "complex system". You have explained why this domain is of interest. The bibliography is complete and correctly formatted.	2 marks
Model design	You have described the complex systems modelling technique(s) used in the investigation in sufficient detail. This includes appropriate references to the modelling platform used and the technique(s) more generally. You must describe your model design and implementation.	3 marks
Experimentation	You have designed and implemented appropriate experiments to explore and compare the behaviour of your Java model and the original NetLogo model. You have described a range of scenarios (parameter settings) used in your study. You should include a discussion of why you chose the simulation experiments listed.	3 marks
Results	You have clearly presented the results of your investigation using appropriate tables, graphs and statistical analysis.	3 marks
Discussion/conclusion	You have discussed the results of your experiments, and the outcome of the replication exercise.	2 marks
Writing	Your writing is well-expressed, clearly proof-read and demonstrates a coherent development of ideas.	1 marks
Total		15 marks

7.2 Solution/code quality

Criterion	Description	Marks
Design	The design of the model is of high quality and is potentially extensible	2 marks
Correctness	The results of the model are similar to the original model OR it is justified why the results are different.	2 marks
Clarity	The design of the model is clear and succinct.	2 marks
Code formatting	The implementation adheres to the code format rules (Appendix A).	2 marks
Executability	The submitted code runs and generates results consistent with the results listed in the report. This implies that you will submit clear instructions describing how to run your model.	2 marks
Total		10 marks

8 Academic Misconduct

The University misconduct policy applies — see the LMS for a statement of the expectations. You are reminded that all submitted project work in this subject is to be your own individual work – or if working with a partner, your groups work.

I have encouraged you to review scientific papers related to the complex systems model you select. In many circumstances, you will find the code used to implement the models on-line. Obviously, the code you submit must be your work. However, if you make use of code snippets that you find on-line, please provide an appropriate references both in the source code of your model and in your report.

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.

A Code format rules

Your implementation must adhere to the following simple code format rules:

- Every Java class must contain a comment indicating its purpose.
- Every method must contain a comment at the beginning explaining its behaviour. In particular, any assumptions should be clearly stated.
- Constants, class, and instance variables must be documented.
- Variable names must 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-statements, etc., must be indented consistently. They can be indented using tabs or spaces, and can be indented 2, 4, or 8 spaces, as long as it is done consistently.
- Each line should contain no more than 80 characters.