The University of Melbourne

SWEN90004: Modelling Complex Software Systems

Assignment 2

First Semester, 2025

Proposal Due Date: 11:59pm, Friday 9 May, 2025

Final Report Due Date: 11:59pm, Friday 30 May, 2025

1 Introduction

This handout specifies Assignment 2, which is worth 25% of your final mark.

Your task is to replicate an existing grid-based NetLogo model in Java or Python, perform experiments to verify that its behaviour matches that of the original model, adapt your model to answer a new research question, and prepare a report on your findings. It is expected that you will work on this Assignment in **groups of three**. You should establish groups as soon as possible (using your contacts or the Canvas Discussion Board).

NB: Let me (Artem) know by the end of May 2, 2025, if you have not found a group, and I will randomly assign remaining students to groups.

The objectives of this Assignment are to provide you with the opportunity to develop your skills in implementing a computational model of a complex system, using it to conduct experiments, reporting on the design of the model and the results of these experiments, and working in a group context.

2 Motivation: Replication

"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, helps to increase our confidence in the validity of its behaviour, and is a first step toward extending the model to address a novel question. Replicating a model in a different computer language to the original model can also help ensure that model behaviour is independent of any implementation details specific to a particular programming language.

3 Process

1. Select one of the following NetLogo models (available from the **NetLogo Model Library** in the **File** menu of NetLogo):
    - Rebellion (Social Science);

    - Wealth Distribution (Social Science); or
    - Daisy World (Biology).

2. Explore the behaviour of the NetLogo model you have selected. How does it work? What behaviours can this model exhibit? Which outputs of the model can be measured? What assumptions does the model make about the system that it represents?

3. Design and implement an equivalent model in Java or Python. You should start by implementing the simplest possible prototype of the system and ensure that it works well before proceeding with more complex designs.

4. Experiment with your new model. Can you replicate the same behaviours as the original NetLogo model? Why/why not? Your experiments should investigate the effects of model parameters on model behaviour. Appropriate statistical analysis of the output of both models, the original NetLogo model and the one you implemented, is expected; e.g., reporting and comparing some output measures of the models across multiple model runs and parameter values; this includes choosing sensible approaches to measuring the behaviour of the models by measuring their outputs.

5. Formulate a question about the NetLogo model you have selected that requires an extension to the model in order to be answered (the suggestions in model documentation may provide some ideas, but I encourage you to generate your own question). Extend your model accordingly by adding a novel feature/behaviour. Design and run one or more experiments that enable you to answer the question you formulated. Finally, present and discuss the results of the experiments.

6. Write a report on your Assignment, as described below.

Please note marks will **not** be allocated for the development of new libraries or GUI interfaces – your model only needs to generate numerical output (e.g., as a CSV file).

By the proposal deadline, you should have chosen a NetLogo model, done background reading on the real world system represented by the model, explored the behaviour of the NetLogo model, started thinking about the design of your Java or Python model, and thought about the breakdown of tasks and how you will allocate these amongst the members of your group.

4 Submission

Note that only one student from each group needs to submit the proposal, final report, and code. **However, ensure that the *names and student numbers* of all group members are clearly visible on the first page of your proposal and final report.**

4.1 Proposal submission

The proposal is to be submitted via LMS by the proposal deadline (above). The proposal (named `A_B_C_proposal.pdf`, where `A`, `B`, and `C` are the last names of each group member) is expected to be 1-2 pages (11pt font, reasonable margins) and contain:

- A descriptive overview of the model you are replicating (e.g., purpose, users);
- The design of the existing model (e.g., states, update rules);

- The design of your model (e.g., classes, attributes, methods);
- The experiments that you intend to run (optionally, some results from the NetLogo model or early results from your model);
- A plan of how you intend to break down your Assignment into tasks and assign them to group members, and a timeline for completing these tasks. **NB: the contribution of each group member should be evenly spread across the tasks and duration of the Assignment.**

The proposal is worth 0 marks. However, failure to submit it by the deadline will incur a 1 mark deduction in your final mark. The proposal will also constitute a 'first draft' of your report for final assessment.

4.2 Final assignment submission

We will use the LMS for the final assignment submission. You are expected to submit the assignment in two parts:

1. A PDF copy of your report (named `A_B_C_report.pdf`, where `A`, `B`, and `C` are the last names of each group member):
   - The first page of your report must contain the names and student IDs of all group members, and **the number of words contained in the report**.
   - Your report should describe the background for the model, the design of your model and extension, the results of your experiments, and a discussion of your findings. The criteria below (Appendix A) provide an indication of the content expected in your report, and should be used to structure the sections of your report.
   - Your report should include an appendix (maximum length 1/2 page) outlining how your group worked together to achieve the project; e.g., successes and challenges confronted, any modifications to the plan outlined in your initial proposal, etc.
   - If relevant, your report should include an appendix (maximum length 1/2 page) describing how your group used artificial intelligence tools, such as large language models, during the development of the report and source code. This may include, for example, idea generation, text formatting and summarisation, code generation, or optimisation. For guidance on the appropriate use of AI in assessment tasks, please refer to: `https://academicintegrity.unimelb.edu.au/plagiarism-and-collusion/artificial-intelligence-tools-and-technologies`.
   - The report must be no longer than **8 pages** (including **all** tables and figures; 11pt font, with reasonable margins), and contain **no more than 1,500 words** of text (including figure and table captions). **NB: marks will be deducted for reports that exceed these limits.** Your reference list (bibliography) and appendices are **not** included in these page or word limits.
2. A zip file (named `A_B_C_code.zip`, where `A`, `B`, and `C` are the last names of each group member, in the same order as in the zip file name) containing:
   - All source code developed in your Assignment.
   - Any scripts required to run the experiments documented in your report.

- Clear instructions describing how to build and run your model (see note below about not requiring 3rd party dependencies).

Code will be tested in a Java SE 24 or Python 3.13 environment, and hence must be compliant with Java SE 24 or Python 3.13. **NB: marks will be deducted if it is not clear how to build and run your model, or your model does not build and run without external dependencies! For example, running your code should not require the use of any third party libraries, IDEs, or build tools. If you use an IDE to develop your code, you *must* check to ensure that it can be built and run independently of the IDE.**

Late submissions: Late submissions will attract a penalty of 1 mark for every day that they are late. Information on how to apply for an extension or special consideration can be found on the LMS page linked from this assignment submission page. Note that late or no submission of a proposal will also incur a 1 mark deduction as described previously.

5 Group contribution feedback

At the conclusion of the project, you are required to complete a short questionnaire on LMS rating your own contribution to the group's efforts and that of your fellow group members against the criteria listed below. I hope that all groups experience a positive and collaborative working relationship. However, where there is substantial disparity in contribution, this may be used as a basis for weighting marks assigned to individual group members.

Group contribution feedback criteria:

1. Motivation, time management and responsibility: attends meetings on time, accepts fair share of work, and reliably completes work on time.
2. Creativity, originality: initiates new ideas, initiates group decisions.
3. Communication skills: good listener, effective contributor to group discussions.
4. General team skills: positive attitude, supports group decisions and helps to achieve consensus.
5. Technical skills: provides technical solutions to problems.

For each criterion, you will be asked to rate your/others contributions on the following scale:

- 4: better than most of the group;
- 3: about average for the group;
- 2: less good than most of the group;
- 1: no help at all to the group; or
- 0: a hindrance to the group.

A Criteria

A.1 Report

Note that achieving *full* marks for a criterion requires that it is satisfied to an exceptional level!

Criterion	Description	Marks
Background & Model	You have clearly stated the aims and objectives of your study and provided an appropriate review of background material on your chosen model, including justifying why the system modelled is of interest and is a "complex" system. You have clearly described the design of your model, including describing the components and interactions, how the model relates to the real world, and how you have designed your Java or Python implementation.	5 marks
Replication & Extension	You have designed and executed appropriate experiments to explore and compare the behaviour of your Java or Python model and the original NetLogo model. You have described a range of scenarios used in your experiments. You have designed and implemented an appropriate extension to your Java or Python model, specified a question that this extended model allows you to address, and used your model to address this question.	5 marks
Results & Discussion	You have clearly presented the results of your investigations using clear and appropriate tables and figures. You have interpreted and discussed the results of your experiments, the outcome of the replication exercise, and the answer to the question addressed by your model extension.	5 marks
Writing	Your writing is well-expressed, clearly proof-read and demonstrates a coherent development of ideas. Your appendix outlines the successes and challenges involved in achieving your group's plan.	3 marks
Total		18 marks

A.2 Code quality

Criterion	Description	Marks
Design	The design of the model is of high quality – clear and succinct – and is potentially extensible (illustrated by the extension you choose to implement)	3 marks
Code formatting	The implementation adheres to the code format rules (Appendix B).	2 marks
Executability	The submitted code builds and runs, and generates output consistent with the results provided in the report.	2 marks
Total		7 marks

B Code format rules

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

- Every Java or Python class must contain a comment indicating its purpose.
- Every function or method must contain a comment at the beginning explaining its behaviour. In particular, any assumptions should be clearly stated.
- Constants, classes, and instance 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-statements, etc., must be indented consistently.
- Each line should contain no more than 100 characters.