Electronics and Computer Science
Faculty of Physical Sciences and Engineering
University of Southampton

Anna Kutseva

11 December,2018

Exploring Turing Machine Simulations on Social Machines

Project supervisor: Thanassis Tiropanis
Second examiner: Adam Prugel-Bennett

A project progress report submitted for the award of BSc Computer Science

Contents

Abstract

This project aims to establish a framework around which social machine models can be built and evaluated. Social machines are defined herein as those aggregating input from a number of potentially untrustworthy sources (oracles), be this noisy data or from human input. After review, it appears there are a number of theoretical models built upon standard Turing machines, but as the field is still emerging there is no tool for quickly constructing and testing the efficacy of said machines.

To address this, a plan is constructed to create a reference implementation of the reviewed social machines by building upon an existing tool for modelling Turing machines. These implementations will tuneable based on the properties of the simulated oracles at which point their performance will be compared and evaluated for extension in future work.

1 Introduction

1.1 Problem

Large and complex computer systems no longer only involve machines performing deterministic operations; there are cases where human input is an integral part in the completion of a certain task or advancing through a continuous process. With the rise of the World Wide Web people have been prompted to not only interact with each other online, but with systems they can contribute to. Such systems can be classified as social machines, as the dependency of their states on the non-deterministic nature of human participants conveys the interweaving of computer logic with real-life knowledge.

1.2 Goals

This project aims at finding a way to simulate such environments, thus evaluating the feasibility of modelling social machine components using Turing machine simulations. There are three major stages this objective has:

Stage 1 Gather a selection of models and classification frameworks to formally define the different behaviours of each component implementation

Stage 2 Find an appropriate open source Turing Machine simulation framework and extend it with implementations of these extra models

Stage 3 Assess the performance and accuracy of these simulations based on theoretical input scenario data or example data if such is available.

Though largely sequential as each stage depends on the previous, it is acknowledged that there will be some amount of overlap as concepts require clarification or alignment moving forward.

1.3 Scope

Following the literature review it would seem that most previous endeavours into the domain have remained purely theoretical and as such evaluation metrics will be relatively basic. There is an intention to contact the wider academic community on this topic, but otherwise this project will focus primarily on the implementation rather than attempting to populate an expansive data set for testing all possible metrics and scenarios moving forward.

2 Literature Review

2.1 Turing Machines

2.1.1 Concept

The initial concept of the now famous Turing machine was created as a mechanism to prove that the Entscheidungsproblem is unsolvable [1] but turned out to have many further applications. While Alan Turing's take of machine learning [2] and human-like conciousness [3], although possible to emulate with state machines, were way ahead of their time, his model of abstract a(utomatic)-machines contributed greatly on the foundations of the computer science field.

Formal definition

$M = \langle Q, \Gamma, \Sigma, b, q_0, \delta, F \rangle$ [4] where:

- Q - the finite set of states
- Γ - the complete alphabet set
- $\Sigma \subseteq \Gamma \setminus \{b\}$ - the set of tape symbols
- $b \in \Gamma$ - the blank symbol
- $q_0 \in Q$ - the start state
- $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma - \times \{L, R\}$ - the transition function
- $F \subseteq Q$ - the set of accepting states[1]

2.2 O-machines

There exists a less known idea for an abstract machine by Turing, which was introduced in his PhD thesis - the *o-machine*. Similarly to a-machines[2], o-machines can be described through tables. The main difference between o-machines and other abstract machines, such as a-machines, is the black box element, called an 'oracle'. The latter's definition in is not expanded on in Turing's paper, beyond being depicted as *"some unspecified means of solving number theoretic problems"*[6]. No details on the internal configuration of an instance of such a machine are available. The o-machine is equipped with one or more oracles, each of which can perform a primitive operation that returns the values of a non-Turing-machine-computable[3] function[7] in the form

$$p : \mathbb{N} \mapsto \mathbf{2}$$

The black box nature of the oracles allows for assumptions about what o-machines could represent. One proposed interpretation is that a human brain can be simulated by an o-machine[7]. Within the context of modelling social machines, a set of oracles can be utilised as stand-ins for a group of human individuals. Any taken series of actions in a certain

[1] There exist variations of the definitions with regards to accepting and rejecting states, but they are all proven to be equivalent. This one follows the definition taken from 'Introduction to automata theory, languages, and computation'[4]. In section 3.1.1 single *accept* and *reject* states are used instead, following the 'Introducing the Ω-machine' paper[5]

[2] The Turing machines that are familiar to the greatest number of people in the field.

[3] In other words - number theoretic

environment can be broken down to a set of decision problems, which will then be emulated by the two-valued functions of the oracles.

According to van Melkebeek[8], an o-machine has the following components:

- Shared with traditional TMs:

  work alphabet a set of symbols which can be written on the work tape

  work tape an infinite sequence of cells that can either be empty or contain a symbol from the tape alphabet

  read/write head a component located on a single tape cell at a time that can move left or right on the work tape, as well as read and write[4] symbols on it

  control mechanism responsible for performing actions such as moving the head, manipulating data and switching states

- Additional components

  oracle alphabet a set of symbols that may or may not be different from the work alphabet

  oracle tape a semi-infinite tape of cells that can either be empty or contain a symbol from the oracle alphabet

  oracle head acts the same way as the RW head, but on the oracle tape

  `ASK` **and** `RESPONSE` **states** when the oracle enters `ASK` state, the current oracle tape contents are taken as a problem instance and given as an input to the oracle, which, upon finding the problem solution, replaces the work tape contents with its output; the head is moved to the beginning of the tape; state is set to `RESPONSE`

2.3 Interaction

Machines established solely on the basis of computable functions fall flat in modelling present-day computing, considering its trademarks are *interaction* and *reactivity*[9]. Having in mind the notion that interactive systems are more expressive than algorithmic ones[10], we can look into evolving the traditional models into something more applicable to the current technological circumstances.

The TM component of the project can be modelled like a Persistent Turing Machine, with a work and output tapes[5].

2.4 Social Machines

The concept of a social machine revolves around the idea of collaborative problem solving [11]. As defined by Tim Berners-Lee, they are *"processes in which the people do the creative work and the machine does the administration"*[12].

[4] or delete by placing a 'blank symbol'

[5] Possibly an input tape as well, if determined as needed.

The combination of predefined machine transitions and probabilistic human input allows for introducing a plethora of outcomes for even a single system if it gets exposed to different selections of people. This is closer to the reality of modern computing, than the strict schemas of Turing machines. [6]

By reinforcing connection and collaboration between people, social machines indicate existing relationships among themselves. One type of association they exhibit is belonging to one another and thus having types that can be arranged in a hierarchical tree with the World Wide Web as an infrastructure being its root (See Figure 1) [13].

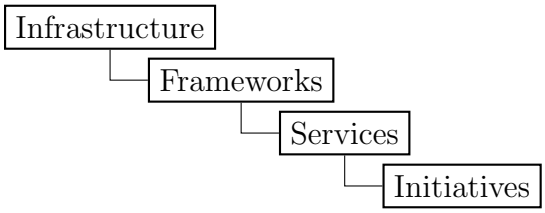


Figure 1: Granularity levels of social machines - *Initiatives* (a.k.a. *Projects*) using *Services* built upon *Frameworks* contained into the Infrastructure

The emerging web of social machines Studies the relationships between SMs[14]

From the semantic web to social machines: a research challenge for AI on the World Wide Web Connects the semantic web with SMs and provides future goals[15]

2.4.1 Social Machine and Computation Examples

Given a lack of evaluation data found during the literature review, a few real-world social machines have been identified with the possibility of simulation by reducing the domain and action set of the oracles and machine respectively to a few representative outcomes (both positive and negative):

Wikipedia Serving as one of the most well-known cases of social machines, Wikipedia would not exist without the endless flow of human input. It confirms the capacity of social computation by being the largest encyclopedia in existence, that is being updated at such a rate that physically publishing its content is absolutely redundant. Unfortunately, it is speculated that Wikipedia has recently been in decline[16]. A correct simulation of the lifetime of a similar (and preferably simplified) online collaborative knowledge project could predict tendencies in increases and declines of activity

Games with a Purpose It is beyond doubt that gamification is a powerful motivation method - simply adding a score number next to a person's username sparks the desire to increase said number. As such, it can be used to encourage meaningful contributions to a real-world broad scope problems through an entertainment platform [17]. However, this genre of games is far from being as popular as other genres. The few notable examples that

[6] While non-deterministic counterparts of the latter exist, they are closer in potential to being abstract machines than actual representations of the ever-increasing and practically boundless total of branches each interaction presents in real time.

have been used to justify the importance and efficacy of GWAPs[18] are as of today defunct. Carrying out simulations of such games during their development may demonstrate potential issues, whose resolution may lead to the games' longer life and higher recognition.

The DARPA Network Challenge The DARPA Network Challenge The Defense Advanced Research Projects Agency organises annual prized challenges to stimulate problem solving through group effort in search of revolutionary research approaches. In 2009 DARPA presented the efficiency of social computation and crowdsourcing with the following challenge: ten weather balloons were scattered across the United States and had to be located in the shortest time possible. The participating teams used various forms of social networking to achieve their goal, and the winning team managed to complete the task in just 9 hours by putting a referral system and cash incentives to use. Many of the strategies were essentially incorporating social computation to tackle the problem in a much faster way by engaging people, rather than by setting up a massive fully automated detection system[19]. As several of the winning strategies are publicly shared[20], they could potentially be simulated and compared given differing oracle behaviours as part of this project.

3 Work Completed

3.1 Desing and Implementation

3.1.1 The Omega Machine

The proposed implementation of a social machine model is the Ω-machine, that serves as an extended model of the Turing machine, employing o-machines whose oracles act as human individuals[5]. The initial configuration based on the omega-machine idea is:

$$\langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, \Gamma_\omega, \delta_\omega \rangle$$

where Γ_ω is the oracle alphabet and δ_ω is the transformation function for the oracles. Upon creation, each oracle belonging to a certain omega-machine instance will only need to have its `ASK` and `RESPONSE` states be specified. In this case:

$$\langle M, s_{ASK}, s_{RESPONSE} \rangle$$

where M is the omega-machine that the oracle belongs to and from which it *inherits* the oracle alphabet and the transformation function. The first version of the social machine model will be built following this structure.

If time allows, the the δ_ω component may be broadened into Δ_ω, which would support a different function for each oracle instance, looking like:

$$\langle M, \delta_\omega, s_{ASK}, s_{RESPONSE} \rangle$$

where $\delta_\omega \in \Delta_\omega$, meaning there is a variety of functions that could be assigned to the oracle from a predefined whitelist, and the a machine would be modified into:

$$\langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, \Gamma_\omega, \Delta_\omega \rangle$$

In this way the oracle cluster part of the model can come closer to resembling a crowd of diverse individuals.

Ideally, the ultimate configuration goal would be to have o-machines be as independent from the core a-machine as possible:

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, \rangle$$

$$O = \langle M, \Gamma_\omega, \delta_\omega, s_{ASK}, s_{RESPONSE} \rangle$$

where the o-machine has a reference to the a-machine, but the a-machine does not have a reference to any of the o-machines, mimicking the behaviour of a system, which is logically independent from the specific properties of its users. Conclusively, a social machine, or an omega-machine will then be defined as

$$\Omega = \langle M, \mathbb{O} \rangle$$

where M represents the technical platform, or the *machine* part, and $\mathbb{O}$ represents the set of oracles, or the *social* part.

3.2 Open Source Analysis

3.2.1 Criteria

Deciding on a framework to extend is a virtually irreversible(See Table 3 in Appendix A) and as such should be considered strongly before making a decision. The web contains a myriad of open source software options, most of which can be found on websites such as `sourceforge.net`. Notwithstanding, many, if not most, of those projects lack support, reviews, and proper documentation, which marks them as unsound options. To support the final choice, each option was evaluated against the following list of criteria:

Acclaim A well recognised, favourably reviewed piece of software promises a good groundwork for the experiment and increases the likelihood of it being built upon in future work.
Questions to ask: *Is there any academic or industrial usage? Is the overall feedback thorough and positive?*

Language Implementation in a modern and widely utilised language ensures quick uptake.
Questions to ask: *What language is it written in? Is the language an adequate choice for the task? Am I familiar with this language?*

Maintenance Proper and frequent support promises fewer performance issues and bugs. Also, large involvement with the project indicates large amount of effort, and consequently - good quality of the product.
Questions to ask: *What are the activity levels on any associated forums or code repositories? How often have past updates been rolled out? When was the most recent update?*

Functionality A large code-base reduces the need to implement basic or common features. Questions to ask: *Is the program expressive enough to be extended efficiently? Are there components I can reuse in multiple ways?*

Documentation Well commented code will be easier to work with; any additional instructive materials, such as help pages, example tutorials, readme files, or full manual documents provide evidence of usability and accessibility; thorough change logs give more insight on development, and further inform about handled or potential issues. Questions to ask: *Is it well commented? Is there a dedicated website, user manual, or a readme file? Are there any examples or tutorials? Is there a comprehensive change log for recent updates?*

3.2.2 Options

Below is a selection of eight contenders that were considered.

JFLAP Educational software by the Duke University, written in Java. It has been around since 1996 but it is still being updated - the latest release is from 2018 and running on Java 8. The program creator Susan Rodger has been recognised in the academic field as an ACM Distinguished Member, and awarded the ACM Karl V. Karlstrom Outstanding Educator Award. Beside Turing machines, JFLAP enables simulations for numerous other topics on automata and formal languages. The user interface is done using Java Swing.

Tursi Created in Java by the German developer Claus Schtzle. It can be used both through a GUI or in a console. It works with specifically designed .tm files for the system written in a Turing machine based language. Its distinctive features include exporting tables to state diagrams, history and break states.

Owen's Turing Machine Simulator A Turing Machine simulator, developed in Java by Owen Kellett. It uses files with a .tmo extension. Its user interface is also built with Java Swing.

Tuatara Written in Java Swing, last updated in 2007. It has a simple point-and-click interface. Out of the selection found in SourceForge it was the only project with recent downloads, as well as a review.

TMSimulator The only application to use JavaFX instead of Swing. A student project from the technical university of Vienna. Last updated in 2011.

JSTuring Web-based simulator in JavaScript. It uses .txt files for processing machines. The last commit was made in June 2018.

Tm HTML based with sample machines that are hard-coded.

TuringSym Written in Delphi. Last modified in 2004.

3.2.3 Choice

The majority of open source projects available are implemented in Java and have user interfaces built with the Swing toolkit. After inspecting the source code contents and considering factors such as support, functionality, public feedback and quality of documentation,

I strongly decided in favour of JFLAP [21], as it excelled in all the aforementioned requirements in comparison to the rest of the projects. In fact, its diverse functionality and multitude of classes for any sub-entity that might be needed will enable easier and cleaner implementation of the social machine model.

4 Plan of Remaining Work

4.1 Timeline

With the bulk of the literature review complete below is outlined the timeline for the remaining work as part of this project (see Figure 2 in Appendix A). Having decided on JFLAP as the framework of choice, the majority of the upcoming weeks will be spent on design and implementation of the various models identified in the literature review, based on the structure and available components within JFLAP. As this is the centrepiece product of the project, unfortunately this cannot be broken down to run in parallel with any other tasks except for testing once implementation nears completion. After implementation, the various models will be compared and evaluated based on metrics that are yet to be identified at this point and conclusions can be drawn as the tests are run.

Alongside the consideration for concurrently executable tasks, extra time has also been set aside for known and unknown interruptions. Most notably are the blocks of time dedicated to the more intense study periods surrounding exams and the weeks during the holidays where inability to progress due to family engagements are highly likely. Particularly the Christmas period has been set aside for recovery before the next term begins. Although the Easter break has also been marked as a holiday for planning purposes, its proximity to the final deadline suggests that a certain amount of work will need to be done in this period as well. Finally all expected task durations have been overestimated by roughly 20% to accommodate for unforeseen events and general delays. With this in mind, ideal progress should be ahead of the Gantt chart estimation and falling behind would be an indication of some underlying time management or complexity issue.

4.2 Risk Analysis

To complement the timeline breakdown of upcoming work, a risk analysis has been performed to anticipate potential issues that would have a significant impact on the project's progress. Each risk was assigned a Likelihood (L) and Severity (S) which multiply together to produce the overall Risk score colour coded by concern (see Tables 1, 2 and 3 in Appendix A). Once classified, mitigation strategies and contingency plans were assigned in an attempt to address the concerns ahead of time and help factor in any recovery period into the timeline assessment. Concluding from this analysis, issues arising during implementation accompanied the highest risk factor and as such have been granted the largest allocation of time in the Gantt chart.

A Appendix

Table 1: Management Issues

Problem	L	S	Risk	Mitigation	Contingency
Falling behind schedule due to clashes with other assessed tasks	*3*	*3*	**9**	Extra time allocated for the period surrounding exams. One hour of work allocated per day as a minimum in parallel with other modules.	Identify problem modules and assessments as early as possible to raise with supervisor ASAP.
Falling behind schedule due to slow progress	*2*	*3*	**6**	Tasks defined requiring more time than estimated by default to act as an on-going buffer. Burn-down chart to be kept up-to-date for sub-goals across entire year.	Actual work should ideally run ahead of the Gantt chart given the extra time included. If falling behind then remaining work and Gantt chart should be re-evaluated with supervisor.
Unavailability of supervisor	*1*	*3*	**3**	Maintain attendance of weekly supervisor meetings. Prepare multiple tasks to carry out ahead of time in case a meeting has to be skipped.	Continue with tasks as planned, maintain contact with supervisor by any means available. Contact university for support in extreme case of extended absence.
Licensing issues	*1*	*2*	**2**	Utilise open-source software, ideally that which has already been attributed by other academic use-cases. Contact authors for permission.	Comply with author's requests. Publish products of project as permissible under appropriate license with referencing as necessary.
Illness	*2*	*2*	**4**	Extra time buffer can be used for sick days as necessary.	Re-evaluate remaining tasks as appropriate.Contact supervisor and university to advise in extreme cases.

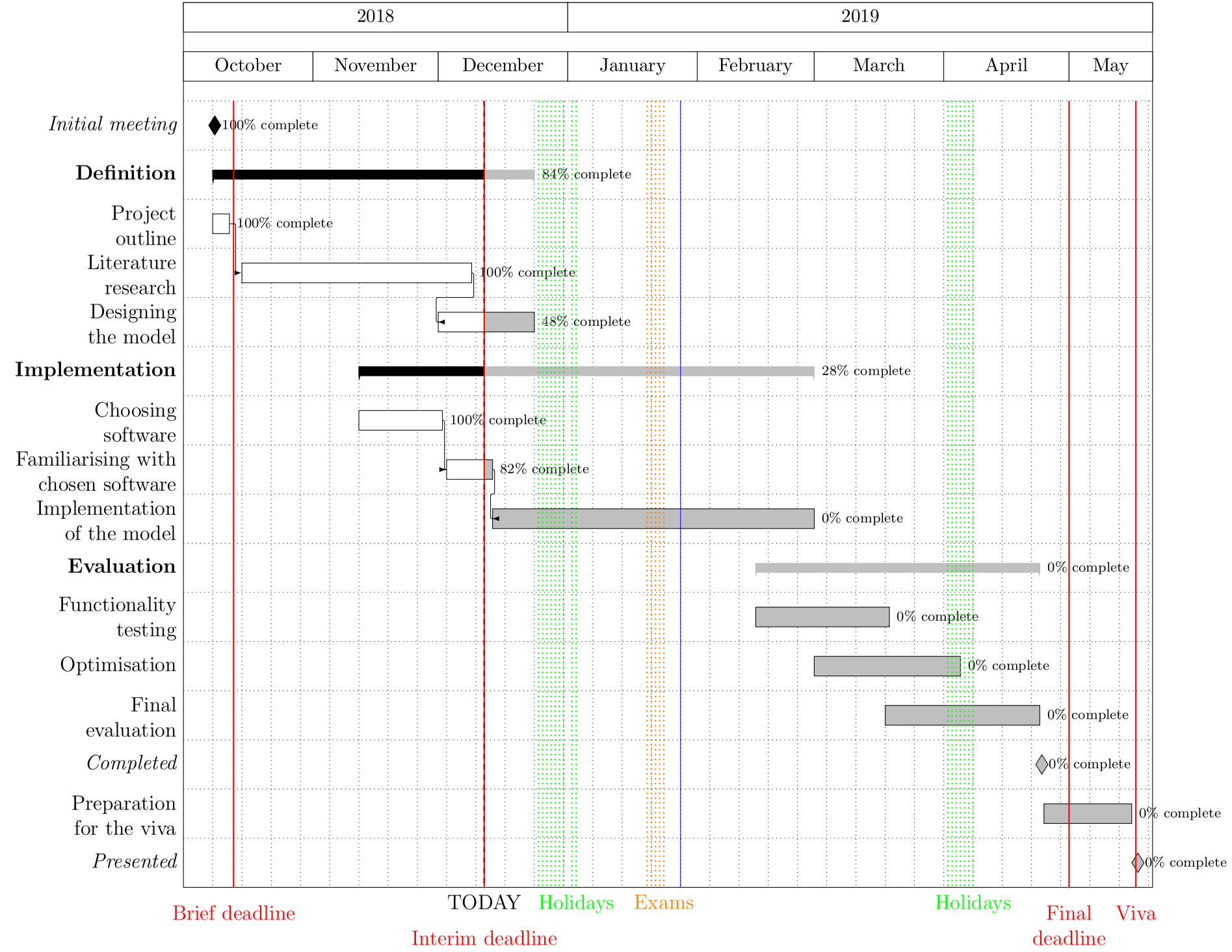
Table 2: Technical Issues

Problem	L	S	R	Mitigation	Contingency
Hardware not powerful enough	*2*	*2*	**4**	Design model components with performance in mind. Utilising university hardware as necessary.	Contact supervisor to request access to specialist hardware.
Damaged hardware	*1*	*1*	**1**	Ensure program and development environment are accessible from multiple sources (not just personal machine).	Utilise university hardware.
Loss of work (code or report)	*1*	*3*	**3**	Frequent back-ups to local drives and commits to remote repositories.	Retrieve latest copy from backup.

Table 3: Implementation Issues

Problem	L	S	R	Mitigation	Contingency
Chosen framework is not appropriate for implementing the model	*2*	*4*	**8**	Appropriate analysis made before framework selection.	Implement closest approxiamation using next most appropriate tooling. Document findings regardless.
Difficulty grasping important concepts	*2*	*2*	**4**	Extensive literature review before beginning implementation. Regular contact with supervisor and academic community.	Schedule additional meetings with supervisor focussed on specific topics as soon as possible to avoid delays.
Insufficient methods of evaluation	*4*	*3*	**12**	Contact academic community and review any evaluation performed in available literature.	Discuss potential metrics with supervisor and evaluate to best of ability.
High algorithmic complexity	*2*	*3*	**6**	Review and optimise code as implemented to minimise overhead on top of underlying algorithmic complexity.	Request access to specialist hardware to accommodate load.
Poorly designed model	*2*	*3*	**6**	Review code with supervisor during implementation, test as soon as is feasible.	Begin testing and evaluation phases as soon as possible to avoid issues with write-up due to missing data.
Bugs	*4*	*2*	**8**	Utilise functionality tests to ensure correctness as soon as possible.	Prioritise evaluation of correct work. Address issues as timing permits.

Figure 2: Project Gantt chart.



References

[1] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. s2-42, no. 1, pp. 230–265, 1936.

[2] A. M. Turing, "Intelligent machinery," in *Machine Intelligence* (B. Meltzer and D. Michie, eds.), vol. 5, ch. 1, pp. 3–23, Edinburgh, UK: Edinburgh University Press, 1969.

[3] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, pp. 433—-460, 1950.

[4] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to automata theory, languages, and computation*, vol. 32. ACM, 2001.

[5] L. Zhang, T. Tiropanis, W. Hall, and S.-H. Myaeng, "Introducing the omega-machine," *Proceedings of the 23rd International Conference on World Wide Web - WWW '14 Companion*, pp. 905–908, 2014.

[6] A. M. Turing, "Systems of Logic Based on Ordinals," *Proceedings of the London Mathematical Society*, no. 1938, 1938.

[7] B. J. Copeland, "Turings o-machines, searle, penrose and the brain," *Analysis*, vol. 58, no. 2, pp. 128–138, 1998.

[8] v. M. Dieter, "Randomness and Completeness in Computational Complexity," *Lecture Notes in Computer Science 1950*, 2000.

[9] D. Q. Goldin, S. A. Smolka, P. C. Attie, and E. L. Sonderegger, "Turing machines, transition systems, and interaction," *Information and Computation*, vol. 194, no. 2, pp. 101–128, 2004.

[10] P. Wegner, "Why interaction is more powerful than algorithms," *Commun. ACM*, vol. 40, pp. 80–91, May 1997.

[11] M. Luczak-Rösch, R. Tinati, K. O'Hara, and N. Shadbolt, "Socio-technical computation," *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing (CSCW'15 Companion)*, pp. 139–142, 2015.

[12] T. Berners-Lee and M. Fischetti, *Weaving the Web: The original design and ultimate destiny of the World Wide Web by its inventor*. DIANE Publishing Company, 2001.

[13] N. R. Shadbolt, D. A. Smith, E. Simperl, M. Van Kleek, Y. Yang, and W. Hall, "Towards a Classification Framework of Social Machines," *SOCM2013: The Theory and Practice of Social Machines*, vol. 19, no. 2, pp. 387–401, 2009.

[14] S. R. L. Meira, V. A. Burégio, L. M. Nascimento, E. G. M. de Figueiredo, M. Neto, B. P. Encarnação, and V. C. Garcia, "The emerging web of social machines," *CoRR*, vol. abs/1010.3045, 2010.

[15] J. Hendler and T. Berners-Lee, "From the Semantic Web to social machines: A research challenge for AI on the World Wide Web," *Artificial Intelligence*, vol. 174, no. 2, pp. 156–161, 2009.

[16] T. Simonite, "The decline of wikipedia: Even as more people than ever rely on it, fewer people create it," *MIT Technol Rev*, 2013.

[17] L. V. Ahn, "Games with a Purpose," *Computer*, vol. 39, pp. 92–94, 2006.

[18] L. Von Ahn and L. Dabbish, "Designing games with a purpose," *Communications of the ACM*, vol. 51, no. 8, pp. 58–67, 2008.

[19] D. Robertson and F. Giunchiglia, "Programming the social computer.," *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, vol. 371, no. 1987, p. 20120379, 2013.

[20] J. C. Tang, M. Cebrian, N. A. Giacobe, H.-W. Kim, T. Kim, and D. B. Wickert, "Reflecting on the darpa red balloon challenge," *Communications of the ACM*, vol. 54, no. 4, pp. 78–85, 2011.

[21] S. H. Rodger and T. W. Finley, *JFLAP: an interactive formal languages and automata package*. Jones & Bartlett Learning, 2006.