Community Modelling and Communication with PhiloLab

Peer-Olaf Siebers
School of Computer Science, University of Nottingham, UK
Email: peer-olaf.siebers@nottingham.ac.uk

What is it all about? What are successful model development strategies for complex social systems modeling? How does modelling work in large, collaborative, and multi-disciplinary projects in academia, non-governmental and governmental organisations, and industry? How do we derive new and general insights from modelling complex social systems? Anecdotal evidence suggests that the community of agent-based modellers partially suffers from a lack of structured and standardised ways for model development. In order to close this gap, we have created a model development framework, namely the Engineering Agent Based Social Simulation framework (or EABSS for short) which supports model development and model documentation in a structured way. Figure 1 shows a high level overview of the framework. Full details together with an illustrative example can be found in Siebers and Klügl (2017) and some guidelines can be found in Appendix A.

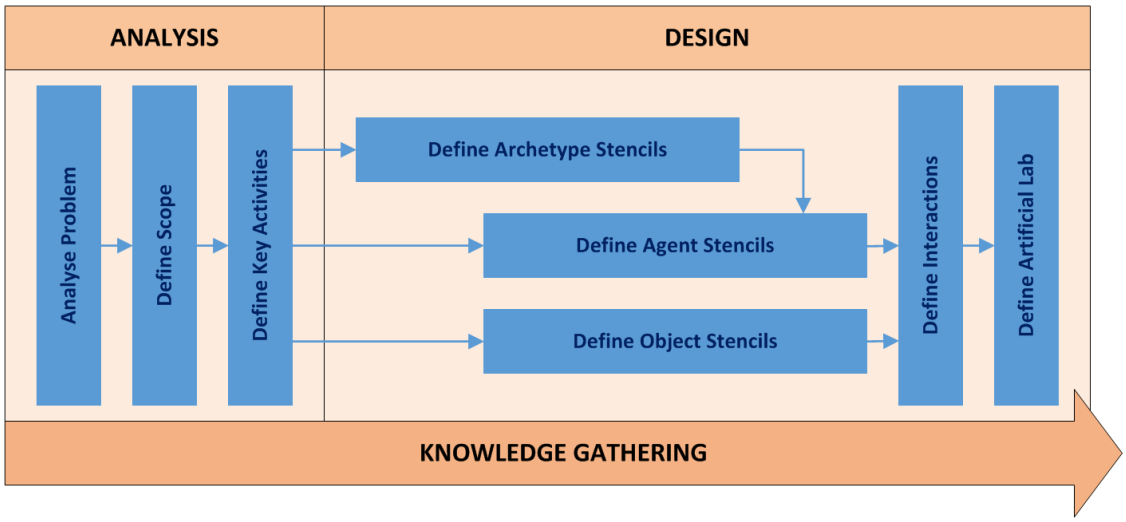

Figure 1: High level overview of the EABSS

PhiloLab is a concept based on the EABSS and its purpose is to stimulate and formally support discussions about philosophical questions of future societal models. Here, the EABSS becomes a tool for stimulating a focused conversation, in order to capture as many thoughts about a specific topic as possible. The goal in this case is to use the guided discussions to capture the diversity of perspectives held by various stakeholders related to the topic being discussed.

How does it work? The EABSS is grounded in the concept of co-creation (Mitleton-Kelly 2003) and ideas from Software Engineering (Sommerville 2015). In addition it draws on elements of Kankainen's focus group approach to service design (Kankainen et al 2012). The framework implicitly provides ground rules, which is something commonly done when working with children but often forgotten when working with grownups. These grounded rules are in line with De Bono's philosophy of parallel thinking, and state that people are going to listen to each other and that people respect each other's point of view. To capture information it uses predefined table templates, and UML (a graphical notation used in Software Engineering) as main forms of stimulating and documenting contributions from all participating stakeholders during problem analysis and model design. It is this combination of tools and methods that makes it approachable for everyone, who wants to give it a go.

When do we use it? PhiloLab can be used for two purposes: (1) for collaborative model development (from scratch or to extend/validate existing models) and (2) to stimulate and formally support discussions about philosophical questions of societal issues that need to be addressed. We have tested the framework in several domains, including Architecture, Geography, Organisational Behavior (Secchi et al 2018), and Mental Health (Nielsson et al under review). It is designed with the aim to look at a complex system in more detail with every further step. There is always information from previous steps that can be used to get started with the next step. This principle serves validation, as getting stuck in the current step is a good indicator that something in previous steps is not quite right and needs to be amended.

What do we get out of it? The outcome of a PhiloLab session is a structured record of the key points of the focus group discussions, in a format that is easy to understand by all stakeholders, and easy to extend. With a little effort this can often be translated into an agent-based social simulation model, which can then be used by the stakeholders as a "what-if" analysis tool.

What is our experience so far? Interestingly we found that each of the academics involved in running the focus groups finds PhiloLab supportive in a different way, perhaps embedded within the research method used in their domain. So the concept itself can be seen as interdisciplinary, while it was originally only intended to support social simulation model development. It has now been used for model development, reverse engineering of existing models for validity checking, discussions to extend existing models and confirm their validity, debates to analyse research topics and work on defining new directions for research.

Where are we going from here? More recently I got interested in the philosophical debates of Richard David Precht, a German Philosopher who debates about "the digital revolution of society". I would like to use the EABSS to test some of his future visions and to see if we can visualise his worlds with the help of ABSS. I would like to approach this in a more systematic way - i.e. to build a kind of toolbox that allows us to build this new genre of models easier. In the end we could have a collection of templates that allow the community to put together models for tackling philosophical questions - perhaps in form of a toolbox as an extension to existing simulation packages.

References

- De Bono E (1985) Six Thinking Hats: An Essential Approach to Business Management from the Creator of Lateral Thinking. Little, Brown and Co (1985).
- Kankainen A, Vaajakallio K, Kantola V, and Mattelmäki T (2012) Storytelling Group–a co-design method for service design. Behaviour & Information Technology 31, 3 (2012), pp. 221–230.
- Mitleton-Kelly E (2003) Complexity Research - Approaches and Methods: The LSE Complexity Group Integrated Methodology. A Keskinen, M Aaltonen, E Mitleton-Kelly (eds.) Organisational Complexity. Tutu Publications. Finland Futures Research Centre, Turku School of Economics and Business Administration, Turku, Finland, pp. 56-77.
- Nilsson T, et al (under review) Applying Software Engineering Principles to Unpack Multi-Stakeholder Perspectives in Emerging Mental Healthcare Technologies. Submitted to the 2019 ACM CHI Conference on Human Factors in Computing Systems
- Secchi D, Siebers PO, and Herath DB (2018) Organisational Plasticity: A Community Modelling Experience. In: Proceedings of the Social Simulation Conference 2018 (SSC 2018), 20-24 Aug, Stockholm, Sweden.
- Siebers PO and Klügl F (2017). What Software Engineering has to offer to Agent-Based Social Simulation. In: Edmonds B and Meyer R (Eds.) Simulating Social Complexity: A Handbook - 2e. Springer.
- Sommerville I (2015) Software Engineering (Global Edition), 10e, Pearson.

Appendix A: EABSS Guidance

Small print orange remarks are meant to guide the focus group moderator regarding the re-use of information; purple remarks list the tools to be used in that particular step

Analyse Problem {also clarify terminology and come up with a common pool of term definitions}

- Clarify the "Purpose" of the model
- Define a list of "Hypotheses" to be tested
- Define a list of "Experimental Factors" to allow creating scenarios relevant to testing those hypotheses {look at objectives/hypotheses to work these out}
- Define a list of "Responses" to accept/reject hypotheses {look at objectives/hypotheses to work these out}

Define Scope {look for nouns in previous text to find elements}

- Define the level of abstraction
- List entities (key actors - represented by the role they play, and key objects) and concepts {key actors can also represent social/economic units, as for example families or firms}
- Indicate if these should be included/excluded in the model and justify your decision
- Use pre-defined table (headers: Category; Sub-Category; Element; Decision; Justification {while categories are provided, sub-categories are flexible and depend on the context}; categories: Actor; Physical Environment; Social and Psychological Aspects; Other)

Define Key Activities {actors come from scope table; use cases come from hypotheses and by creating user stories}

- Assign key actors to relevant activities (use cases)
- Use UML use case diagram

Define Archetype Stencils {these allow to define behaviour of actors}

- Come up with categorisation schemata for relevant key actors (agents) that will allow to separate a simulated population into behaviourally different groups
- Use habit template(s) and/or demographics and/or utility function(s)

Define Entity (Agent/Object) Stencils {attributes can be derived from archetype criteria, theory parameters, methods can be derived from the states in the related state charts} {states can often be derived from use cases}

- Create templates by defining key states an entity can be in, how these are linked, and what triggers transitions (note that this might not be required for all entities)
- List variables that ought to be tracked at the micro/meso level in order to gain insight about the issues identified during the problem analysis
- Use UML state machine diagram(s); transition table(s); class definition(s)/diagram(s)

Define Interactions {all elements defined in the agent/object stencil step need to be listed on the horizontal axis} {use cases could be listed on the vertical axis}

- Define sequences of interactions that can take place between agents and between agents and objects in specific use case realisations
- Use UML sequence diagram(s)

Define Artificial Lab {attributes provide storage for all agents/objects and initialisation parameters required for experimental factors; methods related to responses}

- List entities that need to be created; listing variables that ought to be tracked at the macro level in order to gain insight about the issues identified during the problem analysis
- Define order of execution (if relevant)
- Use UML class definition(s)/diagram(s) and sequence diagram(s)