Agent-based Building Blocks Architecture (ABBA): How software engineering can help computational social science models

Liz Verbeek[1], Tatiana Filatova[1], Amineh Ghorbani[1], Martijn Warnier[1], Igor Nikolic[1]
[1]Faculty of Technology, Policy and Management, Delft University of Technology, 2628BX Delft, The Netherlands

Keywords: agent-based model, socio-environmental systems, reusability, human decisions

1. Introduction

Agent-based models (ABMs) have become increasingly popular across various applications. Over the past decades, the number of ABM publications has grown explosively in engineering, environmental and social simulation literature (Fig 1). As a result, the need for improvements in the modeling process is growing. Specifically, more efficiency and transparency in code development [1], improvements in algorithmic representation of human behavior [2] and institutions and their alignment with social science theories are desired [3].

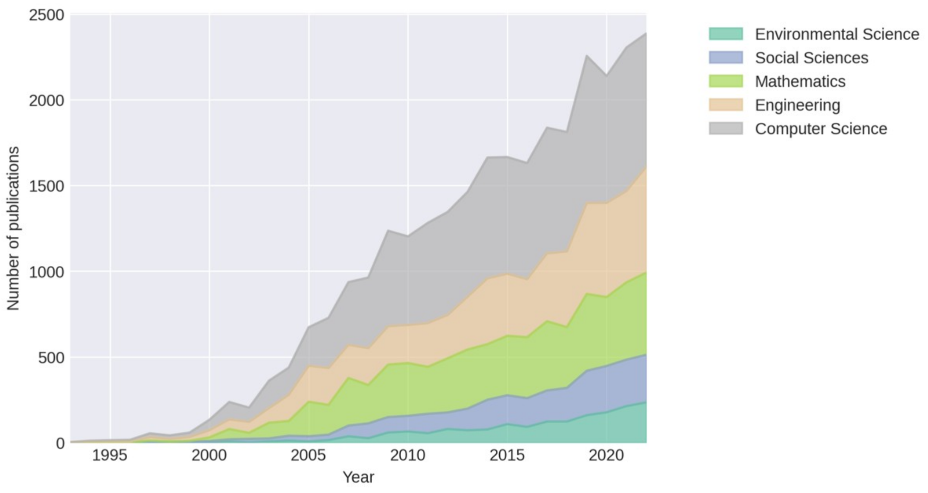

Fig 1. Number of peer-review articles and conference papers publishing about agent-based modeling between 1993-2022 across the five most common subject areas. Publications were retrieved using Scopus search query: "(TITLE-ABS-KEY('agent-based') AND (TITLE-ABS-KEY('simulation') OR TITLE-ABS-KEY('model'))".

In addition, ABM is increasingly picked up as a method by mathematics and engineering-related disciplines who may have limited experience with modeling human decisions and social processes. Yet, many of the agents we model in ABMs are humans, organizations, and social institutions. We are typically concerned with how to model their decisions, interactions, learning, adaptation or exchange of information, of experiences or of goods/services. Although much is argued on which social science theories are best suited for ABMs, there is little guidance on implementation in the model code. Standardization and documentation of ABM components representing social science concepts, both in terms of theories and processes as well as data required to capture in software code how these processes take place in reality, could thus be of importance not only to the field itself, but also to a larger community.

For over a decade, the ABM community has been contemplating the idea of reusing building blocks (RBBs) of code that are repeatedly employed across applications [4]. A modular approach with the core code standardized and tested is less error-prone, allowing ABM developers to focus more on modeling and ontologies of context-dependent human decisions, and less on coding. Furthermore, while theories conceptualizing human behavior and social institutions are vast, most of them operate with vague theoretical constructs. Their operationalization in code often remains a black box, depending on intuitive judgements of modelers. Instead of reproducing code for recurring decisions and theories, the community will benefit from learning based on experience and reusing code fragments that are tested across contexts.

2. Lessons from related fields

The standard documentation of module's features in time, space and structure, clear ways to communicate underlying theoretical assumptions and data flows across such modules have been explored for process-based models in the environmental model integration community. ABMs, however, are different in including more direct empirical entities in combination with theoretical constructs. This requires new and alternative ways of modules' standardization to make them reproducible and reusable.

From the field of software engineering, we learn that its main goal is to produce reliable, high-quality software that is easy to maintain [5, 6]. This aim is pursued in numerous ways, ranging from verification techniques such as code testing [7] and static analysis [8] to design methods such as model driven architecture and design patterns [9]. Design patterns are a well-known method to assure code quality and reusability at the software design phase. They specify how common problems in software development should be addressed, helping developers to use well-known solutions for various standard situations. Typically, design patterns are abstract descriptions of known best practices that can be programmed in many ways, providing generic methods for developing high-quality software. However, agent-based development principles that support interoperability and reusability of high-quality software enabled by design pattern principles are still scarce.

To address this gap, we propose an approach that combines design pattern principles with model-driven software development (MDSD). MDSD offers a level of abstraction that provides an overview of the inner structure of the software and reduces the details required for development to the essence [10]. To develop executable software with this approach, a model is built using a high-level language called a meta-model. A software platform then takes some transformation rules to generate an executable software from the model. Following this approach can provide several benefits for social simulation, among which are 1) speeding up the development process by giving more structure to the software, 2) managing 'complexity through abstraction', since modeling languages enable 'programming' at a more abstract level, 3) managing complexity through structure because modeling languages provide predefined placeholders for software components, 4) separating tasks in the simulation development process by splitting up the modeling and coding processes, 5) facilitating reuse and regeneration of simulation or its components because user knowledge becomes widely available in software format and 6) improving software quality, performance, maintainability and portability because the architecture of software may recur uniformly in the implementation [11].
A basic model driven approach is illustrated in Figure 2. A *computational independent model* (CIM) describes the context of the model at a conceptual level. A *platform specific model* (PSM) describes the realization of the conceptual model for a specific software platform. The transformation between a CIM and a PSM requires mapping modules that describe what each concept in the CIM is translated to in a PSM. One option for such a mapping would be to introduce a *platform independent model* (PIM) to facilitate a model-to-code translation. A PIM focuses on the operation of a software system while hiding details necessary for a particular platform [11]. As illustrated in Figure 2, a PIM acts as an

interface between a CIM and a PSM so that one CIM can be translated to many PSMs by defining only one mapping module between a CIM and a PIM. Defining a PIM meta-model can therefore make the translation of an RBB meta-model platform independent so that RBBs can be translated to any of the already existing ABM platforms (e.g. NetLogo, Repast, MASON, etc.).

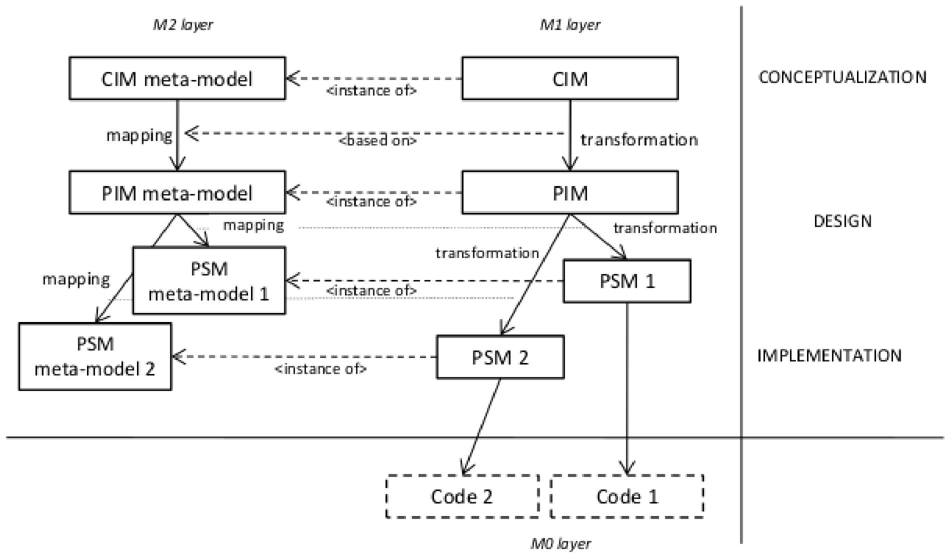

Fig 2. Platform independent modeling in MDSD, from Ghorbani (2013) [11].

3. Building block architecture

In Figure 3, we present an agent-based building block architecture following the MDSD approach. We suggest for any such building block to contain: 1) Background & Purpose – describing a recurring design problem and its application in ABMs, 2) Key concepts, definitions and use of data – defining the standard vocabulary that applies to this design problem, theory or phenomenon, and, optionally, a description of how these concepts can be quantified using empirical data, 3) a detailed specification, including standardized input and output, a (standardized) classification of the building block (e.g. if it concerns a single agent, multiple agents, or agents interacting with the environment), 4) example implementations of this building block in (tested, verified and validated) existing ABMs. Elements 1-3 make up a CIM of the building block, connecting the implementation to existing (social science) theories and concepts, whereas the last element (4) form the PIM and PSM of the building block. As theoretical concepts can be implemented in models in several ways, every building block can consist of several PIMs, and in turn, as illustrated above, every PIM can map the CIM to multiple PSMs – here: implementations in different programming languages. In our proposed architecture, the PIM(s) of a building block consist of 1) a flow chart that visually describes the building block process(es), 2) the input parameters and output forming the necessary connections across modules, 3) a sequence diagram describing the position of this module in a full ABM and 4) algorithmic design in the form of pseudocode. In turn, this PIM can be mapped to PSMs multiple programming languages. Any PSM should be provided as a working, tested code example from an existing agent-based model, with references to the existing ABM.

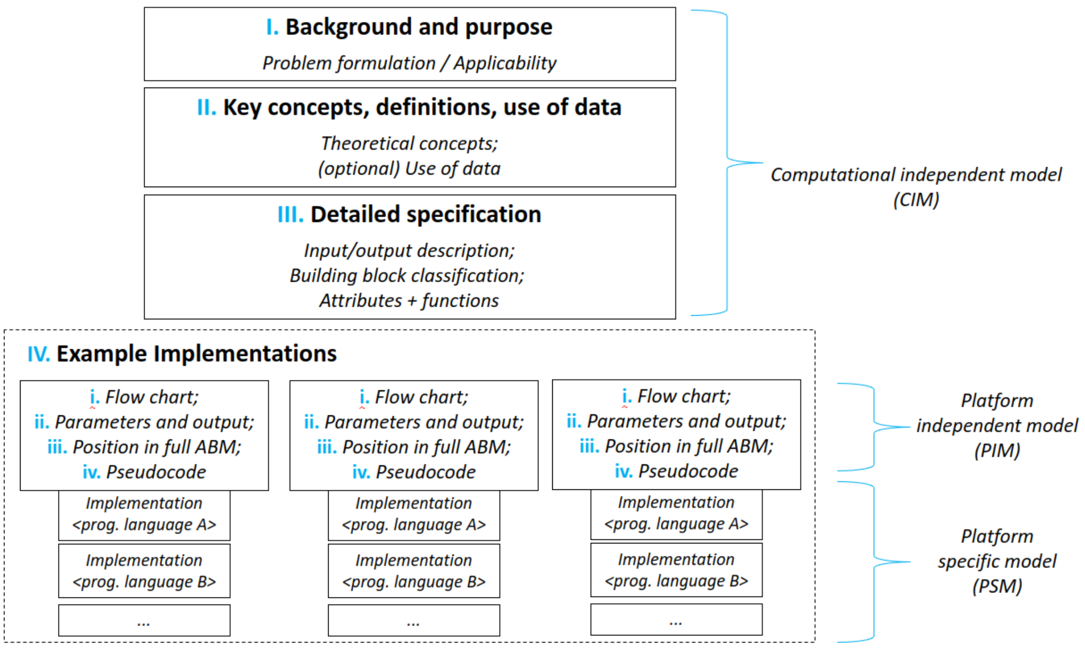
I. Background and purpose
Problem formulation / Applicability
II. Key concepts, definitions, use of data
Theoretical concepts; (optional) Use of data
III. Detailed specification
Input/output description; Building block classification; Attributes + functions
Computational independent model (CIM)
IV. Example Implementations
i. Flow chart; ii. Parameters and output; iii. Position in full ABM; iv. Pseudocode
Implementation <prog. language A>
Implementation <prog. language B>
...
Platform independent model (PIM)
Platform specific model (PSM)

Fig 3. Schematic overview of a building block following the MDSD approach.

In our presentation at the conference, we will discuss the lessons learned in the environmental model-integration community advocating modularity, define a minimal set of principles grounded in software engineering that the agent-based building block architecture could leverage from, substantiate this building block architecture with several examples and present an online platform for sharing reusable building blocks. The online platform is under construction at this moment, with several RBBs already tested using this template. We expect the launch of the beta version of the online platform to be in early September and hope to use this opportunity to discuss the first generation of reusable ABM building blocks and test the platform with the conference participants.

References

[1] V. Grimm *et al.*, "The ODD Protocol for Describing Agent-Based and Other Simulation Models: A Second Update to Improve Clarity, Replication, and Structural Realism," *J. Artif. Soc. Soc. Simul.*, vol. 23, no. 2, p. 7, 2020.

[2] M. Schlüter *et al.*, "A framework for mapping and comparing behavioural theories in models of social-ecological systems," *Ecol. Econ.*, vol. 131, pp. 21–35, Jan. 2017, doi: 10.1016/j.ecolecon.2016.08.008.

[3] H. Muelder and T. Filatova, "One Theory - Many Formalizations: Testing Different Code Implementations of the Theory of Planned Behaviour in Energy Agent-Based Models," *J. Artif. Soc. Soc. Simul.*, vol. 21, no. 4, p. 5, 2018.

[4] A. R. Bell, D. T. Robinson, A. Malik, and S. Dewal, "Modular ABM development for improved dissemination and training," *Environ. Model. Softw.*, vol. 73, pp. 189–200, Nov. 2015, doi: 10.1016/j.envsoft.2015.07.016.

[5] B. W. Boehm, "Software Engineering Economics," *IEEE Trans. Softw. Eng.*, vol. SE-10, no. 1, pp. 4–21, Jan. 1984, doi: 10.1109/TSE.1984.5010193.

[6] F. Tsui, O. Karam, and B. Bernal, *Essentials of Software Engineering*. Jones & Bartlett Learning, 2022.

[7] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software Testing Techniques: A Literature Review," in *2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)*, Nov. 2016, pp. 177–182. doi: 10.1109/ICT4M.2016.045.

[8] P. Emanuelsson and U. Nilsson, "A Comparative Study of Industrial Static Analysis Tools," *Electron. Notes Theor. Comput. Sci.*, vol. 217, pp. 5–21, Jul. 2008, doi: 10.1016/j.entcs.2008.06.039.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns: Abstraction and reuse of object-oriented design". ECOOP'93—Object-Oriented Programming: 7th European Conference Kaiserslautern, Germany, July 26–30, 1993 Proceedings 7. Springer Berlin Heidelberg, 1993.

[10] M. Völter, T. Stahl, J. Bettin, A. Haase, and S. Helsen, *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2013.

[11] A. Ghorbani, "Structuring Socio-technical Complexity - Modelling Agent Systems Using Institutional Analysis," 2013.