Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

TOWARDS REUSABLE BUILDING BLOCKS TO DEVELOP COVID-19 SIMULATION MODELS

Shane A. Schroeder
Christopher Vendome
Philippe J. Giabbanelli

Department of Computer
Science & Software Engineering
Miami University
205W Benton Hall, 510 E. High St.
Oxford, OH 45056, USA

Alan M. Montfort

IMT Mines Ales
6 Av. de Clavières
30100 Alès, FRANCE

ABSTRACT

Modeling & Simulation has played an essential role in supporting the decision-making activities of policymakers for COVID-19. However, a proliferation of models has been noted in the literature, and new models are only more likely to emerge given the shift to long-term management of the disease and the call for highly tailored tools. Having a multiplicity of models can have benefits, for example when contributing to ensembles of models. However, if each model is created from scratch, there is significant redundancy in efforts hence time inefficiency and a heightened risk of bugs. Our study examines the naturally occurring practices of modelers who wrote COVID-19 models in NetLogo to identify redundancy in code and thus suggest reusable 'building blocks' that would speed-up the process of model development as well as improving code quality. Based on 28 models, we identified five themes and discussed their transformation into potential building blocks for simulation.

1 INTRODUCTION

COVID-19 has caused over 6 million deaths worldwide, out of almost half a billion cases. As an example, COVID-19 is a leading cause of death in the USA (Woolf, Chapman, and Lee 2021), where it is among the contributing factors explaining the largest decline in life expectancy in over 75 years (Stephenson 2022) – a decline that exceeds the effects of *both* World War II and the flu pandemics that followed. Several studies showed that life expectancy decreased in most of the countries that were examined (Aburto et al. 2022; Islam et al. 2021), which underscores the global impact of the disease. Given the eventual shift from a pandemic to an endemic phase, the global community will need to manage COVID-19 for the long term, without any 'misplaced complacency' (Katzourakis 2022). There is thus an ongoing need for tools that support the decision-making processes of policymakers with regard to specific populations (e.g., elderly, children), places, and times (e.g., prevalence, seasonal effects). Modeling & Simulation has been a part of this toolbox from the very beginning (Lorig et al. 2021; Padmanabhan et al. 2021; Childs et al. 2021) and has had to face numerous transitions (Giabbanelli et al. 2021), from new soft skills (e.g., communicating expectations and results with the general population or policymakers) to improvements in data (e.g., detailed mobility datasets as in Elarde et al. 2021 and the various under-exploited sources mentioned by Santosh 2020) and more comprehensive models (e.g., accounting for vaccine logistics as in Liu and Lou 2022).

The growing field of M&S applied to COVID-19 has often been portrayed as a "proliferation of models, often diverging widely in their projections", thus calling into question "to what extent results can be trusted" (James et al. 2021). Practices have improved, for example by shifting from a single model

to *ensembles* of models (Kim et al. 2020), which can be accessed by policymakers via interactive web portals for detailed queries about places and times (Srabanti et al. 2021). Instead of picking 'whichever curve' (Hutson 2020) best aligned with a message, we thus rely on aggregated projections (e.g., via the COVID-19 Forecast Hub). Similarly, lessons learnt from early mistakes in science communication (Kreps and Kriner 2020; Van Dooren and Noordegraaf 2020) help us to more cautiously explain model uncertainty and increase public involvement in modelling (Harvard et al. 2021). There is also a heightened awareness in the M&S community about the tradeoff between model development costs and possible harms caused by using the model for high-stakes decisions (Horner and Symons 2020). Finally, the wide confidence margins of some models can be narrowed thanks to a much larger number of simulation repeats, made possible by various forms of simulation acceleration (Kulkarni et al. 2022; Lutz and Giabbanelli 2022).

Despite these changes, the basic observation remains: there is a multitude of models, and calls to create 'highly tailored models' for specialized cross-sections of the population (Mokhtari 2022) can further fuel this proliferation. Having so many models is *not necessarily a problem*. After all, they serve to answer very precise questions at a local level: they differ in purpose, data, and stakeholders. Developing a myriad of local models is thus one approach to ensure that each one is adequate, and having multiple models helps to build rigorous aggregates. The challenge is when *each of these models is developed from scratch*, which is often the case (Giabbanelli et al. 2021) despite the availability of COVID-19 frameworks such as (Kerr et al. 2021) which can be customized for a population (Li and Giabbanelli 2021b) or extended to account for new interventions (Li and Giabbanelli 2021a). Repeatedly developing models with the same techniques (e.g., Agent Based Models, Compartmental Models) for the same disease leads to two issues from a software engineering viewpoint. First, re-inventing the wheel is *potentially* inefficient, thus leading to elevated model development costs. Second, it raises the possibility of bugs (Rahman and Farhana 2020), which may be difficult to find since model code is not systematically shared (Barton et al. 2020).

In this paper, we examine whether COVID-19 simulation models independently created by several teams share commonalities at the code level. The identification of such *reusable* code fragments or 'building blocks' is a key step to then provide modelers with open and extensively tested libraries that can be easily integrated in modeling projects, thus addressing potential inefficiencies in model implementation while improving code quality. Our study focuses on the naturally occurring practices for COVID-19 simulation models coded in `NetLogo`, which is one of the environments used in many publications (Daghriri and Ozmen 2021; Cotfas et al. 2020; Li et al. 2021). `NetLogo` allows users to create and import libraries, as observed in several COVID-19 projects (Saleem et al. 2022); hence, our identification of building blocks in this environment can support later efforts at creating and integrating libraries specific to COVID-19.

The remainder of this paper is organized as follows. In Section 2, we explain the notion of building blocks within modeling generally, as well as within `NetLogo` specifically in the form of libraries. In Section 3, we summarize our methods to identify and analyze `NetLogo` models of COVID-19. Results are presented in Section 4, and their implications for building blocks are analyzed in Section 5.

2 BACKGROUND

Building blocks are generic and adaptable components that help simplify and improve the quality of models built with them. Building blocks describe a documented and validated process so that it can be properly implemented in a model. Since blocks should be *validated* (i.e. accurate results based on a range of real world input), the modeler does not have to build a process up from scratch, which might introduce bias and error into the model. In Section 2.1, we summarize the requirement to create and use building blocks, and briefly contrast them with the related notion of sub-models. In Section 2.2, we exemplify the use of building blocks in the context of `NetLogo`; note that building blocks, libraries, and extensions are used interchangeably by modelers employing `NetLogo` to refer to the same idea.

2.1 Sub-models and Building Blocks: General Modeling Perspectives

Sub-models constitute a distinct part of a larger model. That is, they describe an 'inner' model that has been embedded in an 'outer' model. This embedding makes them less generic and more purpose built to the models in which they are employed. For example, a sub-model in an Agent-Based Simulation could take in the agent's observation, run a machine learning algorithm (e.g., neural network), and output the decision for the agent (Negahban and Giabbanelli 2021). In contrast, building blocks are defined as a basic unit from which models are built upon and with (Verbraeck and Valentin 2008). Blocks are supposed to be self-contained, *inter-operable*, *reusable components*. A building block is usually *dedicated to a domain* (Valentin et al. 2005; Carley 2019), thus saving time to users of building blocks who do not have to invest as heavily in finding relevant theoretical constructs and/or propose new complex rules. For example, due to lockdown from COVID-19, commuting patterns were modified and ultimately pollution was affected. A model for large scale impacts of the lockdown could thus rely on building blocks for household water consumption, wastewater processing, and pollution (Dobson et al. 2021). In short, sub-models are usually specific to a model and are less generic, whereas building blocks are meant to be generic and reusable in many different models across different domains.

Each part of a Model Building Block describes a specific calculation or a step in a process, e.g. the temperature impact of light on plankton growth (Albertyn and Kruger 2003). The block as a whole is mainly characterized by its in- and out-going connectors, user dialog, and documentation. Independence and reusability are key requirements to turn model components into building blocks. De Kok et al. (2010) detail these requirements, including the quality of the documentation (e.g., thoroughness, clarity about the functionality), encapsulation (i.e., the implementation can be hidden), easy to adapt, and scientifically correct and mature. Reusability can be a challenge, as the frameworks employed by building blocks may be specific to the needs and capacity of their own development team (Voinov and Shugart 2013). In order to cope with that issue, blocks should (i) include data sets that exemplify to users how to communicate with the block and (ii) favor standard methods of data importing and exporting.

To use a building block, users first identify where they can support a model, then search if the block exists, and finally incorporate it or 'integrate it' into the model by making proper adjustments in the code for inter-operability. Note that integration can be a complex step (Belete et al. 2017), involving pre-integration assessment to check requirements, orchestration during simulation, and transformations to ensure inter-operability in terms of structure and semantics (e.g., to deal with spatial and/or time inconsistencies between blocks as discussed in Iwanaga et al. 2021).

2.2 Building Blocks as Libraries: The Case of NetLogo Extensions

There are two diametrically opposed motivations to create a building block. On the one hand, modelers can engage in requirement gathering, for example by identifying and engaging with potential users to list their needs, or by examining how tasks commonly accomplished within a given context define a set of needs (Giabbanelli et al. 2019). On the other hand, modelers may assume that what they created for themselves may be of interest to others, echoing the phrase "if you build it, they will come". Knowing which of these two approaches is most common in the case of `NetLogo` thus contributes to situating the contributions of our work, which is concerned with requirement gathering for COVID-19.

We examined seven articles that introduced new extensions for `NetLogo`. Based on this sample, the creation of building blocks is mostly driven by the needs of the authors (5 out of 7) rather than a demonstrated desire for functionalities from the modeling community (Table 1). For example the ontology (Polhill 2015), BDI (Sakellariou et al. 2008; Wiens and Monett 2013), GIS (Walker and Johnson 2019) and HubNet (Muscalagiu, Emil, and Negru 2014) extensions often explained the proposed usefulness of the extension, but without evidence or a mention of the specific community needs that functionality. Few works (two out of seven) discussed a need from the community: LevelSpace Extension (Hjorth et al. 2020) and BDI Extension Expansion (Maleš and Ribarić 2016). For example, the LevelSpace extension justifies

the need from the community based on a survey (Morvan 2012) about the increased research interest (as evidenced by a growing number of publications) in multi-level agent-based modeling.

Note that Table 1 does *not* demonstrate the absence of a need. For example, Geographic Information Systems are widely used in Agent-Based Model, and this extension is one of the most used in practice (Vendome et al. 2020). Rather, Table 1 shows that (i) building blocks are developed but (ii) the need is either assumed or implicit. While there exists qualitative research to study software engineering processes and practices (Zhang et al. 2019), this line of work has not yet been applied to the building blocks used in `NetLogo` models, which suggests that our approach (detailed in the next section) is more unique within this context.

Table 1: Table of assessed NetLogo extension papers.

Extension Name	**What does it do?**	**Evidence that the extension addresses a need from the modeling community**
Ontology (owl)	Extract ontologies from simulation models at a specific tick	None
LevelSpace (ls)	Build multi-level agent-based model systems with hundreds or thousands of concurrent models	Growing research topic of Multi-level Agent-based models, as evidenced by a survey
BDI (beliefs-desires-intentions)	Allows agents to now exhibit beliefs, desires, and intentions to more accurately depict behaviors.	None.
BDI Expansion	Improved upon the original BDI extension by creating 'clusters', which are clustered intentions.	The Berlin School of Economic & Law used the library for curriculum development, but nothing is mentioned for other communities
Geographic Information System (GIS)	Includes spatial information in models (e.g., maps)	Mention of a need to increase accuracy by using realistic maps. No need mentioned from the community.
HubNet	Runs models on various devices, such as mobile devices (e.g., Android, iOS)	None
Type-2 Fuzzy Inference System (FIS)	Handles linguistic/numerical uncertainties by representing concepts (e.g., large/small, fast/slow) as fuzzy sets	None

3 METHODS

3.1 Model Selection

In line with our previous work on mining `NetLogo` models (Vendome et al. 2020), we searched for them on two platforms: (i) https://comses.net/, since it is the main hosting platform for academic ABM code per the review of Janssen (2017); and (ii) GitHub.com, which has a leading position as a hosting platform for software development. We searched for files including the keyword 'COVID' and bearing the extension 'nlogo', used by `NetLogo`. Models were identified in July 2021, then analyzed both manually for building blocks (Section 3.2) and automatically for software quality (Section 3.3).

3.2 Qualitative Analysis: Thematic Analysis

The objective of our qualitative analysis is to manually identify commonalities (or 'themes') across NetLogo codes on COVID-19. This is performed by using thematic analysis, one of the most common methods within qualitative research. The two student authors received training on thematic analysis and performed it via a four steps process. First, they 'transcribed' the data, which simply means moving it from NetLogo files into Word such that they can be annotated. Students then independently tagged codes of interest with labels created on the fly, such as 'initialize agent traits', or 'placing agents in locations'. These labels were then combined to generate initial themes; for example, both cited labels pertain to the theme of 'agent heterogeneity'. Themes were reviewed and reworked by re-reading the models, thus arriving at a final list of themes. The objective is often to arrive at five to seven themes. These steps are detailed in numerous textbooks (Guest et al. 2011), with a few examples of their application in software engineering (Cruzes and Dyba 2011). For concision, the analysis of the model with the *least* code is shown in Figure 1. The annotations for every model are available at the Open Science Framework portal https://osf.io/d7vqa/ to provide full transparency into our analysis of publicly available models.

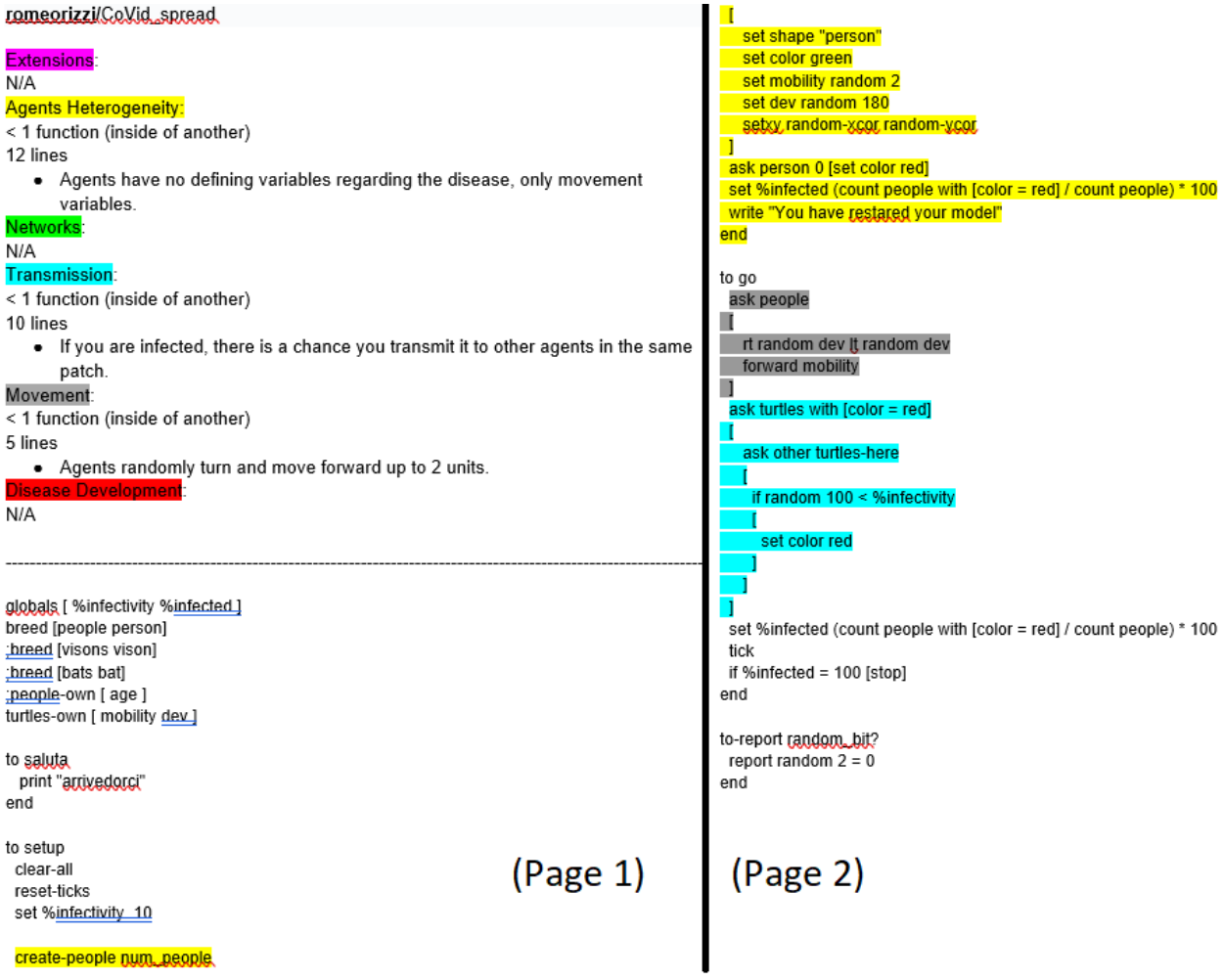

Figure 1: Thematic analysis for the model with the fewest lines of code. Our themes are first listed, with a brief description, the corresponding color, and the number of lines in the model. These colors are then used to highlight the original code. All codes with highlights are available at https://osf.io/d7vqa/.

3.3 Quantitative Analysis: Software Quality

We used our custom parser to characterize each code with respect to metrics commonly used in software engineering related complexity and readability. Several of the metrics tell us about the level of sophistication of the implementation, for example by the number of functions, use of more advanced control-flow structures (nested blocks), reliance on diverse types of agents (breeds), or the diversity of `NetLogo` constructs utilized by the programmer (Unique Keywords). More sophisticated implementations would suggest that individuals implementing these models are sufficiently skilled with the language to eventually integrate building blocks. We also examined the efforts of programmers on creating quality code, as evidenced by the use of quality identifiers for variables (at least three letters) and commenting. Greater efforts may suggest that programmers are willing to spend the time on a quality model, thus they may be receptive to the idea of using validating building blocks in the future. We also recorded the number of building blocks already in use (dependencies).

4 RESULTS

4.1 Thematic Analysis

We identified 28 models, with most originating from GitHub (n=21) and a few from Comses (n=7). Our *qualitative analysis* focused on five themes (Figure 2): heterogeneity of the agents, their movements and decision-making processes, disease transmission, disease development, and contact networks. *Agent Heterogeneity* is used in all models, taking 38.9 ± 46.1 lines on average to ensure that agents are diverse with respect to demographic variables, most commonly age and health status, and a variety of model-specific variables (e.g., nationality, income, trust in government). It also involves COVID-specific constructs, such as mask wearing and their efficacy. Most of the time, those variables rely on official datasets. Initialization also involves spatial variables to set the initial location of each agent, such as the positions of various staff in a restaurant. While spatial aspects of initializations are rare and varied, demographic aspects form repetitive sequences of assigning values to variables, which gets very lengthy (up to 247 lines).

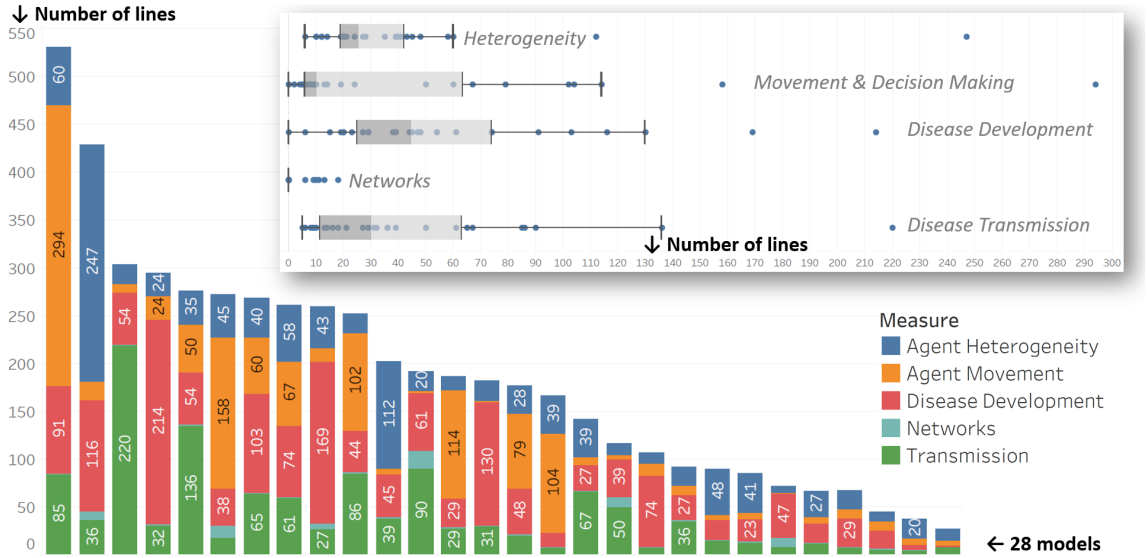

Figure 2: Number of lines devoted to each theme per model (left) and across models (top inset).

The vast majority of models (26 out of 28) account for *movements and decision-making processes* in the agents, dedicating on average 42.3 ± 65.1 lines to this purpose. About half of the time (14 out of 26), these movements are random, hence agents are akin to spheres drifting through space and (optionally) bouncing off the walls. These codes are highly similar and may be abstracted by making use of a few variables, for example for the speed of drifting and whether the environment is closed. The other half

of the time, agents have purposeful movements, accounting for the nature of the destination (e.g., taking the elevator to a different floor, shopping or working, waiting in line) and their own status (e.g., essential worker, infected). While the codes differ in these highly customized models, they often share the abstract notion of identifying certain types of location and moving agents through them based on simple conditions.

Disease transmission happens in all models, taking on average 43.8 ± 47.2 lines. The overwhelming pattern (25 out of 28 models) consists of checking for each susceptible agent whether its neighbors are infected; if so, we compute the probability of infection depending on characteristics such as age and personal precautions, which abstract preventative measures including mask wearing. There is thus a high potential for simplification in most models. In contrast, there are only three models in which the spread can happen via infected surfaces. These three models account for aspects as varied as sanitation procedures, air flow (e.g., confined public spaces vs. open spaces), or sneezing/coughing frequencies.

Since disease transmission is mostly a stochastic event when agents are close to each other, it is not necessary to use *networks*. This may explain why they are only used by 6 out of 28 models. Network codes are short, comprised of 2.4 ± 4.9 lines on average, and serve more often to create a complete social network of the population (4 out of 6 models) rather than for contact tracing after an agent is exposed (2 out of 6 models). There are thus little opportunities to simplify network codes by building blocks.

All models handle *disease development*, devoted on average 57.7 ± 49.9 lines to this goal. Depending on age, agents will express various forms of symptom severity and ultimately either recover or die. Additional aspects include the handle of isolation and quarantine, or whether individuals were vaccinated. Few models represent the health system (5 out of 28), through the abstract notion of 'being hospitalized' or 'sent to the ICU', which affects the severity of the infection. Although disease development tends to be the most complex part of the models, the models have several parts in common in terms of both states and transition, owning to the underlying SIR and SEIR compartmental models of disease progression.

Table 2: Results for each software metrics. 'COVID Models' refer to the 28 `NetLogo` models identified here, while CoMSES and GitHub refer to characteristics of `NetLogo` models found on these platforms in general, per our previous work (Vendome et al. 2020).

Category	Metric	COVID Models		CoMSES		GitHub	
		Mean±Std Dev	Median	Mean±Std Dev	Median	Mean±Std Dev	Median
Readability	# Blank Lines	93.5 ± 109.5	58	114.0 ± 123.6	69	76.8 ± 90.3	49
	#Breeds	3.6 ± 10.4	0	3.7 ± 6.6	2	2.9 ± 4.5	2
	#Comments	111.9 ± 132.1	64	150.7 ± 167.0	99.5	87.5 ± 111.3	55
	#Conditionals	57.4 ± 56.9	51	74.6 ± 91.2	44.5	39.6 ± 48.9	22
	#Loops	17.8 ± 19.7	13	20.5 ± 22.1	14.5	16.0 ± 25.1	9
	#Identifiers	21 ± 32.7	7	40.1 ± 45.8	24	18.6 ± 23.7	10
	#Quality Identifiers	19.0 ± 29.7	7	33.1 ± 39.1	20	15.1 ± 19.9	8
	#Nested Blocks	1.4 ± 3.5	0	2.4 ± 10.8	0	2.9 ± 18.5	0
	#Keywords	444.7 ± 454.0	315	673.6 ± 636.6	529	361.5 ± 382.6	215
	#Unique Keyword	36.6 ± 15.8	34	47.1 ± 19.3	45	36.7 ± 14.8	34
	Func Size (Avg)	16.4 ± 7.5	16.29	22.8 ± 16.0	18.6	17.8 ± 14.4	12.6
	Func Size (Max)	68.2 ± 59.9	54	134.7 ± 367.2	67.5	64.4 ± 84.7	39
	Line Length (Avg)	38.3 ± 23.3	32.91	43.5 ± 14.9	41.1	35.6 ± 12.1	32.9
	Line Length (Max)	176.6 ± 71.2	161	397.2 ± 1478.4	211	172.9 ± 84.6	156
Complexity	# Dependencies	0.5 ± 0.9	0	0.8 ± 1.3	0	0.5 ± 0.8	0
	# Functions	22 ± 18.3	19	22.5 ± 19.7	17	17.2 ± 15.9	13

4.2 Software Quality

For the quality evaluation, we focused on metrics related to readability and complexity, which impact comprehension of the code and the ability to modify existing code. Our chosen readability metrics are

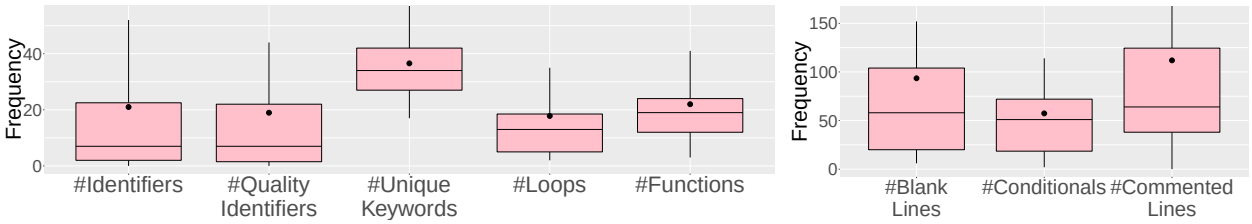

Figure 3: Distribution of software metrics; note the two different vertical (y-axis) scales.

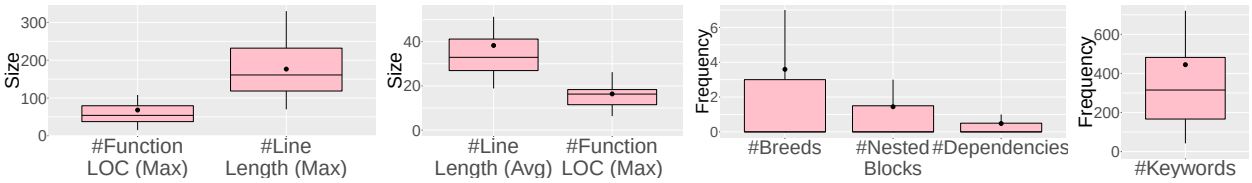

Figure 4: Distribution of software metrics quantified either in size (left) or frequency (right).

derived from the prior work of Dorn (2012) and Buse and Weimer (2008) and have been leveraged in recent work (Scalabrino et al. 2021). While their readability models are based on correlative analysis for features like line length or number of loops and conditionals, there are other works that have considered it from a cognitive perspective. For example, lower quality and number of identifiers can reduce the ability to understand some existing code based on limitations of human memory (Binkley et al. 2008). Similarly, guidelines suggest function size to be smaller (20-100 lines of code) (Martin 2008), since programmers are typically limited to what they currently see immediately on the screen and recalling information not currently visible increases the cognitive load of memory (Weinberg 1985). Additionally, comments in code have been shown to increase understanding (Takang et al. 1996). Readability of software is an important component of understanding software and has been shown as the dominant activity, taking over 40% of the time, when understanding some code (LaToza, Venolia, and DeLine 2006). From the complexity metrics, the number of functions provides insight on how many different behaviors are implemented (if the model has more behaviors, it is more complex). The number of dependencies relates to the coupling of a model to other libraries, which increases complexity through this linkage to the dependencies.

Although most of our models (21 out of 28) come from GitHub, software metrics are higher than expected in `NetLogo` models from GitHub (Table 2; Figures 3 and 4) for all but two aspects (number of nested blocks and average function size). Indeed, we note significantly more comments, conditional statements, types of agents (i.e., breeds), or functions. In sum, the models are not yet at the level of a CoMSES sample (often used by professionals to support peer-reviewed articles), but they are more sophisticated than the average GitHub model in several aspects. The metrics suggest that programmers (i) have a sufficient level of skills in `NetLogo` and (ii) are are willing to put time into programming these models, which are important conditions to later (i) be able to integrate a building block into the code and (ii) be receptive to using building blocks as a means to improve code quality. The need for building blocks is demonstrated both by the redundancy of functionality across themes (section 4.1) and the current lack of libraries. Indeed, 20 codes do not use any library and the other ones use few, which result in an average of 0.5 ± 0.9 library per code. Only three libraries are used in more than one case: `gis` is used twice to load maps, while `csv` and `profiler` are used three times, to load comma-separated files and monitor time spent on each function, respectively.

5 DISCUSSION

The prolific creation of simulation models for COVID-19 is set to continue, given the need to manage the disease for the long-term and the shift to highly tailored models supporting local decision-making activities.

Our previous analysis of models in `NetLogo` together with the present analysis devoted to COVID-19 show that modelers rarely use building blocks to develop these models. When building blocks (a.k.a. 'libraries' or 'extensions') are involved, they primarily support input/output operations such as loading maps or data files, or serve as timers. The functions of each model are thus written from scratch repeatedly by modelers, which can result in inefficiency for development and raises the risk of bugs.

Our work shows that COVID-19 models in `NetLogo` are more sophisticated than the average model on GitHub, which suggests that modelers have the skills to work with libraries and are willing to spend time on their models. To identify which libraries are needed, we performed a thematic analysis on the codes of 28 models and identified five themes. Three themes were present in all models and had significant code redundancies: *heterogeneity* was achieved across models by repeatedly setting up a set of demographic variables, *disease transmission* checked whether neighbors were infected and then computed a probability of infection, and *disease development* followed classic compartmental models (SIR or SEIR). *Movements and decision-making processes* were present in almost all codes; they either abstracted agents 'drifting' across space, or gave them series of conditions to move between designated locations. Although *networks* is the only theme for which a library already exists (`Nw`), this theme was rarely present in models hence there would be little benefits in transforming codes to use the library.

Once a *need* for a library is identified, the next steps consist of designing, implementing, and disseminating each library. Our work contributes to guiding these future steps by offering several possibilities, based on our observations in the practices of modelers. For *agent heterogeneity*, we suggest to create one building block that initializes agents' demographics from a standard configuration file, which can be as simple as a CSV file (each line is one attribute followed by instructions to initialize it) or as comprehensive as an Excel file (consisting of tables from the statistics office as in Hunter, Mac Namee, and Kelleher 2018). Moving some of the agents' initialization into a file instead of being hard-coded would also have two additional benefits. First, models are easier to compare: using the configuration of one model for another can serve to contrast results on the basis of differences in rules rather than in population. Second, models are easier to run on different populations, either due to a naturally occurring change in the demographic or to account for different demographic scenarios.

For *disease transmission* and *disease development*, it may not be desirable to split these two notions through different libraries. Rather, we observed that the 'skeleton' used in every model was either a SIR or SEIR compartmental model, which is relevant for transmission (how susceptible individuals become sick) and development (how sick individuals recover or die). It may thus be possible to use a library from which the user can point to a desired template, configure the relevant state transitions, and then add the custom ones. For example, dozens of lines may be summarized as follows:

```
load-compartments('SEIR')
set-transition('S->E','rate',0.4)
set-transition('E->I','pdf',lognormal(4.1, 4.8))
set-transition('I->R','pdf',lognormal(1, 1.8))
add-state('vaccine')
add-transition('S->vaccine','rate',0.2)
#no transitions out of the vaccine state hence agents cannot be exposed
```

In contrast, *movements and decision-making processes* may require two libraries as modelers were split about equally between drastically different approaches. One set of modelers did not represent physical space, hence agents were randomly moving, optionally bouncing off the walls. This could be achieve solely by calling a (hypothetical library) function such as `move-randomly(distance,angle,bounceWalls?)`. Another set of modelers defined locations and activities that take agents between them, for example going home after work. The corresponding abstraction consists of a graph, where locations are nodes and transitions are labeled edges. While this can be straightforwardly handled in a multi-paradigm environment

such as `AnyLogic`, `NetLogo` is primarily geared towards Agent-Based Models and it is not designed to alternate between graphical representations of a system.

Our examples showed that the translation of an identified need for libraries into their enactment requires further work. In particular, we noted that needs and libraries do not have a one-to-one mapping, as several needs may best be served by a joint library while variations of another need may require different libraries.

6 CONCLUSION

A physician recently described how "the pandemic's greatest source of danger has transformed from a *pathogen* into a *behavior*", as the emphasis is now on modifiable health risks such as vaccination (Mazer 2022). From a modeling viewpoint, providing building blocks for the relatively stable pathogen part (e.g., disease transmission and development) and basic aspects of human behaviors (e.g., heterogeneity in demographics, random or scheduled movements) would allow modelers to devote their effort to aspects that are truly unique in their model, such as the more complex facets of human behavior (e.g., vaccine hesitancy). Our identification of redundancies in codes and its organization into themes thus support modelers in providing such building blocks.

REFERENCES

Aburto, J. M., J. Schöley, I. Kashnitsky, L. Zhang, C. Rahal, T. I. Missov, M. C. Mills, J. B. Dowd, and R. Kashyap. 2022. "Quantifying Impacts of the COVID-19 Pandemic Through Life-Expectancy Losses: a Population-Level Study of 29 Countries". *International Journal of Epidemiology* 51(1):63–74.

Albertyn, M., and P. S. Kruger. 2003. "Generic Building Blocks for Simulation Modelling of Stochastic Continuous Systems". *South African Journal of Industrial Engineering* 14(2):47–61.

Barton, C. M., M. Alberti, D. Ames, J.-A. Atkinson, J. Bales, E. Burke, M. Chen, S. Y. Diallo, D. J. Earn, B. Fath et al. 2020. "Call for Transparency of COVID-19 Models". *Science* 368(6490):482–483.

Belete, G. F., A. Voinov, and G. F. Laniak. 2017. "An Overview of the Model Integration Process: From Pre-Integration Assessment to Testing". *Environmental Modelling & Software* 87:49–63.

Binkley, D., D. Lawrie, S. Maex, and C. Morrell. 2008. "Impact of Limited Memory Resources". In *2008 16th IEEE International Conference on Program Comprehension*, 83–92.

Buse, R. P., and W. R. Weimer. 2008. "A Metric for Software Readability". In *Proceedings of the 2008 International Symposium on Software Testing and Analysis*, ISSTA '08, 121–130. New York, NY, USA: Association for Computing Machinery.

Carley, K. M. 2019. "Social-Behavioral Simulation: Key Challenges". *Social-Behavioral Modeling for Complex Systems*:741–752.

Childs, M. L., M. P. Kain, M. J. Harris, D. Kirk, L. Couper, N. Nova, I. Delwel, J. Ritchie, A. D. Becker, and E. A. Mordecai. 2021. "The Impact of Long-Term Non-Pharmaceutical Interventions on COVID-19 Epidemic Dynamics and Control: The Value and Limitations of Early Models". *Proceedings of the Royal Society B* 288(1957):20210811.

Cotfas, L.-A., C. Delcea, R. J. Milne, and M. Salari. 2020. "Evaluating Classical Airplane Boarding Methods Considering COVID-19 Flying Restrictions". *Symmetry* 12(7):1087.

Cruzes, D. S., and T. Dyba. 2011. "Recommended Steps for Thematic Synthesis in Software Engineering". In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, 275–284. IEEE.

Daghriri, T., and O. Ozmen. 2021. "Quantifying the Effects of Social Distancing on the Spread of COVID-19". *International Journal of Environmental Research and Public Health* 18(11):5566.

De Kok, J.-L., G. Engelen, and J. Maes. 2010. "Towards Model Component Reuse for the Design of Simulation Models–a Case Study for ICZM". In *Proceedings of the 5th International Congress on Environmental Modelling and Software*.

Dobson, B., T. Jovanovic, Y. Chen, A. Paschalis, A. Butler, and A. Mijic. 2021. "Integrated modelling to support analysis of COVID-19 impacts on London's water system and in-river water quality". *Frontiers in Water* 3:641462.

Dorn, J. 2012. "A general software readability model". Master's thesis, University of Virginia, Charlottesville, VA, US.

Elarde, J., J.-S. Kim, H. Kavak, A. Züfle, and T. Anderson. 2021. "Change of Human Mobility During COVID-19: A United States Case Study". *PloS one* 16(11):e0259031.

Giabbanelli, P., M. Fattoruso, and M. L. Norman. 2019. "Cofluences: Simulating the Spread of Social Influences via a Hybrid Agent-Based/Fuzzy Cognitive Maps Architecture". In *Proceedings of the 2019 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 71–82.

Giabbanelli, P. J., J. Badham, B. Castellani, H. Kavak, V. Mago, A. Negahban, and S. Swarup. 2021. "Opportunities and Challenges in Developing COVID-19 Simulation Models: Lessons From Six Funded Projects". In *2021 Annual Modeling and Simulation Conference (ANNSIM)*, 1–12. IEEE.

Guest, G., K. M. MacQueen, and E. E. Namey. 2011. *Applied Thematic Analysis*. SAGE Publications.

Harvard, S., E. Winsberg, J. Symons, and A. Adibi. 2021. "Value Judgments in a COVID-19 Vaccination Model: A Case Study in the Need for Public Involvement in Health-Oriented Modelling". *Social Science & Medicine* 286:114323.

Hjorth, A., B. Head, C. Brady, and U. Wilensky. 2020. "Levelspace: A Netlogo Extension for Multi-Level Agent-Based Modeling". *Journal of Artificial Societies and Social Simulation* 23(1).

Horner, J. K., and J. F. Symons. 2020. "Software Engineering Standards for Epidemiological Models". *History and Philosophy of the Life Sciences* 42(4):1–24.

Hunter, E., B. Mac Namee, and J. Kelleher. 2018. "An Open-Data-Driven Agent-Based Model to Simulate Infectious Disease Outbreaks". *PloS One* 13(12):e0208775.

Hutson, M. 2020. "The Mess Behind the Models: Too Many of the COVID-19 Models Led Policymakers Astray. Here's How Tomorrow's Models Will Get it Right". *IEEE Spectrum* 57(10):30–35.

Islam, N., D. A. Jdanov, V. M. Shkolnikov, K. Khunti, I. Kawachi, M. White, S. Lewington, and B. Lacey. 2021. "Effects of COVID-19 Pandemic on Life Expectancy and Premature Mortality in 2020: Time Series Analysis in 37 Countries". *BMJ* 375.

Iwanaga, T. et al. 2021. "Socio-Technical Scales in Socio-Environmental Modeling: Managing a System-of-Systems Modeling Approach". *Environmental Modelling & Software* 135:104885.

James, L. P., J. A. Salomon, C. O. Buckee, and N. A. Menzies. 2021. "The Use and Misuse of Mathematical Modeling for Infectious Disease Policymaking: Lessons for the COVID-19 Pandemic". *Medical Decision Making* 41(4):379–385.

Janssen, M. A. 2017. "The Practice of Archiving Model Code of Agent-Based Models". *Journal of Artificial Societies and Social Simulation* 20(1).

Katzourakis, A. 2022. "COVID-19: Endemic Doesn't Mean Harmless.". *Nature*:485–485.

Kerr, C. C. et al. 2021. "Covasim: an Agent-Based Model of COVID-19 Dynamics and Interventions". *PLOS Computational Biology* 17(7):e1009149.

Kim, J.-S., H. Kavak, A. Züfle, and T. Anderson. 2020. "COVID-19 Ensemble Models Using Representative Clustering". *SIGSPATIAL Special* 12(2):33–41.

Kreps, S. E., and D. L. Kriner. 2020. "Model Uncertainty, Political Contestation, and Public Trust in Science: Evidence From the COVID-19 Pandemic". *Science Advances* 6(43):eabd4563.

Kulkarni, S., M. M. Krell, S. Nabarro, and C. A. Moritz. 2022. "Hardware-Accelerated Simulation-Based Inference of Stochastic Epidemiology Models for COVID-19". *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 18(2):1–24.

LaToza, T. D., G. Venolia, and R. DeLine. 2006. "Maintaining Mental Models: A Study of Developer Work Habits". In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, 492–501. New York, NY, USA: Association for Computing Machinery.

Li, J., and P. Giabbanelli. 2021a. "Returning to a Normal Life via COVID-19 Vaccines in the United States: a Large-Scale Agent-Based Simulation Study". *JMIR Medical Informatics* 9(4):e27419.

Li, J., and P. J. Giabbanelli. 2021b. "Identifying Synergistic Interventions to Address COVID-19 Using a Large Scale Agent-Based Model". In *International Conference on Computational Science*, 655–662. Springer.

Li, K. K., S. A. Jarvis, and F. Minhas. 2021. "Elementary Effects Analysis of Factors Controlling COVID-19 Infections in Computational Simulation Reveals the Importance of Social Distancing and Mask Usage". *Computers in Biology and Medicine* 134:104369.

Liu, K., and Y. Lou. 2022. "Optimizing COVID-19 Vaccination Programs During Vaccine Shortages: A Review of Mathematical Models". *Infectious Disease Modelling* 7:286–298.

Lorig, F., E. Johansson, and P. Davidsson. 2021. "Agent-Based Social Simulation of the COVID-19 Pandemic: A Systematic Review". *JASSS: Journal of Artificial Societies and Social Simulation* 24(3).

Lutz, C. B., and P. J. Giabbanelli. 2022. "When Do We Need Massive Computations to Perform Detailed COVID-19 Simulations?". *Advanced Theory and Simulations* 5(2):2100343.

Maleš, L., and S. Ribarić. 2016. "A Model of Extended BDI Agent with Autonomous Entities (Integrating Autonomous Entities Within BDI Agent)". In *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, 205–214. IEEE.

Martin, R. C. 2008. *Clean Code: A Handbook of Agile Software Craftsmanship*. 1 ed. USA: Prentice Hall PTR.

Mazer, B. 2022. "COVID Won't End Up Like the Flu. It Will Be Like Smoking.". *The Atlantic*.

Mokhtari, A. 2022, January. "Learning From Covid-19 Requires a Modeling Renaissance". *STAT*.

Morvan, G. 2012. "Multi-Level Agent-Based Modeling-A Literature Survey". *arXiv preprint arXiv:1205.0561*.

Muscalagiu, I., P. H. Emil, and V. Negru. 2014. "Enhancing DisCSP-Netlogo From Simulation to Real-Execution of Agents in Distributed Constraints". *Procedia Computer Science* 35:261–270.

Negahban, A., and P. J. Giabbanelli. 2021. "Hybrid Agent-Based Simulation of Adoption Behavior and Social Interactions: Alternatives, Opportunities, and Pitfalls". *IEEE Transactions on Computational Social Systems*.

Padmanabhan, R., H. S. Abed, N. Meskin, T. Khattab, M. Shraim, and M. A. Al-Hitmi. 2021. "A Review of Mathematical Model-Based Scenario Analysis and Interventions for COVID-19". *Computer Methods and Programs in Biomedicine* 209:106301.

Polhill, J. G. 2015. "Extracting OWL Ontologies From Agent-Based Models: A Netlogo Extension". *Journal of Artificial Societies and Social Simulation* 18(2):15.

Rahman, A., and E. Farhana. 2020. "An Exploratory Characterization of Bugs in COVID-19 Software Projects". *arXiv preprint arXiv:2006.00586*.

Sakellariou, I., P. Kefalas, and I. Stamatopoulou. 2008. "Enhancing NetLogo to Simulate BDI Communicating Agents". In *Hellenic Conference on Artificial Intelligence*, 263–275. Springer.

Saleem, K., M. Saleem, R. Zeeshan, A. R. Javed, M. Alazab, T. R. Gadekallu, and A. Suleman. 2022. "Situation-Aware BDI Reasoning to Detect Early Symptoms of Covid 19 Using Smartwatch". *IEEE Sensors Journal*.

Santosh, K. 2020. "COVID-19 Prediction Models and Unexploited Data". *Journal of Medical Systems* 44(9):1–4.

Scalabrino, S., G. Bavota, C. Vendome, M. Linares-Vásquez, D. Poshyvanyk, and R. Oliveto. 2021. "Automatically Assessing Code Understandability". *IEEE Transactions on Software Engineering* 47(3):595–613.

Srabanti, S., G. E. Marai, and F. Miranda. 2021. "COVID-19 EnsembleVis: Visual Analysis of County-Level Ensemble Forecast Models". In *2021 IEEE Workshop on Visual Analytics in Healthcare (VAHC)*, 1–5. IEEE.

Stephenson, J. 2022. "COVID-19 Deaths Helped Drive Largest Drop in US Life Expectancy in More than 75 Years". In *JAMA Health Forum*, Volume 3, e215286–e215286. American Medical Association.

Takang, A., P. Grubb, and R. Macredie. 1996, 09. "The Effects of Comments and Identifier Names on Program Comprehensibility: An Experimental Investigation". *Journal of Programming Languages* 4:143–167.

Valentin, E. C., S. Steijaert, R. A. Bijlsma, and P. Silva. 2005. "Approach for Modelling of Large Maritime Infrastructure Systems". In *Proceedings of the 2005 Winter Simulation Conference*, 9–pp. IEEE.

Van Dooren, W., and M. Noordegraaf. 2020. "Staging Science: Authoritativeness and Fragility of Models and Measurement in the COVID-19 Crisis". *Public Administration Review* 80(4):610–615.

Vendome, C., D. M. Rao, and P. J. Giabbanelli. 2020. "How Do Modelers Code Artificial Societies? Investigating Practices and Quality of NetLogo Codes from Large Repositories". In *2020 Spring Simulation Conference (SpringSim)*, 1–12. IEEE.

Verbraeck, A., and E. C. Valentin. 2008. "Design Guidelines for Simulation Building Blocks". In *Proceeding of the 2008 Winter Simulation Conference*, 923–932. IEEE.

Voinov, A., and H. H. Shugart. 2013. "'Integronsters', Integral and Integrated Modeling". *Environmental Modelling & Software* 39:149–158.

Walker, B., and T. Johnson. 2019. "NetLogo and GIS: A Powerful Combination". *EPiC Series in Computing* 58:257–264.

Weinberg, G. M. 1985. *The Psychology of Computer Programming*. USA: John Wiley & Sons, Inc.

Wiens, J., and D. Monett. 2013. "Using BDI-extended NetLogo agents in undergraduate CS research and teaching". In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, 1. The World Congress in Computer Science.

Woolf, S. H., D. A. Chapman, and J. H. Lee. 2021. "COVID-19 as the Leading Cause of Death in the United States". *Jama* 325(2):123–124.

Zhang, H., X. Huang, X. Zhou, H. Huang, and M. A. Babar. 2019. "Ethnographic Research in Software Engineering: a Critical Review and Checklist". In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 659–670.

AUTHOR BIOGRAPHIES

SHANE A. SCHROEDER is a Senior Computer Science Student at Miami University, where he performed funded summer research under the supervision of Drs Giabbanelli and Vendome. His email address is schroe51@miamioh.edu.

CHRISTOPHER VENDOME is an Assistant Professor in the Department of Computer Science & Software Engineering at Miami University (USA). His research interests include software evolution and maintenance, as well as software repository mining. His email address is vendomcg@miamioh.edu.

PHILIPPE J. GIABBANELLI is an Associate Professor of Computer Science & Software Engineering at Miami University. He holds a Ph.D. from Simon Fraser University. He has over 100 publications, primarily on Modeling & Simulation and Machine Learning. He is an associate editor for five journals, including SIMULATION. His email address is giabbapj@miamioh.edu.

ALAN MONTFORT is completing his studies in industrial performance and mechatronics at the IMT School of Mines in Ales, which specializes on innovation in the fields of engineering and digital technology. His email is alan.montfort@mines-ales.org.