Transdisciplinarity and the Future of Engineering
B.R. Moser et al. (Eds.)

doi:10.3233/ATDE220656

Generic User Interface for Inclusive Interactive Simulation

Ira WINDER[a,b,1] and Kazuo HIEKATA[b]
[a] *Massachusetts Institute of Technology, Engineering Systems Laboratory, USA*
[b] *University of Tokyo, Industrial Systems Laboratory, Japan*

Abstract. Inclusive interactive simulations are boundary objects that make it possible for engineering know-how to be placed directly into the hands of non-technical stakeholders. However, interactive simulations are also complex technologies that are difficult to implement on a bespoke basis. In order to make deployment of interactive simulations more feasible for engineers, especially those without prior background in user interface design, we created a generic, open-source simulation user interface (OpenSUI) with an associated communication schema. OpenSUI is agnostic to domains, and can communicate with any computational simulation that follows its schema. To demonstrate, we convert an existing computational simulation for real estate, FuzzyIO, into an inclusive interactive simulation using OpenSUI. We then review preliminary, qualitative feedback from non-technical users and discuss our intent for further validation through experimentation at scale.

Keywords. Decision Support Tools and Methods, Democratization of Design, Rapid Prototyping, Methods for Transdisciplinary Engineering, Interactive Simulation for Engineering, Boundary Objects

Introduction

Systems engineers increasingly develop sophisticated, domain-specific computational simulators to design and evaluate solution candidates in transdisciplinary, multi-stakeholder systems design contexts. In multidisciplinary settings, however, computational simulators themselves are not necessarily ideal boundary objects for collaborating with non-technical stakeholders; Rather, such simulations are often operated exclusively by engineer stakeholders who are well-versed in computer programming. As such, non-technical stakeholders might only experience the echo of a simulator via a curated set of solution candidates, thus precluding the possibility of useful and direct feedback with the simulator itself. This is especially worrying in instances where non-technical stakeholders are the key decision-makers for a specific engineering problem, as solution candidates ultimately being presented to them may not be optimized to incorporate their complete knowledge or understanding of the problem.

Inclusive interactive simulation makes it possible for computerized engineering know-how to be placed directly into the hands of non-technical stakeholders, improving the effectiveness of simulations as boundary objects in transdisciplinary design. We observe myriad instances where highly skilled systems engineers have developed such

[1] Corresponding Author, Mail: jiw@mit.edu.

interactive simulators. However, these cases are often bespoke, and appear to only be possible in unlikely instances where engineers have confluent skills in computer science, user interface design, and of course any requisite knowledge of the engineering domain at hand. If the process of creating interactive simulators were not so dependent on such an unlikely confluence of skills, we might reasonably expect to see wider usage of interactive simulation in transdisciplinary engineering projects.

In prior work, we specified the features for a generic user interface that, if implemented, would make it easier for an engineer to deploy a simulator as an interactive simulation, even if the engineer has no background in user interface design [1]. This would be done by having the engineer configure their simulator to conduct two-way communication with a generic user interface via standardized communication protocol, so that a non-technical stakeholder can adjust several variable parameters made available to them. This prior work serves as the basis for building the Open Simulation User Interface (OpenSUI) which we present and discuss in this paper.

Our work is relevant in the context of a Systems Decision Process (SDP), defined as a "collaborative, iterative, and valued-based decision process" by Parnell et al [2]. SDP contextualizes systems as having circular lifecycles that oscillate between problem definition, solution design, decision making, and solution implementation (Figure 1). During SDP, boundary objects are commonly used artifacts for standardizing ideas and discussion between stakeholders. For this purpose, each step in SDP often utilizes unique boundary objects [3].

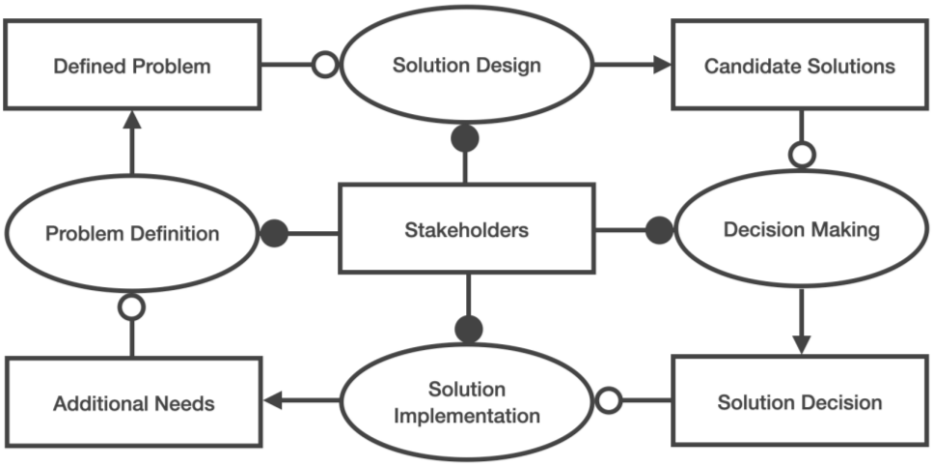

Figure 1. These are the core components of a Systems Decision Process (SDP) illustrated by using an object-processing methodology (OPM). Note the circular feedback loop between 4 key processes (problem definition, solution design, decision making, and solution implementation) and their products. Also note the involvement of a system's stakeholders at all stages of the process.

In a further narrowing of scope, we are interested in specific instances of SDP that involve the use of computational simulation as a boundary object to facilitate the processes of *solution design* and *decision making*. We define a computational simulator as any set of variable and fixed parameters, relationships, and algorithms that allows one to model, configure, and evaluate hypothetical system states. In such cases, simulators are boundary objects between two major stakeholder groups: technical systems *engineers*

and non-technical *decision makers*. These largely mirror the bifurcated roles of "systems experts" and "decision making experts" proposed by Mieg [4]. In this case, engineers are primarily responsible for the development of relevantly parameterized and accurate system simulators, while decision makers must evaluate and choose between several candidate solutions generated with the assistance of such simulators.

The most basic example of computational simulation in SDP depends upon the technical skills of an engineer to act as both builder and operator of a simulator in order to generate candidate solutions for a decision making process (Figure 2). In this case, "non-interactive" means that it is not feasible for a non-technical decision maker to operate a simulator themselves. A key limitation of non-interactive simulation is that it can isolate a decision maker from the solution design process, which can make it less effective as a boundary object. Ideally, a decision maker is involved at all stages of SDP, including the solution design phase. Hypothetically, simulations as boundary objects can be improved by allowing decision makers to interact with them directly (Figure 3).

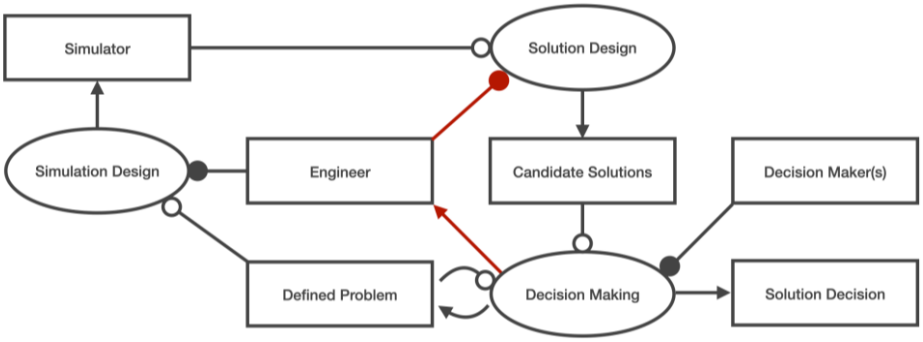

Figure 2. These are the components of a Systems Decision Process (SDP) adjusted to incorporate non-interactive computational simulation. In non-interactive simulation, decision makers are not directly part of the solution design process. Instead, they must work through engineers to generate solution candidates. As this process involves multiple steps, it is inefficient at best, or vulnerable to miscommunication at worst. Note that the SDP process of "solution implementation" is out of scope and therefore excluded.

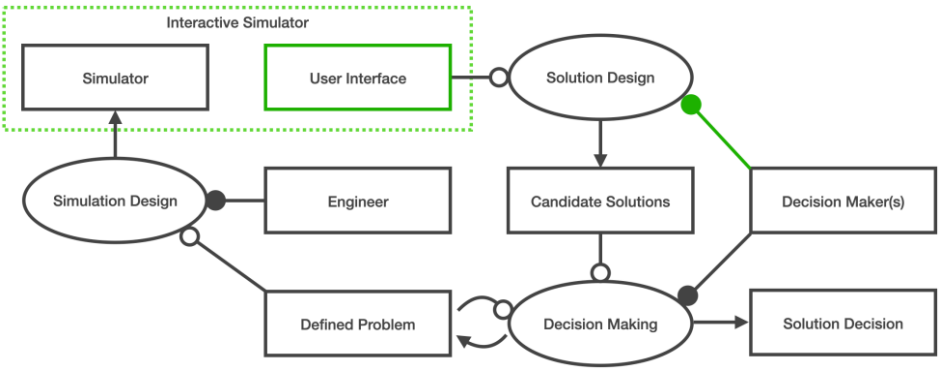

Figure 3. These are the components of a Systems Decision Process (SDP) adjusted to incorporate inclusive interactive simulation as the boundary object between engineer and decision maker. By adding a friendly user interface to the simulator, the engineer is no longer required to mediate the process of solution design. In this model, processes of solution design and decision making are now directly accessible to decision makers.

The fundamental goal of interactive simulation is to make computational simulators inclusive for non-technical stakeholders (i.e., decision makers) via friendly user interfaces (Figure 4). However, an interactive simulation may be challenging for an engineer to implement, since it can require concurrent skills in user interface and experience design, not to mention the requisite domain expertise for the system at hand. Due to this barrier, engineers in our laboratory often make do with non-interactive simulations, even if they would prefer to deploy their work as an interactive simulation. While we do observe some bespoke instances of interactive simulation [5,6,7,8,9,10,11], or large suites of software dedicated to particular domains or simulation types [12,13,14,16,17], we generally witness those engineers building smaller, specialized simulators struggle at user interface implementation.

Figure 4. A group of people role play as decision makers use a bespoke interactive simulation to configure a hypothetical campus during an academic workshop [10].

In recent prior work, we identified the potential of a generically designed user interface that might significantly reduce the burden to engineers who wish to develop interactive simulations [1]. We speculated that if an engineer configured their simulator to communicate with a generic user interface, the user interface could automatically adapt to reflect the input and output states of their simulator (Figure 5). By sampling a collection of existing, bespoke interactive simulations over a range of domains (e.g., maritime shipping, city planning, manufacturing, and transportation), we identified key features (i.e., "user stories") that a generic user interface should incorporate to be broadly relevant to engineers and decision makers. For example, these features included requirements for a series of generic inputs and outputs, visualization of model state, saving and recalling of solutions, an application programming interface, and easy distribution to end users.

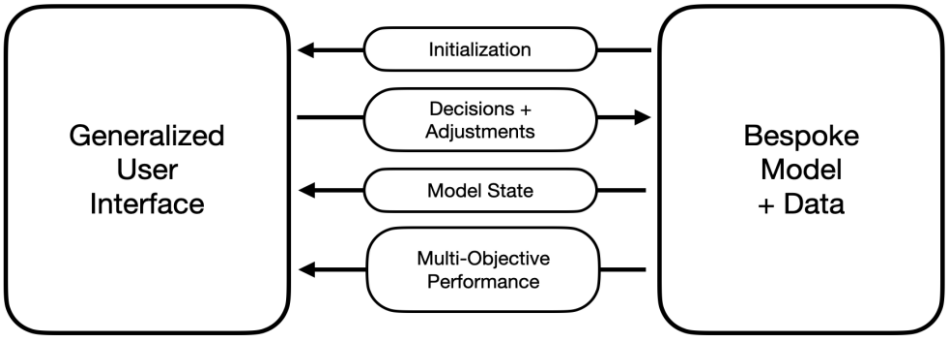

Figure 5. In prior work, we specified the high-level communication protocol between a simulator and a generic user interface. Note that "bespoke model" is synonymous with "simulator" in this context [1].

1. Objective

Our core objective is to implement and demonstrate the feasibility of a completely generic, open-source simulation user interface (OpenSUI) that meets the features established in our prior specification. For the purpose of demonstration, the interface must also be connected to an existing system simulator. The variable parameters of the user interface shall be completely reflective of the simulator. No content specific to a simulator shall be hard coded into the interface's logic, as all such information should be communicated to the interface via the simulator, on demand. A user with no prior experience using the interface must be able to perform unassisted interactive simulation over a structured period of time without significant issue.

During this process, we will also edit and refine features of the generic simulation user interface. These edits will be informed by feedback from preliminary users. We will also discuss the intentions and strategy for further experimentation, at scale, to test the efficacy of the interface in the context of Systems Decisions Processes.

2. Method

While our prior work defines the high-level feature requirements for an open-source simulation user interface, this same prior work did not define the specifics of a technical implementation. Therefore, technical choices such as software architecture and platform were made based on how well they enabled the pursuit of certain core feature requirements (Table 1). For instance, in the category of *deployability*, a non-technical decision maker must be able to easily access the software with minimal effort. Therefore, our implementation is a browser-based web application that requires little or no set up for a typical personal computer. Our implementation assumes that global circumstances (e.g., Covid-19) have rendered in-person workshops infeasible for the foreseeable future, as well, further compounding the need for a user interface that is easy to distribute over the web.

Table 1. These are selected core features (e.g., "user stories") to implement as defined by prior work [1]. Note that certain features, not yet deemed necessary for minimum viability, are unlisted.

Feature Category	User Story
Logical and Quantitative Decision Making	A **decision maker** can configure one or more decisions (i.e., variable parameters) related to the specification of a solution using intuitive means such as toggles, radio buttons, and sliders. A set of decisions constitutes a solution.
	An **engineer** can specify the scope and nature of decisions (i.e., variable parameters) available in the user interface.
	An **engineer** 's simulation can receive decisions (i.e., variable parameters) from the user interface formatted as a series of booleans, integers, or floating-point numbers.
Geographic Decision Making	A **decision maker** can specify geo-located points and polygons via a map or abstract surface.
	An **engineer** can specify the extent and nature of geo-located decisions available in the user interface.
	An **engineer** 's simulation can receive geo-located inputs as a series of coordinates from the interface.
Model State Diagram	A **decision maker** can view an abstraction of a model state as a 2D or 3D diagram of its state.
	An **engineer** can specify the content of model's state, which is then automatically rendered as a 2D or 3D diagram by the interface.
Save, Recall, and Viewing of Solution Iterations	A **decision maker** may save, preview, and recall any number of solution configurations from memory. Multiple solutions are automatically organized as a tree of iterations that reveal the evolution of a user's decisions over time.
Deployability	A **decision maker** can easily access a pre-configured interface (e.g., via secure web application)
	An **engineer** can easily generate and share their computational simulation to decision makers as an interactive simulation.
Adoptability	An **engineer** can easily understand the application programming interface (API) for the generic simulation user interface

2.1 FuzzyIO

We chose an actively utilized real estate simulator called "FuzzyIO" to be the initial engine with which the interface will be coupled. FuzzyIO is a Java library that converts several variable parameters, such as polygons and floor heights, into fuzzy masses for estimating total built area of a real estate development (Figure 6). It is currently used for research and education in real estate, but not in practice [10]. We believe FuzzyIO is a good initial use case because (1) it has a wide range of known variable and fixed parameters, (2) it has an open-source code base that we have permission to modify, and (3) it would be a great demonstration of OpenSUI's potential to make simulators more inclusive. Though we will make no change to the underlying logic of FuzzyIO, we do suspect that some modifications will be made to allow for hypertext transfer protocol (HTTP) communication. By checking that the interface successfully links with a relevant,

working simulation, it grounds our initial demonstration to reality and provides a clear finish line for development.

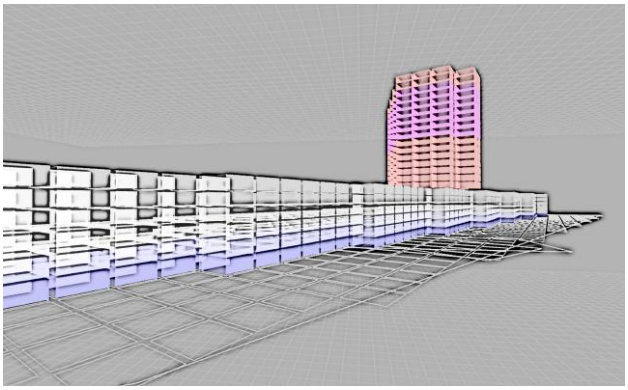

Figure 6. This visualization was produced by a technically capable engineer, manually configuring FuzzyIO's variable parameters without the aid of a user interface. It is unlikely that a non-technical decision maker could use FuzzyIO libraries in this way.

2.2 Software Architecture

The components necessary to make this system work are presented in Figure 7. Since our goal is for the user interface to work readily with simulators from any domain, not only FuzzyIO, it is essential that all of the logic of the interface is completely generic. This means that any domain-specific content or algorithms must remain solely with the simulator, not the interface. This is accomplished by creating a standard data schema for communicating simulation inputs and outputs via HTTP. Whenever a user begins a session, the interface makes a request to a specified simulation server. The server responds by sending an initialization file, which is essentially a template for a default set of simulation configurations. The user is then able to simulate solutions by modifying and submitting the default configuration template.

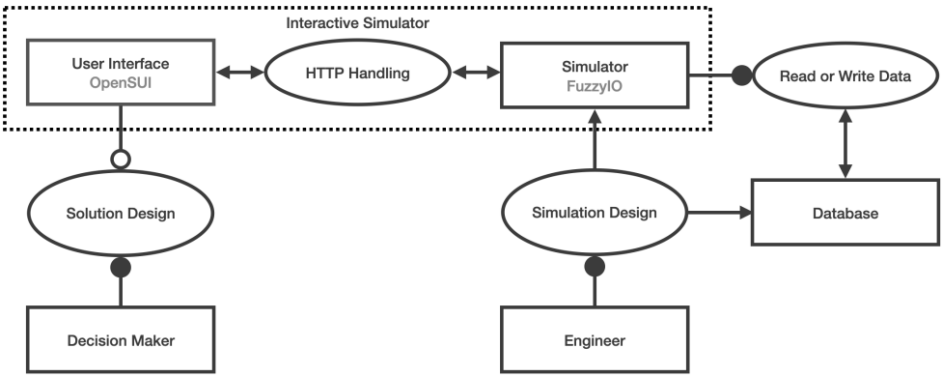

Figure 7. The software architecture for an interactive simulator using OpenSUI involves communication with a separate simulation server that adheres to OpenSUI's generalized communication protocol. Note that solution candidates may be saved or loaded via the OpenSUI interface, but it is the responsibility of the simulator to handle the persistent storage and retrieval of such scenarios and their associated users.

2.3 User Feedback Session

Upon completion, a small number of non-technical users without prior exposure to the tool are asked to complete a 30-minute structured exercise using the interface to generate multiple candidate solutions for a hypothetical real estate development in Boston. Some users are proficient in real estate, while others are not. At the end of the exercise, users are asked to select only one of their candidate solutions for execution. They are also asked to qualitatively express confidence in their decision.

3. Results

The idea of the Open Simulation User Interface (OpenSUI) was successfully reduced to practice as a browser-based webtool accessible over the internet. Furthermore, FuzzyIO was successfully adapted to serve HTTP requests and responses formatted according the OpenSUI's standard communication protocol. The entire system is deployed using low-cost web infrastructure services provided by Github and Amazon. When a user enters OpenSUI for the first time, they are prompted to enter the web address for a separate simulation server. This is because OpenSUI itself contains no specific content; it must work in tandem with a simulator to know the extent and bounds of variable parameters that the simulation designer has made accessible to the user.

Several non-technical users were able to view and generate solution candidates for a hypothetical real estate development in Boston (Figure 8). Users were also allowed to view and edit pre-loaded candidate solutions configured in advance. Though not yet conclusive, feedback suggests that users with an existing background in real estate may be more likely to select a self-generated solution over the pre-loaded solutions.

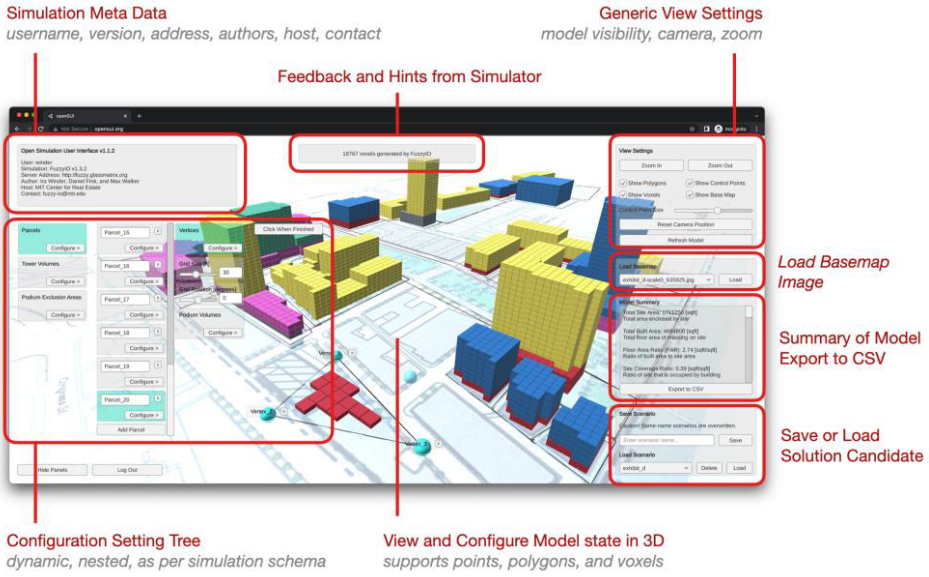

Figure 8. Screenshot of OpenSUI being used to view and configure FuzzyIO simulation in browser.

4. Discussion and Future Work

Building a truly generic simulation user interface was a challenging technical endeavor. It is no wonder that most interactive simulations are bespoke, given how difficult it is to identify and codify generalized rules for simulation systems. For example, FuzzyIO's variable parameters are structured yet unbounded. While it is relatively easy to codify individual settings (such as a slider to represent the number of floors in a building, or a dropdown menu to represent land use), the amount and variation of these settings in aggregate was potentially infinite. For instance, a user could make as many buildings as they wanted, all with their own unique settings. We believe we found an elegant solution to this problem by structuring configuration settings into groups, some of which are extendable and some of which are not. This allows a simulator to specify a default configuration tree that the interface can understand to be potentially infinite, even if it is not practically infinite.

There are still a few more features we hope to implement as time goes on. For instance, introducing more sophisticated UI for multi-objective decision making (MODM) is a priority, as specified in our prior work [1].

While we are indeed satisfied with the technical feat of building a tool like OpenSUI, the true impact of this work will be in the future research it enables. For instance, the preliminary feedback sessions led to important questions related to the effect of making pre-loaded solution candidates available to decision makers. Pre-loaded solutions may help kickstart solution design and improve quality of candidates. On the other hand, pre-loaded solutions could arguably have the opposite affect by reducing non-technical stakeholders' confidence in their own designs. We simply must do more tests at scale to find out.

Now that this work has reached an important milestone, we are confident that we can increase the quality and pace of workshops to gather data to test such hypotheses. This is because we specifically designed OpenSUI to run continuously via a web browser, allowing asynchronous usage with no practical limit to the number of users it can handle. While we suspect the primary indicators of any study will relate to the quantity and quality of candidate solutions for any given use session, OpenSUI also allows us to consensually gather a rich set of digital footprints associated with moment-by-moment user activity.

5. Conclusion

Our work suggests that implementing a completely generic, open-source simulation user interface (OpenSUI) is demonstrably feasible. We hope this work is an important step toward generic, inclusive interactive simulation techniques for transdisciplinary engineers. Though there is much work to be done to demonstrate its efficacy through workshops and experiments, this work sets the foundation for deploying requisite experiments at scale. In the meantime, we encourage any transdisciplinary engineers to reach out to us if they would like to try using OpenSUI with their own simulator.

Acknowledgement

The author of this paper is supported by a generous grant from Mercari Japan via University of Tokyo's Research Institute for an Inclusive Society through Engineering (R.I.I.S.E.), which is directed by Professor Yoshihiro Kawahara. The simulation software used in this work, FuzzyIO, was provided by the MIT Department of Urban Studies and Planning. FuzzyIO was cowritten by Ira Winder and Daniel Fink. The logic and principles of FuzzyIO and OpenSUI are partly inspired by conversations with Dr. Bryan Moser and associated members of The Global Teamwork Lab at MIT.

References

[1] I. Winder and K. Hiekata, User Interface Design for Multi-Objective Decision Making, *Advances in Transdisciplinary Engineering*, 2021, Vol. 16, pp. 566–573.
[2] G.S. Parnell, P.J. Driscoll, and D.L. Henderson (eds.), *Decision making in systems engineering and management*, John Wiley & Sons, Hoboken, 2011.
[3] Wohlrab, Rebekka, et al., Boundary objects and their use in agile systems engineering, *Journal of Software: Evolution and Process*, 2019, 31.5, e2166.
[4] H.A. Mieg, System experts and decision making experts in transdisciplinary projects, International Journal of Sustainability in Higher Education, 2006, 7(3), pp. 341–351.
[5] L. Pelegrin, B. Moser, S. Wanaka, M.-A. Chavy-Macdonald and I. Winder, Field Guide for Interpreting Engineering Team Behavior with Sensor Data. In E. Bonjour et al. (eds.) *Complex Systems Design & Management*, Springer International Publishing, Cham, 2019, pp. 203–218.
[6] I. Winder, D. Delaporte, S. Wanaka and K. Hiekata, Sensing Teamwork during Multi-objective Optimization, *IEEE World Forum on Internet of Things*, WF-IoT 2020, pp. 1–6.
[7] C.M. Rose, E. Saratsis, S. Aldawood, T. Dogan and C. Reinhart, A Tangible Interface for Collaborative Urban Design for Energy Efficiency, Daylighting, and Walkability, *Proceedings of the 14th Conference of International Building Performance Simulation Association*, 2015, https://dspace.mit.edu/handle/1721.1/106597, accessed June 20, 2022.
[8] J.I. Winder and K. Larson, Bits and Bricks Tangible Interactive Matrix for Real-time Computation and 3D Projection Mapping, *Future Technologies Conference (FTC) 2017*, 2017, pp. 1113-1116.
[9] I. Winder, Lego Logistics: Tangible Interactive Matrix Meets Last Mile Logistics Simulation - Interactive Simulation. Massachusetts Institute of Technology, 2016, https://ira.mit.edu/blog/lego-logistics.
[10] P. Manandhar, K. Rong, K. Carroll, R. De Filippi, I. Winder, J. Dieffenbach and B.R. Moser, Sensing systemic awareness and performance of teams during model-based site design. *IEEE World Forum on Internet of Things, WF-IoT 2020*, DOI: 10.1109/WF-IoT48130.2020.9221406.
[11] JF Finn III, D. Vasquez, A. Starr, K. Kusina, K. Silvester, I. Winder, *Autonomous Vehicles and Their Impact on Real Estate. Gensler Research Institute*, 2018, https://www.gensler.com/gri/autonomous-vehicles-and-their-impact-on-real-estate, accessed June 20, 2022.
[12] H.R. Weistroffer, C.H. Smith and S.C. Narula, Multiple criteria decision support software, *International Series in Operations Research and Management Science*, 2005, Vol. 78, pp. 989–1018.
[13] G. Montibeller, H. Gummer and D. Tumidei, Combining scenario planning and multi-criteria decision analysis in practice, *Journal of Multi-Criteria Decision Analysis*, 2006, 14(1–3), pp. 5–20.
[14] S.B. Eom, S.M. Lee, E.B. Kim and C. Somarajan, A survey of decision support system applications (1988±1994), *Journal of the Operational Research Society*, 1998, Vol. 49, Issue 2, pp. 109-120.
[15] S. Eom and E. Kim, A survey of decision support system applications (1995-2001), *Journal of the Operational Research Society*, 2006, Vol. 57(11), pp. 1264–1278.
[16] Razmak, J., & Aouni, B. Decision Support System and Multi-Criteria Decision Aid: A State of the Art and Perspectives, *Journal of Multi-Criteria Decision Analysis,* 2015, Vol. 22(1–2), pp. 101–117.
[17] P. Fritzson, A. Pop, K. Abdelhak, A. Ashgar, B. Bachmann, W. Braun, D. Bouskela, R. Braun, L. Buffoni, F. Casella, R. Castro, R. Franke, D. Fritzson, M. Gebremedhin, A. Heuermann, B. Lie, A. Mengist, L. Mikelsons, K. Moudgalya, … P. Östlund, The OpenModelica Integrated Environment for Modeling, Simulation, and Model-Based Development. *Modeling, Identification and Control: A Norwegian Research Bulletin*, 2020, 41(4), pp. 241–295.