Exploring Medical Practitioners Abilities to Use Visual Programming to Code Scenarios for Virtual Simulations

Bjørn Arild Lunde, Joakim Karlsen
Department of Computer Science and Communication
Østfold University College
Halden, Norway
bjorn.a.lunde@hiof.no, joakim.karlsen@hiof.no

Abstract— **Virtual simulations provide a safe environment to practice medical skills and has become more common in the health sector. To maintain and update virtual simulations with state-of-the-art medical procedures require expert knowledge in programming and IT development. Significant resources could be saved if medical educators and students could update the virtual simulation with new scenarios themselves. Based on a qualitative study of end users solving visual programming tasks, we identify constraints and opportunities in achieving this. The main constraint was their inability to break down scenarios into smaller codable steps. The main opportunity was how their familiarity with some elements in the visual programming language increased their ability to write code.**

Keywords-end-user programming; virtual simulations medical training

I. INTRODUCTION

The healthcare sector is actively pursuing the development of technology to support training [1]. Several experiments indicate that serious games and virtual simulations are promising platforms to support practitioners in the field with learning activities [2]–[4]. Due to rapid advancements in health research, activities are not necessarily done in the same way as they were ten years ago. To keep up with these changes, training tools and learning material must be updated to keep the practitioners' skills up to date. In the case that the learning tools are complex entities, such as virtual simulations and serious games, it is not unusual to hire personnel, either internal or external, who can adapt the training tools to reflect new knowledge. Valuable resources can be saved by giving end users the ability to make these changes themselves, using end-user programming tools.

Having the right skillset to write code and update virtual simulations requires an understanding of programming. A motivation for this study is to lower this knowledge barrier by wrapping text-based programming in a graphical interface that is user-friendly for end users without prior knowledge of programming. Visual programming languages that try to achieve this already exist, with block-based and node-based approaches being the most popular. A familiar example of block-based programming is Scratch, which was created specifically for end users without programming experience [5]..

The purpose of this study is to explore whether end-user programming can provide educators and students in the healthcare sector the ability to code a sequence of events for their training scenarios without the help of external personnel. Visual programming can probably give end users this capability, by being adapted to their requirements. The research question is therefore as follows.

What do observation of health personnel challenged to do visual programming to adapt virtual simulations to their training needs, tell about the opportunities and constraints of providing end-user programming tools for this purpose?

The result of the study will be a consideration of what this means for further development of end user-tool in this context.

In Section II, we will summarize previous work on this topic. After this, in Section III, we will describe the methods for data collection and analysis. Then, we will present the results in Section IV. In Section V, the discussion will be presented. Lastly, conclusion and future work will be discussed in section VI.

II. BACKGROUND

A typical end user will be a domain expert in a field other than computer science, in our case educators and students in the healthcare sector. End users do not possess the knowledge or understanding required to create and maintain software [6]. Giving the end users the opportunity to customize software without the assistance of external resources is the general idea of end-user tools [7]. Fischer claims that end-user tools are necessary to not get stuck in old routines as a result of outdated software [8].

The main challenge when learning how to write code that computers understand, is the different ways humans and machines interpret signs. Tanaka-Ishii [9] illustrates this issue with the following code example:

```
int x = 15
```

In this example, the content (the number 15) is represented by three different signs. The first is quite obviously the number itself, 15. This value is then represented by x, which points to where the value is stored. Finally, we have int, which represents the data type of the value. The challenge for newcomers according to Tanaka-Ishii, is that it can be confusing how these three signs are interpreted. In end-user tools, this problem can partially be eliminated by relying on visual blocks or nodes, direct manipulation and degrees of domain-specificity.

Variants of node-based and block-based visual programming both scrap the traditional textual programming in favor of visual elements. A block-based approach offers blocks that are put together almost like a puzzle where only certain pieces fit together to prevent error in the code [10]. A node-based language will consist of nodes that often have ports for input and output that are connected by threads where the information flows from node to node through these threads [11]. A strength of some block-based programming languages is that it's quite clear which parts fit together, something which minimizes the possibility of making mistakes. This constraint is not as prevalent in a node-based approaches. Since the user decides how nodes are connected, however, a node-based solution can be seen as more flexible. Both approaches to visual programming relies on ″direct manipulation" which provides visual elements that end users can point and click on [12]. Three characteristics define direct manipulation:

- Continuous visual representation of objects.
- All actions involve pressing buttons instead of using syntax.
- Operations must be possible to reverse quickly and easily.

Further, a visual programming tool can implement a Domain Specific Language (DSL), using terminology and concepts used by the target group to create the visual programming language [13]. This provides an additional layer of familiarity to the language, easing the learning process by reducing technical terms and jargon.

III. Method

To be able to shed light on the opportunities and constraints in providing end-user programming tools to medical educators and students, two digital task sets were created giving the informants tasks of programming a virtual simulation of a simplified scenario using a block-based or node-based DSL. The DSLs were designed specifically for this study. The tasks and supporting information built on each other so the informants could become familiar with the concepts and see different use cases for each node or block without being overwhelmed.

The final task set consisted of 18 pages in total, of which 6 of these were tasks in different forms that the informants had to answer. Each informant only completed one of the task sets, to prevent the results being skewed based on the informant having more experience with the tasks at hand. Initially in the task set, the study and purpose were introduced so the informants could gain an understanding of what they were about to do. The main purpose of programming and the way humans and computers handle information differently was presented in this step. The informants were then introduced to a task requiring using blocks and nodes to handles text prompts to the players. This was done to provide an easy introduction on how to connect these together. In its simplest form, three different block and nodes were introduced through the task sets, but some variations of these appeared as they progressed.

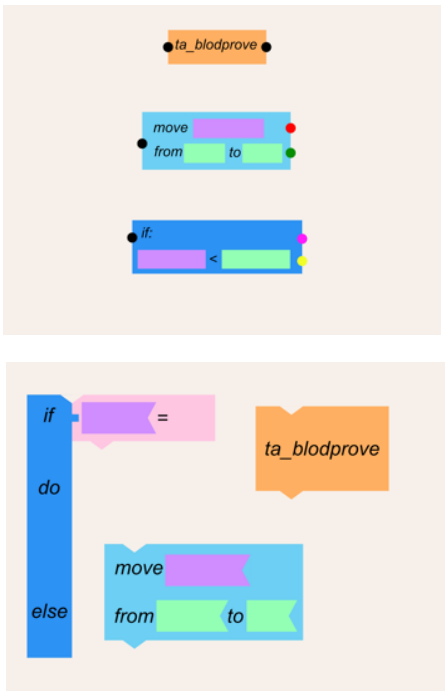
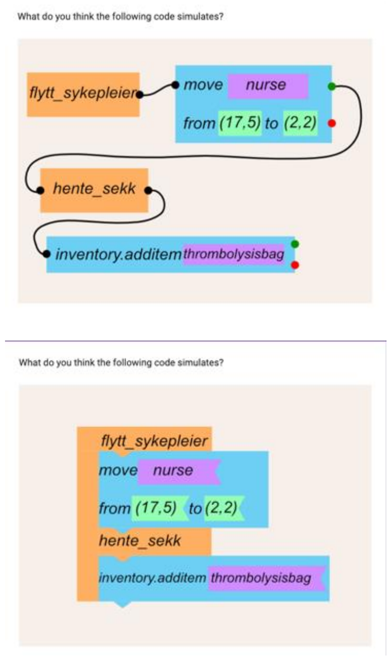


Figure 1. The different nodes (top) and blocks (bottom) in the task sets

Figure 1 illustrates how the different nodes and blocks look. The orange elements handle text prompts, which is displayed to the player going through the scenario. The light blue handle instructions for how the computer should carry out operations such as moving objects, handling time or similar uses. The dark blue represents if / else statements. Inside the nodes and blocks themselves, there are colored fields which have different functionality. The purple fields allow the users to point to objects that exists in the scene they are working on, this includes characters, medicines, devices and more. The green field allow the user to enter manual values such as coordinates, blood levels and more. All of these are introduced and demonstrated both separately and combined with each other through the task set.

The next two tasks were multiple-choice tasks that presented the informants with pre-written code, where the task is to choose the option that the code expresses. These tasks had two intentions: evaluate the understanding of the informants' abilities to read code and to present them with pre-written code so that they could become more familiar with how the blocks or nodes should be connected. From this point, more components were gradually added to the code while removing the aids more and more. In the fourth task, the informants were presented with code and they had to write the meaning of the code, as illustrated in Figure 2.


What do you think the following code simulates?

What do you think the following code simulates?

Figure 2. Task 4 in the node-based (top) and block-based (bottom) task set.

The fifth task introduced if / else statements and the informants were presented with a task where they should describe whether they should provide the patients with insulin based on blood sugar values. The informants did not get any alternatives to rely on and were asked to write how they understood the code in their own words.

In the final and heaviest task, the informants were asked to draw code themselves to simulate a sequence of events as given in the medical scenario illustrated in Figure 3. The task was solvable by using the tools they had learned previously.

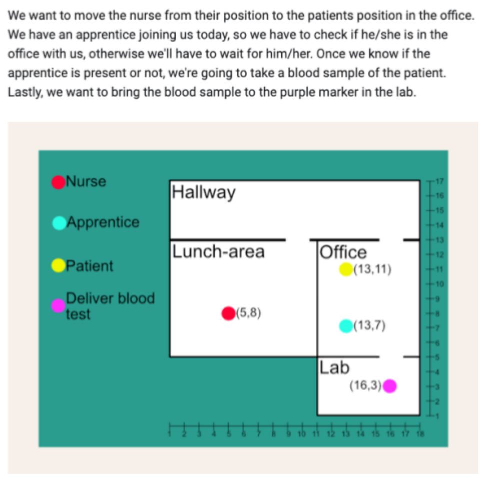

We want to move the nurse from their position to the patients position in the office. We have an apprentice joining us today, so we have to check if he/she is in the office with us, otherwise we'll have to wait for him/her. Once we know if the apprentice is present or not, we're going to take a blood sample of the patient. Lastly, we want to bring the blood sample to the purple marker in the lab.

Figure 3. Task 6 in both task sets

A. Data collection and analysis

The medical students and educators recruited to the study, were affiliated with a small university college in Norway, having a state-of-the-art simulation center used to educate health personnel. The data was collected both digitally and physically to secure participation from both students and staff at the university. The digital data collection lasted from May 15th to September 15th, 2021. There was a total of 9 informants completing the digital task sets, both male and female nursing students and medical educators ranging from 20-35 years of age. Out of these 9 informants, 5 completed the node-based task set while 4 completed the block-based task set. The physical data collection was completed in weeks 43 and 44 in 2021. There was a total of 5 informants completing the task set, all of them being female medical educators ranging from 35-60 years of age. Out of these 5 informants, 2 completed the node-based task set while 3 completed the block-based task set. All the informants are listed in Table I with the associated task set they completed as well as which data collection they participated in.

TABLE I. INFORMANTS

Participant	Task set	Data Collection
#1-1	Node-based	Digital
#1-2	Node-based	Digital
#1-3	Node-based	Digital
#1-4	Node-based	Digital
#1-5	Node-based	Digital
#2-1	Block-based	Digital
#2-2	Block-based	Digital
#2-3	Block-based	Digital
#2-4	Block-based	Digital
#3-1	Node-based	Physical
#3-2	Node-based	Physical
#4-1	Block-based	Physical
#4-2	Block-based	Physical
#4-3	Block-based	Physical

As the digital collection had to be anonymous in line with the approved application submitted to the Norwegian Center for Research Data (NSD), recruitment had to happen through group pages on social media and neutral third parties reaching out to informants. The subsequent data analysis followed the model proposed by Creswell & Creswell [14], which consists of five steps; 1) sort and prepare data for analysis, 2) create a general understanding of the data, collect and sort thoughts and feelings from the informants, 3) code and categorize data, 4) describe factors such as places, people and sequences of events in the data, 5) consider different perspectives and quotes, and compare them to each other, present data in a narrative giving expected findings, surprising findings and unusual or conceptual findings.

IV. RESULTS

The multiple-choice tasks, tasks 2 and 3, in both task sets were answered correctly by all informants, both in the digital and physical sessions. These were not the most complex tasks, but provided an indication that the programming languages were both readable and understandable. In the physical sessions, a few informants were on the wrong track on task 3 before they ended up with the right answer. The correct answer to task 3 is option “C”, but there were two informants who quickly chose alternative “B”. These two options are quite similar, where option “C” suggests the code simulates moving the bed from one position to another, while option “B” suggests moving the patient. As soon as the context was removed and they didn’t have a visual representation to support them, the informants quickly seemed uncertain. This issue is illustrated in the response from participant #4-1.

> So I’m going to move the bed? Then it is alternative number 2 (B). Move patient from position 10.4 to bed at position… But it is… Yes… or wait…. #4-1

Author: Why do you think that?

No, now I’m thinking… I want to… We will move the patient… move… moving the bed is impossible because it’s nothing there. There is no code. But we are going from 10.4 to 7.8. Then it must be: Move bed from position 10.4 and to the bed at position 7.8? But it may still be that… This one was a bit more difficult. This is to check if it is easy to use or not, I must think about that. #4-1

Author: What if you start at the top and work your way down?

Okay. I’m going to move the bed. From 10.4 to 7.8. Then it must be alternative “C”? #4-1

From task 4 no alternatives were offered, and they had to write their answers without support. This resulted in a major drop in the quality of the results. The correct sequence of events simulates a nurse that moves from where he / she is to where the thrombolysis bag is and retrieves it. The informants took freedom in the way this was interpreted, evident in the following examples from informants #1-2 and #2-1.

The nurse should go from for example the patient room to the rinsing room and pick up thrombolysis bag #1-2

Nurse picks up thrombolysis bag #2-1

On the same task there were some misunderstandings regarding how the code worked. In the example below, informant #2-2 interprets the code as the interface itself, and that the blocks are buttons to press.

If you press the orange block, we will move the nurse from position 17.5 to 2.2. Then press the lower orange block, and bring along the bag on your way. #2-2

Something that is pervasive in these examples is that it is troublesome for the informants to distinguish what is information to the player from the information to the computer. Another observation based on the block-based responses is that it was challenging for the informants to understand which order to read the code.

In the fifth task, that dealt with if / else statements, the general understanding seemed good. The informants all understood that a condition decide the outcome. There were some differences in the way informants read the “greater than” or “lesser than” symbols, however, as seen in the examples below.

If BS is over 17mmol/l, you should be given insulin, if below, do not give insulin. #1-1

If the blood sugar value is less than 17mmol/l then give insulin. If not, then do nothing. #2-1

In one of the physical sessions, informant #4-2 seemed to misunderstand the concept of “greater than” and “lesser than” and thought that if the value was anything else than what was being checked, 17mmol/l in this case, that the else condition would be triggered.

It simulates a blood sugar measurement, that a nurse should manage blood sugar. And if the value is 17mmol/l you should give insulin, if not, then do nothing. #4-2

In task 6, which is the last and most complex task, more effort was required, and the quality of the answers varied accordingly. Overall, Figure 4 illustrates what could be expected. Informant #1-1 took advantage of a variety of nodes with mostly correct use of color codes and proper connections.

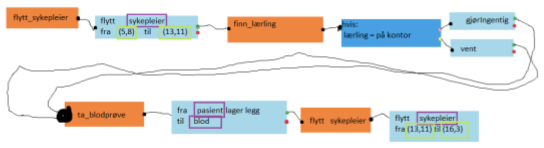


Figure 4. Response from participant #1-1 on task 6

One of the strengths of block-based programming is that you can only connect blocks if they fit together. This seemed to be forgotten or ignored in several of the responses, and the informants took freedoms that would not be possible. The blocks are in some cases stacked on top of each other without regards to these rules as displayed in Figure 5.

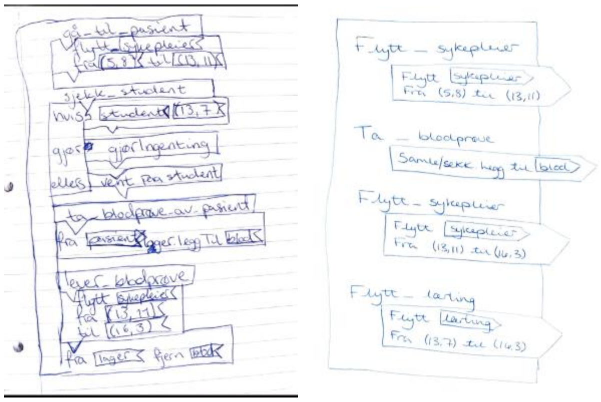


Figure 5. Responses from participants #2-2 and #2-3 on task 6

With multiple hand-drawn responses on task 6 from the digital sessions, it was more appropriate to conduct a discussion with the informants in the physical sessions. This way, the informants could put words to their thought process when solving the tasks. The following quotes from informant #3-1 seem to indicate a pretty good overall understanding.

> I think the first one is perhaps an orange one, considering the task is to move the nurse from one place to another. Behind the orange node there is more code, and that is the purple for the nurse who is going from the office to the lunch area. #3-1

As in all examples presented in the task set, the informant starts with an orange node to prompt the user with information. When asked to check if the apprentice was present, using an if / else statement were quickly suggested. This was also the intended way to solve this part of the problem.

> We need to check if the apprentice is in, then we use that if and else node again, so you do it if he is there, otherwise, we do nothing. I think I get that one correctly. #3-1

While not completely sure on how to connect the orange nodes for the text prompts, the informant was aware of their existence. This was also the only case where breaking down the scenario into tasks was addressed.

> Only I might be a little unsure of how many activities and these orange nodes I should put between the actions. I split up the scenario into tasks, so I thought the first thing about moving the nurse is a task. #3-1

In the responses to the block-based task set in the physical sessions, there were a few more challenges as seen in the response from participant #4-2 below. In the same way as the digital task set, several code components such as text prompts are forgotten. In addition to this, color codes are not commented on at all.

> To me it looks like we are just passing by and going from one place to another, and then I look to see if someone is there. But how the interaction takes place, how to ensure that the nurse brings the apprentice and how they take the blood test, I do not know. #4-2

An interesting observation is the way the informant attacked the problem. Instead of dividing the scenario into smaller parts like in the previous response on the node-based task set, she tried to solve the entire scenario at once.

> I move the nurse to the office, and then I check if the apprentice is there, then I should be able to move on? So, I take a blood test of the patient? #4-2

As soon as all the supporting materials were removed, the informants quickly felt overwhelmed and somewhat insecure about the order to do things. As seen in the quotes above, the informants solving the node-based task set included the concepts of the language itself in explaining how they were trying to solve the problem at hand. The informants solving the block-based task set did this to a lesser degree.

A. Recurring themes

While not required to complete the tasks, the informants had the opportunity to display information to the players using the orange nodes or blocks. This, however, seemed to be ignored for the most part. An interesting observation in this regard occurred in one of the physical interviews conducted for the node-based task set, where one of the informants tried to improve the pre-written code by posting questions to the players using the orange nodes.

Several of the informants in the physical sessions had previous experience with real-life simulations at the college in which they as educators observe students and provide them with instructions using a communication system. In these simulations, the students go through different scenarios, and in one of the interviews on the node-based task set, comparisons were drawn between the programming task and these physical simulations. The informant explained that the actions and the way they gave instructions to the students were quite similar, and that the flow of the code running from node to node was kind of the same as reading the instructions in the simulations.

Another comparison occurring in the responses from the informants solving the node-based task set, was that it was reasonably easy to follow the flow of the code as it looked somewhat similar to flow charts. No such comparisons to previous experiences were mentioned in any of the responses for the block-based task set.

While comparing the responses from the last tasks in both digital task sets, it is immediately apparent that the answers in the node-based task set follow the rules as intended when compared to the block-based responses. The nodes are to a greater extent connected properly, and there is more active use of color codes and text prompts to the players, although this is in several cases forgotten here as well.

V. Discussion

Based on the analysis of how the informants solved the task sets, we identify one major constraint and one major opportunity for creating end user programming tools in this case.

The main constraint revolves around the inability to break down scenarios into smaller steps. Instead of looking at the individual steps in the scenarios and which elements were needed to represent them, some looked at the problem as a whole, and tried to code multiple or all the parts of the scenario at the same time. The ability to adapt complex problems to lesser, solvable problems through reduction, algorithmic thinking or other means are referred to in the literature as computational thinking [15]. Increasing the end user's familiarity with this kind of problem-solving may decrease the significance of this constraint over time. In addition to this, the informants wanted to express themselves more freely than the languages allowed them to, and several wrote code that would be syntactically impossible (for a machine to understand).

The main opportunity was the familiarity end users have for certain visual tools and procedures. Even though flow charts were not on the agenda to be explored in the study, it turned out to be a form of visualization healthcare professionals recognize and understand. Further, to rely on elements from their practice, or to make the language domain specific, seemed to work well. The if / else task supported this, as they were already familiar with how the measurement of blood sugar impacts what action should be taken. Based on this knowledge they understood that based on a condition, being the blood sugar values in this case, one of the listed actions should be taken.

We conclude that combining domain specificity in the language, using familiar visual elements such as flow charts and adopting the concepts of direct manipulation, are the three main aspects that could help health educators and students in coding sequences of events in virtual training scenarios.

VI. Conclusion and future work

After investigating different approaches to visual programming, enough data has been collected to answer our research question. The data indicates that the idea of letting medical educators and nursing students manipulate and maintain their own training tools using visual programming is one worth pursuing. As discussed earlier in Section V, there are however several prerequisites and aspects of both the languages themselves and the interfaces to them that needs to be further experimented with to make tools like these accessible to the end users.

The next step will be to develop a new and more in-depth visual programming interface using node-based programming, direct manipulation and incorporating elements from flow charts, as a baseline. While this study indicates that medical educators and students can express a sequence of events in scenarios using code, this is only a step on the way towards developing a fully working end-user tool to create, maintain and adapt virtual simulations for health care education. A telling example of the complexity involved, is the need to code meaningful interactions with the students taking part in the simulation, a challenge touched upon in this study by exploring the use of visual elements to express this (orange nodes or blocks). The ultimate goal of future work is to support further development of virtual simulations for training purposes in healthcare education, by giving the end users the means to make these without the help of trained IT professionals.

References

[1] N. Sharifzadeh *et al.*, "Health Education Serious Games Targeting Health Care Providers, Patients, and Public Health Users: Scoping Review," *JMIR Serious Games*, vol. 8, pp. 1-16, Mar. 2020, doi: 10.2196/13459.

[2] C. Foronda, B. MacWilliams, and E. McArthur, "Interprofessional communication in healthcare: An integrative review," *Nurse Educ. Pract.*, vol. 19, pp. 36–40, Jul. 2016, doi: 10.1016/j.nepr.2016.04.005.

[3] D. King *et al*., "Virtual health education: Scaling practice to transform student learning," *Nurse Educ. Today*, vol. 71, pp. 7–9, Dec. 2018, doi: 10.1016/j.nedt.2018.08.002.

[4] W. Westera, "How people learn while playing serious games: A computational modelling approach," *J. Comput. Sci.*, vol. 18, pp. 32–45, Jan. 2017, doi: 10.1016/j.jocs.2016.12.002.

[5] M. Resnick *et al.*, "Scratch: programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009, doi: 10.1145/1592761.1592779.

[6] M. F. Costabile, P. Mussio, L. Parasiliti Provenza, and A. Piccinno, "End users as unwitting software developers," in *Proceedings of the 4th international workshop on End-user software engineering - WEUSE '08*, Leipzig, Germany, 2008, pp. 6–10. doi: 10.1145/1370847.1370849.

[7] Z. Menestrina and A. De Angeli, "End-User Development for Serious Games," in *New Perspectives in End-User Development*, F. Paternò and V. Wulf, Eds. Cham: Springer International Publishing, 2017, pp. 359–383. doi: 10.1007/978-3-319-60291-2_14.

[8] G. Fischer, "End-User Development and Meta-design: Foundations for Cultures of Participation," in *End-User Development*, vol. 5435, V. Pipek, M. B. Rosson, B. de Ruyter, and V. Wulf, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 3–14. doi: 10.1007/978-3-642-00427-8_1.

[9] K. Tanaka-Ishii, *Semiotics of Programming*. Cambridge University Press, 2010.

[10] D. Weintrop and U. Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Trans. Comput. Educ.*, vol. 18, no. 1, p. 3:1-3:25, Oct. 2017, doi: 10.1145/3089799.

[11] D. Mason and K. Dave, "Block-based versus flow-based programming for naive programmers," in *2017 IEEE Blocks and Beyond Workshop (B B)*, Oct. 2017, pp. 25–28. doi: 10.1109/BLOCKS.2017.8120405.

[12] B. Shneiderman, "Direct manipulation for comprehensible, predictable and controllable user interfaces," in *Proceedings of the 2nd international conference on Intelligent user interfaces*, New York, NY, USA, Jan. 1997, pp. 33–39. doi: 10.1145/238218.238281.

[13] J. Sprinkle and G. Karsai, "A domain-specific visual language for domain model evolution," *J. Vis. Lang. Comput.*, vol. 15, no. 3, pp. 291–307, Jun. 2004, doi: 10.1016/j.jvlc.2004.01.006.

[14] J. W. Creswell and J. D. Creswell, *Research design : qualitative, quantitative & mixed methods approaches*, 5th edition. Los Angeles, California: Sage, 2018.

[15] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006.