Offline Situational Game for Teaching IT Systems

KOROM Szilárd, ILLÉS Zoltán

Abstract. There are still many offline methodologies and playable games out there for teaching various curriculum elements. These games help the students in understanding difficult IT concepts. This approach is called "Computer Science Unplugged" or *CSUnplugged* in short. As an example, think about sorting algorithms, binary numbers, data structures, or databases. In the following article, we would like to introduce a new situational game that focuses on software systems, and network communications by simulating multiple programmable devices, that communicate with each other in real-time. We believe that the game makes the students activities and helps them to understand critical and important terms in an interesting and playful way. The game is a simulation of a smart home system in a classroom environment and does not need any digital devices.

Keywords: CSUnplugged, without computers, smart home, informatics systems, network communication, offline game

1. Introduction

The essence of teaching computer science without computers is to give students a playful real-world experience of the important IT concepts and terms. The most important requirements against this methodology are that they should not need any digital devices and they should have as least prerequisites as possible. The goal is to help the students figure out an intuitive, playable solution for an IT problem.

On the main CS Unplugged webpage[1], there are many examples, translated into more than 20 languages. The "*CSUnplugged*" term was introduced in 1990 by Tim Bell, Mike Fellows, and Ian Witten in the first book about the topic [1]. Since then, hundreds of papers, and articles have been created for various students ages from primary [2] to high school [3] [4]. They cover a wide variety of topics within IT education. Sorting algorithms, for example, are not easy at all, but there are several very simple practices that can be played in the classroom [5]. For example, if we ask the students, they will likely be able to line up very quickly in terms of height. The actual algorithm may not be realized, but if they observe their own activity, the used methodology can be formalized. Most of the sorting algorithms can be played in reality, so the teacher is able to suggest and guide the students to line up in other ways.

Nowadays, these offline games have extensive literature. The concept can be applied to different subjects [6], but it is worth mentioning that not everything can be replaced, but certain parts of the curriculum can be substituted by these unplugged games. By the way, it has already happened with a few textbooks [7] [8] and systems that support coding learning, such as code.org [9]. The approach is not just popular but has been proven to be effective [10].

In this paper, we would like to introduce an offline situational game, that can be useful in the teaching of informatics systems. The topic is general, but the game is flexible, so the tone can be easily moved by the teacher based on a concrete curriculum.

[1] https://www.csunplugged.org/en/ (last viewed: 06/06/2022)

2. Description of the situational game

The situation is that we are in a classroom with the students, and we would like to build a smart home project, where the IT devices, controllers, sensors, transmission devices, gateways, and displays are all real students. For example, if we would like to measure the temperate at different points in the classroom, a student has to stand there "measuring" the temperature (actually guessing a number that is approximately realistic), and another student controls the lamp by standing next to the switch, another student monitoring the solution by using the whiteboard (writing the values on it). The aim of the game is to find the most efficient division and forms of communication as possible to operate the system. The key is how to modify the various "components" by expanding and complicating the system. As the task deepens and students invent new and new solutions, they engage new players as the game transforms into software architecture. Of course, the IT analogy is not (necessarily) formulated in the student's minds, but continuous reflection and teacher orientation can help in that. For example, the number of the needed programs or hardware can easily be defined by analyzing the roles (played by the students).

3. Technology background

First, we would like to define what kind of IT tools and knowledge are necessary to implement a solution for the problem above. The description is not exhaustive and is not detailed, as it is outside the scope of the article, but we would like to give a comprehensive picture.

3.1 Sensors and controllers

Sensors and controllers are low-level IT devices that can send or receive signals/data in the real world. Examples can be light, distance sensors or a simple LED, a switch, or even a motor. These are non-programmable target devices, that require a direct controller.

3.2 Local computing

These are the controllers of the previous layer, which are mainly suitable for M2M communication, i.e., they either provide some value while running continuously, or they act on a directed request (eg.: they turn on a lamp, send back data).

At this level, we can also talk about circuit solutions since the controller has to be connected to the peripherals somehow. Of course, "ready-to-use" devices are also available with GPIO (General Purpose Input and Output Connections). The most common of these, which can also be used for educational purposes, are the Ardunio and Raspberry Pi computers. The former can be used to create embedded systems, the latter to produce products very similar to the operation of a desktop computer.

3.3 Network and internet, data transmission

The most important question in this layer is what "communication" actually means. As for the IoT systems, the network protocols are the answers. Perhaps the most common are:

- MQTT or AMQP
- HTTP or REST

3.3.1 The MQTT and AMQP protocols

These protocols (or this type in more general) are very important from our paper's perspective because real-life communication is perhaps the most similar to this. The AMQP and MQTT are publish-subscribe-based protocols. The idea is that whoever forwards the message does not care who will receive the message. It is like having a discussion in a crowd. We say something very loud, and those who are listening will hear and understand the message, the others will ignore the report. The operating principle is that there is a central software (called a broker) that receives the published data and sends it to the entities who have subscribed to that particular type of message. This is very different from the request-response structure, as nobody requested the sent message, and the sender does not even know who will receive it (and whether the clients who wanted it actually received it). In general, this is very common in the multi-device world, including in smart home implementations. For example, a temperature sensor does not necessarily need to know who is curious about the data. The sensor transmits it to a central application (the smart home controller) from which users request data through a web interface or a mobile application.

3.4 IoT architectures in general

The general architecture of a real IoT solution is shown in Figure 1:

[11]

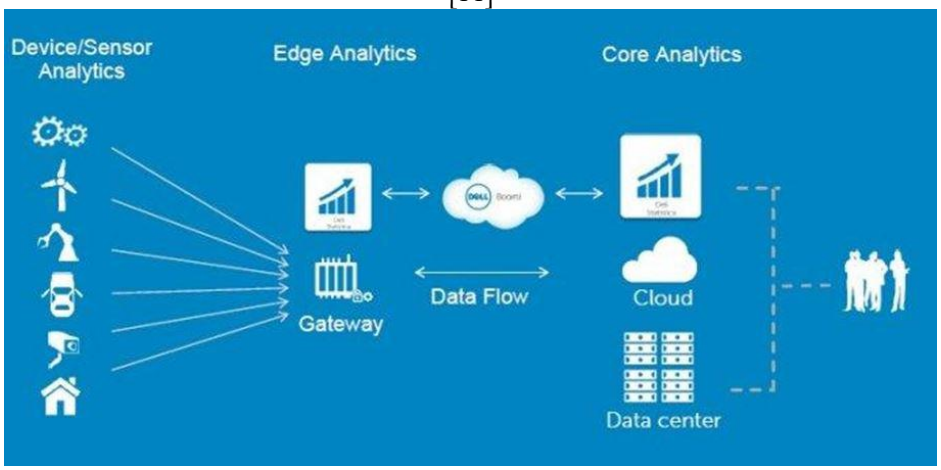

Figure 1: General concept of an IoT solution [11]

3.5 A concrete architecture

In the following, we would like to show what the architecture of our specific smart home solution might look like. We do not intend to fully detail them, as this is not the main focus of the article.

Of course, other solutions are conceivable. The examples are important because we actually want to play these with the students. And during the game, students are expected to design something like these, based on their own experiences.

3.5.1 Smart-home architecture without a server

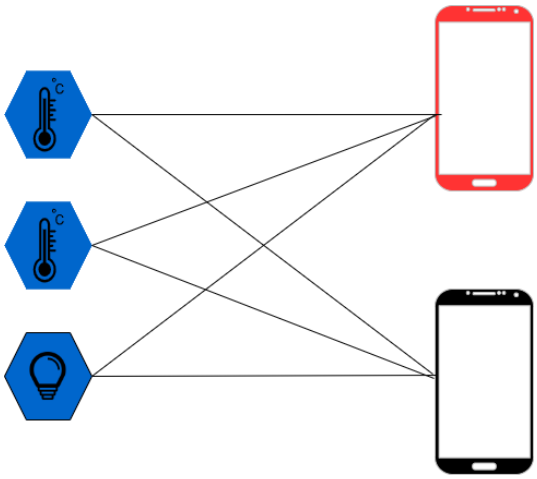

Figure 2: Serverless version of smart home architecture

In this case, we need to create a total of 3 applications:

- Temperature sensors are embedded devices, as they only measure and transmit data. A low-level program is running on them.
- Lighting is also an embedded device, as you only need to serve and handle a simple request. A low-level program is running on them.
- A mobile application that can map and communicate with devices available on the network. High-level programs are required. Graphical user interface needs (GUI) to be addressed here.

This is the most commonly used architecture in a normal user environment. The advantage is that the implementation is cheap and flexible, as it does not require the operation and maintenance of a server, and integration is easy. There are several tools on the market that are functional and usable just after downloading an application.

The biggest problem with the architecture is that the system is not integrated. It means we will need to download and use as many applications as many devices we have. Communication with devices is severely limited and we may run into various type of problems if we want to run many clients at the same time. If we are looking for a more complex solution, where the interoperability of devices and decision logic is necessary, the architecture is inapplicable (e.g.: we want an automatic temperature-based heating controller).

Another big disadvantage is that we do not have a central solution, that could care of the security issues, so all the individual components must handle their own security issues. One of the biggest problems in the IoT world is the lack of security handling [12].

3.5.3 Smart home architecture with a server

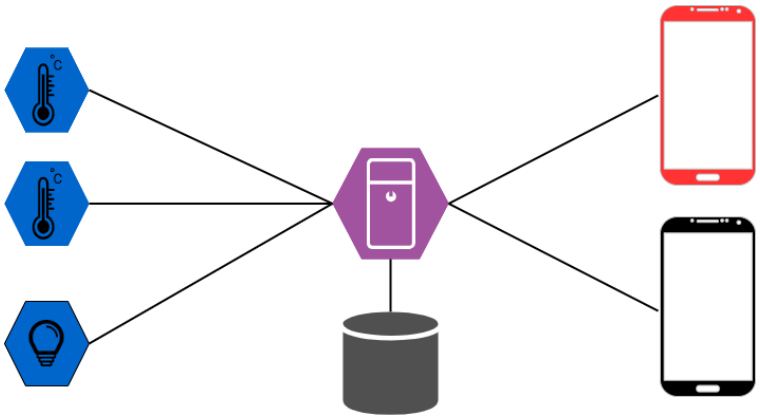

Figure 3: Server version of smart home architecture

In this case, we need to create 4 applications. The previous three as well plus a server. The server program is able to communicate with sensors and controllers, make decisions, serve and store data (for example, in a database), no appearance is required. The server program is by no means low-level, as it has to communicate across multiple threads and perform a variety of tasks.

The advantage is that with proper planning, it can be largely independent of who gets and gives data. The system is therefore scalable and easily expandable with new tools.

The system is more secure than the previous one since the sensors are completely hidden from the outside world, they are only communicating with the central program.

The downside is that it is both expensive and hard to maintain, as the tools are not independent of each other at all. Each component must be selected so that it is capable of being integrated with our own system. We also need a high available server computer.

4. Game description

The problem with playing the operation of IT systems is that a program, a real device, is very different from how we humans would actually act. For example, if we designate someone to measure the temperature at a certain point in a classroom, then write the result on the board, the student would probably solve this by guessing a number and then walking over to the whiteboard and writing it down. However, in the analogy of IT systems, there is a far simpler solution than creating a measurement software and an individual robot that can walk and draw. On the other hand, raising the issue is not useless, as it shows the difference between informatics and everyday thinking.

All in all, the main problem is that communication and perception are different from a machine's perspective. For example, we cannot instruct a student not to see or hear others but still be able to read particular messages.

The ultimate goal is to perform a system with students as actors that models the IT solution of a real smart home system.

4.1 Methodology

The planning must be done by the students and its complexity gradually deepens, layer by layer. The methodology is based on raising the problem and then guiding the students with questions. A possible set of questions is shown in Table 1.

Suggestions – guiding	Expected outcome	The game	IT analogy
What actors do we need if we want to measure the temperature in the 2 corners of the classroom, want to turn on the lamp, and want a user to be able to control this on the whiteboard as if the drawing were the controller application itself?	Someone is planning how to look at the application (draw it on the whiteboard). The thermometers are 1-1 students, someone is at the switch. There is a user who presses the drawn button on the board or yells if he/she needs data.	Everyone watches what the user on the whiteboard wants (presses a button, asks for something), and they respond to it individually.	Request-response protocols (e.g.: HTTP). There are several tools for different purposes. There is a network. GUI should also be designed. Peer-to-peer relationship.
The problem with the previous system is that the sensors/controllers can't actually see the button and can't actually talk.	A new student is assigned to control the application and forward the request to the controllers and draw when information is received.	When a user presses the button, only the controller student speaks and only with the selected "target device" and vice versa.	Server-client communication, but no database yet. Publish-subscribe-based communication.
We would like to monitor the temperature at real-time.	The sensor students keep saying out loud what they "measure".	The result is too chaotic, difficult to extract, and the drawing student cannot follow.	Transmission layer, gateway, queue. Continuous data transmission conforms to the MQTT / AMQP protocols.
Improve the system so that there is no noise or information clutter.	Data transfer students are assigned to transfer information between the sensors and the monitoring student.	The transmitters carry the information so there is no noise. If they get there at the same time, they have to line up.	Very strict architectural and communication rules for noise cancellation and efficient data transmission. Each component has a very well-defined task.
There should be more controllers and users (like for a real smart home project).	Students split the whiteboard in two and involve a new controller person and data transfer people.	Communication slows down because sensors and switches need to provide more data.	

Table 1: A possible course of the game

It is difficult to give specific guidance about the game because each of the suggestions depends on the solution the group gives to the previous one. For example, students may be given a fairly well-designed structure right from the start:

- There is a controller person
- There are data transmitters
- There is a drawing student
- Communication is disciplined and in accordance with the rules

Of course, even in this case, new components can be developed and added to the system. Another approach is to reverse the process to play a bad solution directly. Then the awareness and formalization of the problems can be the key

4.2 The tones

In its original form, the exercise focuses on network communication and software architectures. This can of course be modified. So, it's all about the questions the teacher asks, and the direction the teacher guides the students:

- **Algorithms**: what exactly is the "logic" (sensors, switches, controls, drawers) that each student must implement?
- **Hardware**: exactly what hardware would we need to actually do this? What are the requirements for the hardware, and what should they know?
- **Software system:** what software would we need to write to make this happen? How do we handle problems at the software level (data loss, corrupt data, backlogged requests)? What kind of implementations should be and used, what paradigms should be used (e.g.: console, web, mobile, windows)? Is there a need for an operating system on each hardware? If so, which one? If not, why not?
- **Data**: what data storage is needed? Who should do the storage and data management? What data packets should pass through the communication? What should these data packets look like?
- **Programming languages**: Which programming languages would be useful for the implementation? Which programming environments would be used to implement the different components? What programming paradigms could be used?

5. Real tests

5.1 Description of what happened

The game was tested with first-semester students of Eötvös Loránd University. The session was attended by 14 students and lasted 60 minutes. A questionnaire was also completed at the end of the session.

The game in this form is obviously not the most perfect, as the students already have a high school degree in computer science and are familiar with the concepts presented by the game. Nevertheless, the experience gained there was useful. Students were asked to participate both as "students" and as observing "teachers". Throughout the session, we constantly reflected on our own work and evaluated how useful the session was.

The game started with the presentation of the situation and the task. Without our intervention, the students roughly devised and implemented the following architecture (each circle corresponds to one person):

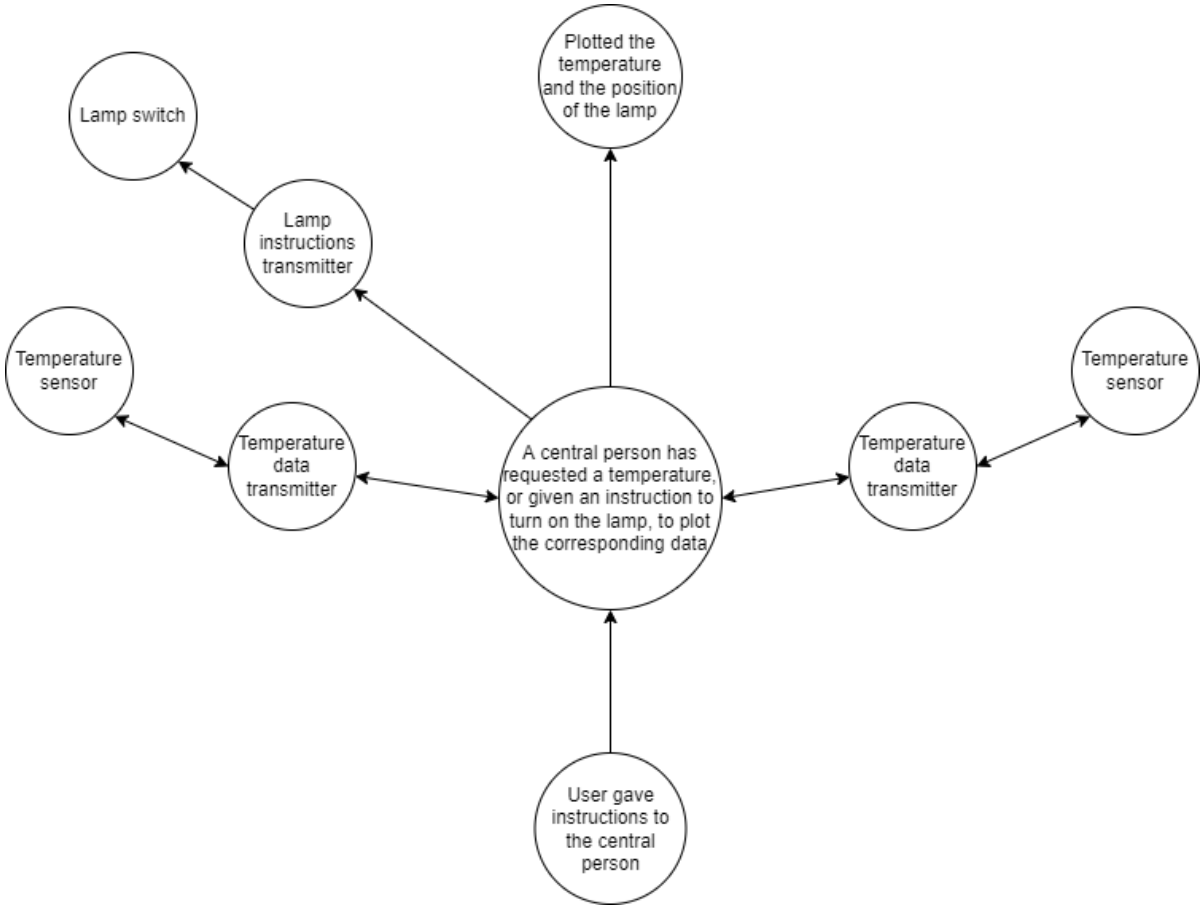

Figure 4: First architecture invented by students

From the planning to actually playing the project and considering it finished took about 20 minutes. During this time, there was a lot of brainstorming, discussion, and refinement, followed by the actual play. Obviously, because of the age and the situation, they were very disciplined and focused, which is not expected in a primary or secondary school, for example.

We had very few roles during the session. When they were ready with 1-1 phase, we gave them the next development opportunity or lesson to learn together (Was it effective this way? Could it be done differently? etc.) What happened in concrete terms is shown in Table 2.

Suggestions – guiding	Actual output	Lessons learned	IT analogy
Enter a new user in the game.	An interesting phenomenon compared to the previous ones was that when the two users asked for the light to be switched on at almost the same time, the central person had to wait until the corresponding data transmitting person returned.	During the game, they experienced this as a problem, because in reality, long seconds passed here, but we discussed that of course we are slower than the way the data actually goes, but this kind of race conditions happen with real software as well. The problem is needed to be solved.	HTTP (we compared it mainly to browsing); server-client connection; we would have to write 4 programs if we wanted to actually implement it.
Enter a second plotter/visualizer.	Here the central man was already overloaded, he couldn't do all his tasks because he would have had to communicate with too many people at the same time.	The two drawings cannot be done by one person, even if they have to draw the same thing. It was suggested among the students that the central man should be taken out of the system.	Queue, display completely separated from the server.
Let's solve it without a human transmitter (by speaking out loud).	Many people spoke at the same time, but the system worked through refinements.	Students perceived that the problem was communication. After 1-2 real-life tests, very strict rules were made about who could speak when, and everyone was given a different tone of voice to distinguish who was speaking. Afterward, we learned the lesson that even in real software you have to somehow strongly separate the messages, you have to send an identifier, so we know where the temperature came from.	Network protocols; data packet standards (data description languages); publish-subscribe protocol.
See the temperatures on the board continuously without being asked.	The two temperature-sensing men were saying the data out loud at the same time. A separate person had to be called in to record only the temperatures continuously.	Following the previous rules, the data can be separated.	Multithreading.
Let's solve it with two central people.	We couldn't actually play it, because we didn't have as many people as we needed.	It unnecessarily increases the number of communications and responsibilities.	-
Let's solve it without a central person.	It has been implemented, leaving many students without an assignment.	In this case, the publish-subscribe mode is more useful than the request-response model. The system has become more flexible.	-

Table 2: What happened during the live test

Overall, we think the session went well and delivered the expected results. The problems encountered were mainly architectural or network communication problems, and all of them can be analogised to an IT concept. The students (although not consciously) designed software architectures, faced critical and common problems and provided solutions.

Last but not least, the session was a lot of fun. The students had to work together, they had to find solutions together, and they could only do this by working well together. And the situations were often funny.

5.2 Answers to the questionnaire

In the followings, we would like to present the answers for the questionnaire.

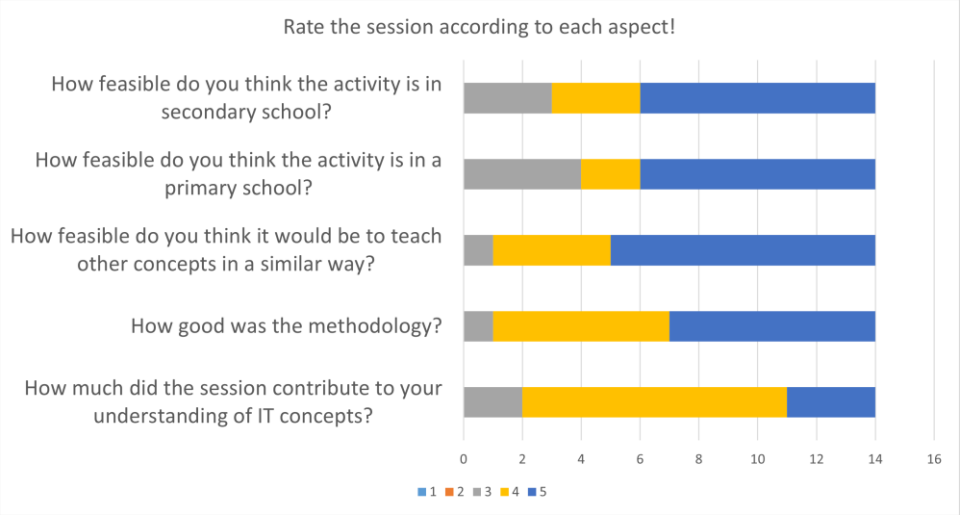

Figure 5: Rate the session according to each aspect!!

According to the graph in Figure 5, the students in the teaching profession rated the session as good or excellent. We welcome the confirmation that the method is worth trying in primary and secondary schools. Unfortunately, we missed this, but we should definitely do it at several levels and in several groups for the sake of completeness.

Although a 4 is good, you can't get past the results of the last question, where only three people answered with a 5. It could be because they were already familiar with most of the concepts, but of course, it could also be that they didn't think this was the most effective way of understanding. We can agree with that. We don't think the methodology is the most effective, but it is the most straightforward. Teaching software architectures or network communications in primary schools is far from trivial and is rarely implemented. The methodology offers a new alternative to this problem.

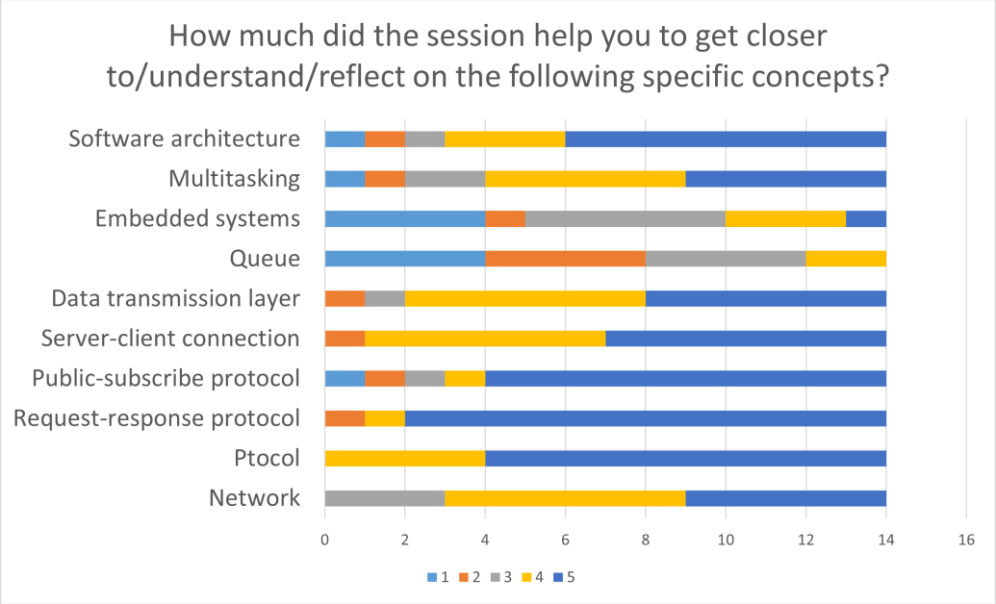

Figure 6: How much did the session help you to get closer to/understand/reflect on the following specific concepts?

Based on the results in Figure 6, it can be stated that students generally believe that the session contributed to their understanding of various IT concepts. Questions about network protocols were particularly well answered, which is important because this is what we wanted to focus on during the session. The notion of "queue" was left at the bottom, which is understandable because it was only just mentioned, and not particularly discussed.

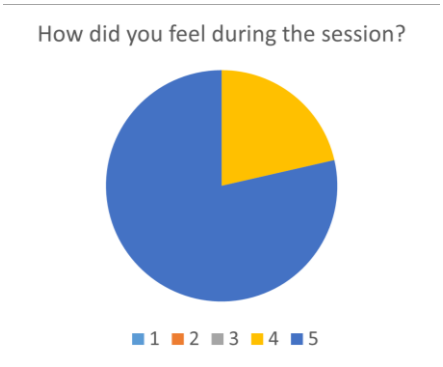

Figure 7: How did you feel during the session?

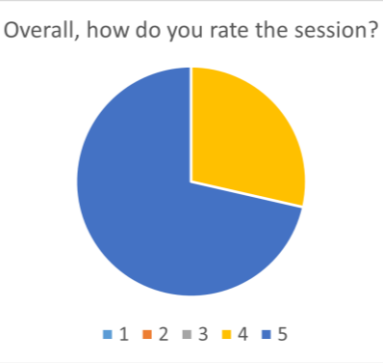

Figure 8: Overall, how do you rate the session?

Figures 7 and 8 show that, overall, students enjoyed the session and found it useful.

6. Conclusion

IT thinking can be taught and demonstrated without a computer. In this article, we present a situational game that can be used with primary and secondary school students, focusing on the design of software systems and network communication. The exercise can be valuable because these concepts are very difficult to demonstrate.

The method includes possibilities for modification (some of which we have presented), so that the teacher can adapt the game to a particular concept, age group and level of prior learning.

Bibliography

1. T. C. Bell, I. H. Witten, M. Fellow: *Computer Science Unplugged . . . off-line activities and games for all ages* (1998)
2. C. Duncan and T. Bell: *A Pilot Computer Science and Programming Course for Primary School Students*, in Proceedings of the Workshop in Primary and Secondary Computing Education, London, United Kingdom (2015) https://doi.org/10.1145/2818314.2818328
3. Y. Feaster, L. Segars, S. K. Wahba and J. O. Hallstrom: *Teaching CS Unplugged in the High School (with Limited Success),* in Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education, Darmstadt, Germany (2011) https://doi.org/10.1145/1999747.1999817
4. Á. N. Erdősné: *Teaching Graphs for Contestants in Lower-Secondary-School-Age*, OLYMPIADS IN INFORMATICS, vol. 11 (2017) 41-53 http://doi.org/10.15388/ioi.2017.04
5. P. Bernát: *The Methods and Goals of Teaching Sorting Algorithms in Public Education*, Acta Didactica Napocensia, vol. 7 (2014), 1-10
6. T. Bell, F. Rosamond and N. Casey: *Computer Science Unplugged and Related Projects in Math and Computer Science Popularization*, in The Multivariate Algorithmic Revolution and Beyond: Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday, H. L. Bodlaender, R. Downey, F. V. Fomin and D. Marx, Eds., Berlin, Springer Berlin Heidelberg (2012) 398-456 https://doi.org/10.1007/978-3-642-30891-8_18
7. B. Clarke: *Computer Science Teacher : Insight into the computing classroom*, Swindon, United Kingdom: BCS, The Chartered Institute for IT (2017)
8. O. Hazzan, T. Lapidot and N. Ragonis: *Guide to Teaching Computer Science*, London: Springer London (2014) https://doi.org/10.1007/978-1-4471-6630-6
9. *Lesson 4: Dance Party: Unplugged*, Code.org, https://curriculum.code.org/hoc/unplugged/4/ [Accessed 01/06/2022].
10. T. Bell and J. Vahrenhold: *CS Unplugged---How Is It Used, and Does It Work?*, in Adventures Between Lower Bounds and Higher Altitudes, H. Böckenhauer and W. Unger, Eds., Springer International Publishing (2018) 497-521 https://doi.org/10.1007/978-3-319-98355-4_29
11. V. Zhang: *TechTarget*, https://www.techtarget.com/iotagenda/blog/IoT-Agenda/The-pros-and-cons-of-IoT-edge-analytics . [Accessed 06 06 2022].
12. M. b. Mohamad Noor and W. H. Hassan: *Current research on Internet of Things (IoT) security: A survey,* Computer Networks, vol. 148 (2019) 283-294 https://doi.org/10.1016/j.comnet.2018.11.025

Authors

KOROM Szilárd
Eötvös Loránd University, Faculty of Informatics, Department of Media and Educational Informatics, Hungary,
e-mail: korom.szilard@inf.elte.hu

ILLÉS Zoltán
Eötvös Loránd University, Faculty of Informatics, Department of Media and Educational Informatics, Hungary,
e-mail: illes@inf.elte.hu

About this document

Published in:

CENTRAL-EUROPEAN JOURNAL OF NEW TECHNOLOGIES IN RESEARCH, EDUCATION AND PRACTICE

Volume 4, Number 1. 2022.

ISSN: 2676-9425 (online)

DOI:

10.36427/CEJNTREP.4.1.4452

License