BLAST: Block Applications for Things

Michael Freund[1][0000−0003−1601−9331], Thomas Wehr[1][0000−0002−0678−5019], and Andreas Harth[1,2][0000−0002−0702−510X]

[1] Fraunhofer Institute for Integrated Circuits IIS, Nürnberg, Germany
firstname.lastname@iis.fraunhofer.de
[2] Friedrich–Alexander University Erlangen-Nürnberg, Nürnberg, Germany

Abstract. An open research challenge in the deployment of digital twins is the interaction with humans. To facilitate this, we have created a digital twin based on Web of Things technology and a block-based visual programming language and execution environment called BLAST. We show that BLAST can be used to create programs that interact with a digital twin and IoT devices to control and monitor a process.

Keywords: Block-based Programming · Web of Things.

1 Introduction

IoT devices can be used for state monitoring, energy consumption analysis, simulation and intelligent optimization [3]. However, there are still some research challenges that need to be overcome. According to Semeraro et al. [6], the main research challenges in implementing a DT are in the areas of architecture, interoperability, and interaction capabilities.

Architecture refers to the fact that no design standard currently exists, so different DTs use varying technologies, interfaces, and communication protocols. A reference architecture could contribute to the flexibility and reusability of DTs. Interoperability includes a DT's ability to capture a multi-stage product lifecycle, which requires collecting data from external sources and from other companies. However, to achieve this, data exchange between multiple DTs must be enabled. The area of interaction capabilities refers to the fact that Digital Twins will not be involved in any important decisions without humans. Therefore, humans should find it easy to interact with DTs. Some other authors have also suggested that human-computer interaction is an open research question that needs to be resolved before widespread deployment of DTs in industry [4].

In this paper, we want to address the problem of human-digital twin interaction. Semeraro et al. [6] propose in this context to focus on the development of the Digital Twin so that it is able to interact smoothly with humans. Our proposed solution follows this approach. We have described a digital twin using RDF and ontologies in human and machine-readable form. However, our solution includes another perspective to improve human-DT interaction. We developed an easy-to-use interaction interface for humans with digital twins based

on Google's Blockly[1]. To show the feasibility of this approach, we developed and implemented a demonstrator for a commissioning process as a proof of concept.

2 Block-based Visual Programming

The programming languages currently in use are primarily text-based and are difficult to access for laypersons without prior knowledge. One possible approach to improve accessibility and simplify the creation of computer programs are so-called Visual Programming Languages (VPL). VPLs do not require the user to enter any text, they are instead based on graphical elements like blocks, tables or diagrams. Programs are created by arranging the available graphical elements in the correct order. The block-based approach of VPLs has become established in recent years, especially as an entry point for novice programmers.

Bau et al. [1] pointed out that block-based programming languages teach programming concepts in a simple way due to an intuitive and friendly user interface and that after using block-based languages it is much easier for users to switch to traditional text-based languages.

The basic idea of the block-based approach is that predefined blocks with images or text are arranged into programs by using the drag-and-drop principle. Blocks can be geometrically aligned with other compatible blocks, like in a jigsaw puzzle, to form complex programs. Since blocks can only be arranged in the correct way, no syntax errors can occur [7]. The created block programs can be translated into any predefined programming language, resulting in a normal source code. The block-based approach can be used not only for learning programming concepts, but also for more complex problems. For example, it has been shown that block languages can be successfully used in programming industrial robots [8] or executing SPARQL queries [2]. Both areas, which normally require advanced programming skills.

3 BLAST - Block Applications For Things

Due to the success of block-based languages in different challenging areas, we have developed our own block language based on Google's Blockly[2] library. Blockly is a pure JavaScript library for creating block-based languages and editors. Our block-based language simplifies the interaction with IoT devices, digital twins or other web resources without the need for a deeper understanding of different programming languages or network protocols such as Bluetooth or HTTP. We have named our block language "Block Applications For Things", or BLAST for short (See figure 1). The browser-based BLAST uses web Bluetooth to communicate with IoT devices, custom control and command blocks, and generates JavaScript code that can be used by any JavaScript interpreter. To interact with

[1] https://developers.google.com/blockly/
[2] https://developers.google.com/blockly

the IoT devices, BLAST requires custom drivers for each device, which represent an extended form of the WoT TD and contain the metadata and interaction affordances as well as the JavaScript code needed for communication.

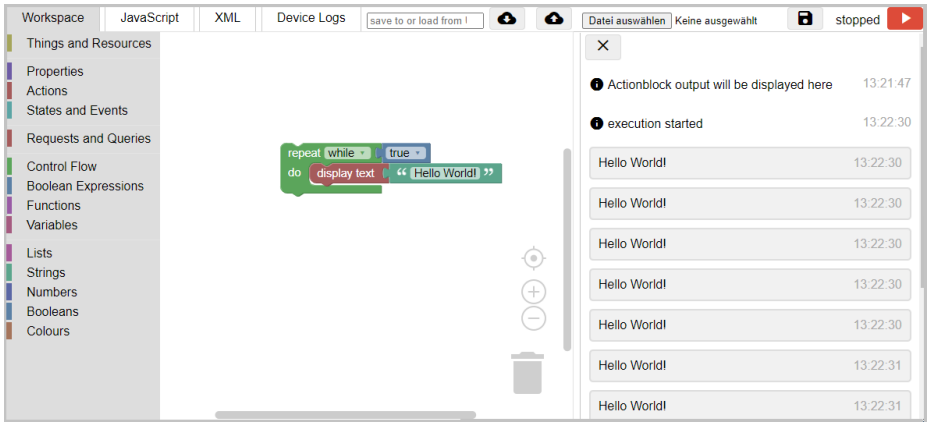

Fig. 1. Graphical user interface of the BLAST programming environment.

When choosing the vocabulary for our block-based language BLAST, we resorted to a mixture of natural language and computer language [5]. This means that in BLAST, the blocks are arranged into readable sentences, as in a natural language, but the blocks also contain a few technical programming terms, such as "repeat while true" instead of "forever". This makes the BLAST vocabulary both understandable for beginners, but still flexible.

BLAST includes standards from other programming languages such as loops, functions, and variables, but also WoT and IoT related blocks that can be grouped under the three interaction affordances Properties, Actions, and Events. Properties of connected IoT devices can be read and written, actions refer to long-running processes that can be initiated and events are asynchronous and based on the "Event Conditon Action" (ECA) rule. If an event occurs, the condition is checked and if necessary the corresponding action is executed. Furthermore, it is possible to send HTTP requests with arbitrary header and body or load knowledge graphs from a URI and execute SPARQL queries on the records, which allows interaction with read-write linked data APIs or other REST APIs. If the available blocks are not sufficient to implement a desired functionality, there is a block in which native JavaScript code can be entered. This block makes almost everything that is possible in JavaScript also possible in BLAST.

4 Supported Devices and Services

To demonstrate how BLAST can interact with WoT devices, we provide a Web of Things abstraction over devices communicating with Bluetooth Low Energy

and USB HID. We also provide access to online APIs related to speech input and output.

cell1	cell2	cell3
cell4	cell5	cell6
cell7	cell8	cell9

5 Conclusion

As IoT devices become more and more relevant in the industrial environment, it is essential to investigate how the interaction between humans and the IoT devices can be seamlessly implemented. So far, papers have tended to take the approach of simplifying the programming of devices.

In contrast, the solution proposed in this paper incorporates another perspective, the user perspective. In addition to creating a digital twin in human- and machine-readable form based on Semantic Web technologies, we explored the use of an easy-to-use block-based programming environment to interact with devices. For this purpose, we built a demonstrator based on Lego. Using the picking process depicted by the demonstrator, we were able to use BLAST control and simultaneously monitor the running process. With the help of the experimental setup, we were able to demonstrate that block-based languages are fundamentally suitable for interacting with DTs and IoT devices. In testing the performance of BLAST, we also found that the execution speed is slower, but only by 51,04 percent. Based on the accessibility and usability benefits, the approach of using block languages to interact with DTs should be pursued.

We are currently working on a way to execute the created BLAST program independently of browsers. To this end, we are developing an execution environment that can load BLAST programs, convert them to JavaScript, and execute them as a server. In the future, it would enrich BLAST if there was a way to consume WoT TDs and derive interaction possibilities, which would allow BLAST to interact with any WoT device.

References

1. Bau, D., Gray, J., Kelleher, C., Sheldon, J., Turbak, F.: Learnable programming: Blocks and beyond. Commun. ACM **60**(6), 72–80 (may 2017). https://doi.org/10.1145/3015455, 10.1145/3015455
2. Bottoni, P., Ceriani, M.: Using blocks to get more blocks: Exploring linked data through integration of queries and result sets in block programming. In: 2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond). pp. 99–101 (2015). https://doi.org/10.1109/BLOCKS.2015.7369012
3. Cimino, C., Negri, E., Fumagalli, L.: Review of digital twin applications in manufacturing. Computers in Industry **113**, 103130 (2019). https://doi.org/10.1016/j.compind.2019.103130

4. Kuehner, K.J., Scheer, R., Strassburger, S.: Digital twin: Finding common ground – a meta-review. Procedia CIRP **104**, 1227–1232 (2021). https://doi.org/10.1016/j.procir.2021.11.206, 54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0
5. Pasternak, E., Fenichel, R., Marshall, A.N.: Tips for creating a block language with blockly. In: 2017 IEEE Blocks and Beyond Workshop (B B). pp. 21–24 (2017). https://doi.org/10.1109/BLOCKS.2017.8120404
6. Semeraro, C., Lezoche, M., Panetto, H., Dassisti, M.: Digital twin paradigm: A systematic literature review. Computers in Industry **130**, 103469 (2021). https://doi.org/10.1016/j.compind.2021.103469
7. Weintrop, D.: Block-based programming in computer science education. Commun. ACM **62**(8), 22–25 (jul 2019). https://doi.org/10.1145/3341221
8. Weintrop, D., Shepherd, D.C., Francis, P., Franklin, D.: Blockly goes to work: Block-based programming for industrial robots. In: 2017 IEEE Blocks and Beyond Workshop (B B). pp. 29–36 (2017). https://doi.org/10.1109/BLOCKS.2017.8120406