Hindawi
Scientific Programming
Volume 2017, Article ID 1231430, 6 pages
https://doi.org/10.1155/2017/1231430

Review Article

A Survey on Visual Programming Languages in Internet of Things

Partha Pratim Ray

Department of Computer Applications, Sikkim University, Gangtok, Sikkim 737102, India

Correspondence should be addressed to Partha Pratim Ray; ppray@cus.ac.in

Received 31 October 2016; Accepted 19 February 2017; Published 8 March 2017

Academic Editor: Alok Mishra

Visual programming has transformed the art of programming in recent years. Several organizations are in race to develop novel ideas to run visual programming in multiple domains with Internet of Things. IoT, being the most emerging area of computing, needs substantial contribution from the visual programming paradigm for its technological propagation. This paper surveys visual programming languages being served for application development, especially in Internet of Things field. 13 such languages are visited from several popular research-electronic databases (e.g., IEEE Xplore, Science Direct, Springer Link, Google Scholar, Web of Science, and Postscapes) and compared under four key attributes such as programming environment, license, project repository, and platform supports. Grouped into two segments, open source and proprietary platform, these visual languages pertain few crucial challenges that have been elaborated in this literature. The main goal of this paper is to present existing VPLs per their parametric proforma to enable naïve developers and researchers in the field of IoT to choose appropriate variant of VPL for particular type of application. It is also worth validating the usability and adaptability of VPLs that is essential for selection of beneficiary in terms of IoT.

1. Introduction

User interaction is the main concern in today's software industry. Among many existing techniques, the implication behind Visual Programming Language (VPL) is the most promising and prevalent. A VPL is like any available programming language that lets user create programs by manipulating program elements graphically (while allowing programming with visual expressions, spatial arrangement of graphic symbol, etc.) rather than by specifying them textually [1]. For instance, many VPLs which are also known as Dataflow Languages are designed based on the idea of utilizing arrows and boxes, where arrows are used to connect the boxes by establishing a seamless relationship between boxes (i.e., entity). VPLs are normally used for educational, multimedia, video games, system development/simulation, automation, and data warehousing/business analytics purposes. For example, (a) Scratch [2], a platform of Massachusetts Institute of Technology, is designed for the kids in class 12 and after school programs; (b) Pure Data (Pd) [3] is designed for creating interactive multimedia and computer music; (c) Unreal Engine 4 [4] uses "Blueprints" to program video games; (d) VisSim [5] allows user to make complex mathematical models in smarter and faster way while executing them in real-time; (e) CiMPLE [6] is used for teaching automation through robotics; and (f) IBM Cognos Business Intelligence [7] is used for front-end programming in Business Intelligence (BI) applications whereby generating SQL queries to run against Relational Data Base Management Systems (RDBMS). Figure 1 presents the percentagewise distribution of VPLs in the illustrated domains. Out of 89 differently surveyed VP/Ls, system simulation and multimedia hold 60% of the market, marking 35% and 25%, respectively.

Although, several domains of applications are under the practice of VPLs, an emerging field of computing-Internet of Things (IoT) is still lingering far behind other sectors. In IoT, researchers direct main attention towards interconnection between several heterogenous objects or "things" with each

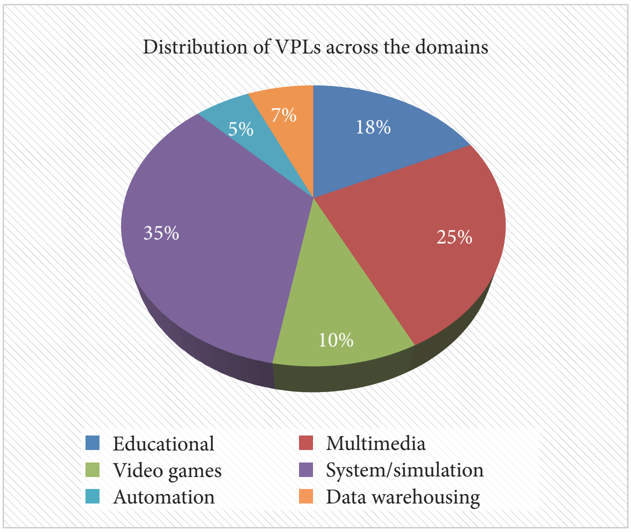

Figure 1: Percentagewise distribution of VPLs.

other by means of interoperable and unified platform creation to provide smarter and more efficient way of communication between digital and physical world. Applications of IoT include key areas like domotics, healthcare, smart city, automation, transport, education, environment monitoring, and industry [8–15]. In this context, we may refer to Gartner which has envisioned that about 50 billion of things will be connected to the Internet by 2020. This has already created a huge buzz in the IT industry especially in manufacturing sector where 12.4 billion USD's business was recorded at the end of 2015 [16].

Despite IoT's prospect, very few attentions have been given over the development and designing process of programming languages, mainly at Device-to-Device (D2D) relationship [17]. VPL would in this regard act as a key tool for further enhancement, progress, and motivation towards developers in this field (i.e., IoT) while reducing time-to-market in product (software/hardware) development life cycle. This, in turn, necessitates the need of study and analysis of VPL and its impact on IoT that has not been yet proved by existing literatures. This paper surveys current VPLs being used for application developments in IoT specific framework. The key contributions of this paper are given as follows:

(i) To use state-of-the-art survey of 13 existing VPLs for development of applications using IoT

(ii) To obtain parametric results on various aspects of VPLs

(iii) To discuss and analyze current trends towards selection of VPLs into an IoT specific application

(iv) To identify key issues in augmentation of VPLs in IoT application scenario

This paper is organized as follows. Section 2 presents the state-of-the-art survey of VPLs in IoT. Section 3 illustrates the discussion and analysis in this regard as well as presenting key challenges that need urgent consideration by the educators, industries, and developers. Section 4 concludes this paper.

2. State of the Art

IoT relates to numerous kinds of heterogeneous microcontroller enabled hardware platforms for multiple number of applications' development. Each of these platforms paves dedicated Integrated Development Environment (IDE) and separates programming languages that creates difficulty in seamless transformation from one domain to another, hence delaying product development while incurring excessive expenditure in terms of cost and man power. This section surveys the VPLs being used and developed by several organizations for smooth and ease of programming hardware platforms for IoT based applications with just few clicks/drag-drop procedures, without knowing much about the language (i.e., expression, statement, loop clause, and functional orientation). Popular electronic research databases (such as IEEE Xplore, Google Scholar, Web of Science, Science Direct, and Springer Link) and IoT specific web blogs (Postscapes, Internet of Things-Wired UK, IoT analysis and commentary-O'Reilly Radar, Internet of Things-The Guardian, and Internet of Things Council) are searched in detail while finding these VPLs.

The existing VPLs, involved in this study, are sometimes not just language by itself but a full-fledged IDE. They have been divided into two segments based on their usage licensing, that is, (a) open source and (b) proprietary platform. Further the survey is performed on four classes of characteristics such as programming environment, license, project repository, and platform support.

2.1. Open Source Platforms

(1) Node-Red. It is a visual tool developed for wiring IoT centric applications being hosted on the Github repository (https://github.com/node-red/node-red). It can be run on variety of hardware and software platforms such as Raspberry Pi, BeagleBone Black, Docker, Arduino, Android, IBM Bluemix, Amazon Web Services, and Microsoft Azure under Open Source-Apache 2.0 license. During development of IoT applications, the following APIs are taken into consideration: Admin HTTP API, Runtime API, Storage API, and Editor UI API for administration, embedding other application, runtime data storage and running jQuery template, respectively. It further supports JavaScript, HTML, and JSON language for node creation activities normally found at following port http://localhost:1880.

(2) NETLab Toolkit. It is helpful for drag and drop based IoT based applications development process hosted at http://www.netlabtoolkit.org. Further it provides a simple web interface to connect sensors, actuators, media, and networks associated with smart widgets for development of quick prototype iteration, experiment, and testing just by sketching in heterogenous genre of hardware and building the connected systems. Arduino and latest Linux embedded

systems like the Raspberry Pi, Intel Galileo, and Arduino Tre are currently supported. HTML5, Node.JS, and JavaScript are used for application sketching and server programming purposes under the GNU General Public License.

(3) Ardublock. It is a dedicated GPL, hosted at http://blog.ardublock.com, for programming Arduino and its variant platforms. It runs on Eclipse IDE (as well as Arduino IDE) while allowing developer to code in Java under the GNU General Public License. It is a popular web based VPL platform that helps user to connect and visually program Arduino to create an IoT based application.

(4) Scratch for Android (s4a). It is Scratch modification, made for integration and experimentation with Arduino base IoT products, currently hosted at http://s4a.cat. The s4a has designated protocol stack for communication with Arduino boards. It also supports Android users to get associated with Arduino through HTTP by means of Scratch based remote sensor protocol under the GNU General Public License v2 (GPL2).

(5) Modkit. It is another drag and drop VPL, designed for popular microcontrollers including Arduino, littleBits, Particle Photon, MSP340, Tiva C launch pad, and Wiring S, being hosted at http://modkit.io. It also supports Scratch like event driven and multithreaded model for building IoT related products at ease. This VPL belongs to get build in desktop environment under the GNU General Public License.

(6) miniBloq. This VPL platform is for programming Multiplo™, Arduino, RedBot, and RedBoard in desktop environment. It is available at http://blog.minibloq.org under the Robot Group Multiplo Pacifist License (RMPL) that is an MIT license with a restriction over the development on defence and military projects. C++ language is key of this VPL that runs with help of wxWidgets (http://www.wxwidgets.org).

(7) NooDL. It provides an efficient and effortless web based visual programming environment for IoT related product developments, currently hosted at http://www.getnoodl.com. It supports Arduino and any other physical devices while considering underlying "virtual things" aspect with it. Besides, Bluetooth based local communication with the devices is also possible that allows MQTT broker agent (API) for seamless connection. Dynamic data visualization and analytics are also integrated with this VPL that may be used to access the data stored at local or remote cloud servers with support of Android. NooDL is restricted with the NooDL End User License (NEUL) [18]. It does not provide any external programming language support for coding the applications however.

2.2. Proprietary Platforms

(1) DGLux5. It is a drag and drop based VPL platform for development of IoT applications, currently hosted at http://www.dglogik.com/products/dglux5-ioe-application-platform. This desktop centric approach has link, command, and control data dash board. It provides a personalized interaction by leveraging flexible deployment options (hardware platforms), customized chart, and real-time visualization tool under DGLux Engineering License.

(2) AT&T Flow Designer. It is built upon cloud based time-series data storage platform while involving an intuitive visual tool that enables developer to create IoT supported prototype, being hosted at https://flow.att.com. It offers a special inclusion named "nodes" that is already preconfigured to allow seamless and smooth access to multiple data sources, communication methods, cloud services, and device profiles. Thus, it reduces time-to-market phase in business development process. It supports several third party commercial platforms/APIs (e.g., Twilio and SMTP push mail/notification) for real-time data aggregation and communication between user and applications under the GNU General Public License v3 (GPL3).

(3) Reactive Blocks. It is a visual cum model-driven desktop development environment designed for supporting following tasks, such as, formal model analysis, hierarchical modeling, and automated code generation, available at http://www.bitreactive.com/reactive-blocks. It also provides built-in Java library so that a developer can create reusable and complex IoT applications graphically. Further, OSGi, Kura, and ESF IoT platforms can get merged with Reactive Blocks. MQTT, HTTP, and other IoT related APIs are also used for application development. Java is the key behind the production of Reactive Blocks VPL platform that helps to connect with Modbus, Raspberry Pi, and USB Camera. It is distributed in the market under the Eclipse Public License (EPL).

(4) GraspIO. It provides a drag and drop based, cloud assisted desktop application development platform for interaction with Arduino, Raspberry Pi, GIO Arm, GIO TetraPod, and GraspIO boards, currently hosted at http://www.graspio.com. It is able to support the USR-WiFi 232-G module to provide standard wireless communication by linking 3 Analog/Digital input, 11 touch points, ultrasonic, and GP2D port as sensors and 2 DC motor ports and 8 servo ports as actuators under BSD license.

(5) Wyliodrin. It is a browser enabled VPL that offers communication and development opportunities with Arduino, BeagleBone Black, Raspberry Pi, Intel Galileo, Intel Edison, UDOO, ZedBoard, and Red Pitaya platforms. It also offers multiple programming languages like C, C++, Objective-C, Shell Script, Perl, Python, JavaScript, PHP, C#, Java, Pascal, and so on to develop IoT applications. It is particularly efficient when developer wants to connect IoT devices from smart phones running on either Android or iOS under GNU General Public License v3 (GPL3).

(6) Zenodys. It a browser based specially designed VPL for leveraging IoT based industry 4.0 revolution that is evitable in coming years, hosted at https://www.zenodys.com. Its run-time environment can be deployed to the Raspberry Pi or similar other prospective Linux based industrial IoT

TABLE 1: Comparison between open source and proprietary VPLs.

Type of VPLs	Name of VPLs	Programming environment	License	Project repository	Platforms supported
Open source	Node-Red	Web	Open Source-Apache 2.0	Github	Raspberry Pi, BeagleBone Black, Docker, Arduino, Android, IBM Bluemix, Amazon Web Services, Microsoft Azure under
	NETLab Toolkit	Web	GPL	Self	Arduino and latest Linux embedded systems like the Raspberry Pi, Intel Galileo, and Arduino
	Ardublock	Web	GPL	Self	Arduino
	Scratch for Android (s4a)	Web	GPL2	Self	Arduino
	Modkit	Desktop	GPL2	Self	Arduino, littleBits, Particle Photon, MSP340, Tiva C
	miniBloq	Desktop	RMPL	Self	Multiplo, Arduino, RedBot, and RedBoard
	NooDL	Web	NEUL	Self	Arduino, Android
Proprietary	DGLux5	Desktop	DGLux Engineering License	Self	Raspberry Pi, BeagleBone, DGBox
	AT&T Flow Designer	Desktop	GPL3	Github	AT&T IoT SIM
	Reactive Blocks	Desktop	EPL	Self	Modbus, Raspberry Pi and USB Camera
	GraspIO	Desktop	BSD	Self	Arduino, Raspberry Pi, GIO Arm, GIO TetraPod, and GraspIO boards, Android
	Wyliodrin	Web	GPL3	Self	Arduino, BeagleBone Black, Raspberry Pi, Intel Galileo, Intel Edison, UDOO, ZedBoard and Red Pitay
	Zenodys	Desktop	—	Self	Raspberry Pi, Zenobox

gateways (e.g., Zenobox). This cloud supported platform (e.g., ZenoCloud and Microsoft Azure) encourages developers to associate 3rd party hardware, protocols (e.g., Modbus TCP/RTU, I2C, HTTP, TCP/IP, UDP, RS232, RF, BLE, One Wire, En-Ocean, and Z-Wave), devices, data, APIs, and applications to interact with it easily.

3. Discussions

Table 1 compares the VPLs per open source and proprietary segments while incorporating programming environment,license,project repository, andsupported platforms. The reason behind choosing these four parameters is quite linear, that is, comparativeness metrics. Comparative metrics is usually designed in such a way that research can be progressed by means of equal opportunity in inclusiveness scenario. Here, 13 VPLs are being studied not only for the sake of informing the readers about what solutions are available in market, but also for the sake of disseminating applicability and selectivity while developing an IoT based application. An IoT designer must be helpful by gaining the preface made herein this article before going for an application development environment (i.e., desktop or web) as well as the convergence with a particular genre of hardware boards. Licensing is a factor in IoT based product development not only for its incurred cost of royalty service the vender, but also for its code stratification and technological adaptation purposes.

The results have been obtained by surveying the VPLs per various metrics. Few surprising items emerged which are described as follows, showing that open source and

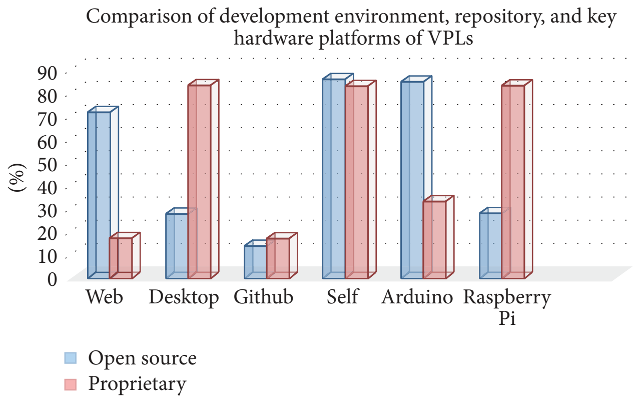

FIGURE 2: Percentagewise positioning of development environment, project repository, and key hardware platforms.

proprietary VPLs do infer in many ways from each other. Figure 2 illustrates the comparative analysis between these two portions of VPLs. For ease of understanding and analysis, only three crucial parameters are considered such as development environment, project repository, and hardware boards. It is observed that 72% of open source and 17% of proprietary VPLs are run from the web browser without any need of installation of packages in the host machine. In context of the desktop usage (i.e., nonweb browser based), this situation is just opposite (i.e., 28% of open source and 83% of proprietary). Going to the next parameter, that is, project repository, it is seen that Github is almost equalled by both the segments (i.e., 15.5% on average) which is followed by the self-server storage facility (i.e., 85.5% on average). Among variety of hardware platforms, Arduino and Raspberry Pi are the top two mostly compatible devices boards. Arduino leads the race, marking 85% and 33% in open source and proprietary segment, respectively. Again, the case is different in the usage of Raspberry Pi, where 28% of open source VPLs do practice on to it and 83% of proprietary. However, BeagleBone Board is also approaching to hold the first two boards. Other devices are less frequently getting accustomed with the deployments by VPLs. The GNU GPL is marking the top of the list, whereas Apache 2.0, EPL, BSD, and DGLux Engineering License are similarly gradually getting attention in the VPLs' market. The applicability of JavaScript has emerged as one of the key programming languages in terms of coding the VPLs. Out of 13 surveyed VPLs, 7 belong to open source and the rest belong to proprietary. This also shows the trend of current scenario in VPLs market for IoT development. Although there is slight difference of division among the two segments, it seems that open source VPLs are in the verge of domination of existing VPL supported IoT domain in coming times. As of Section 1, where 89 VPLs were studied over distribution among multiple domains of applications, none of these 13 VPLs was included in that list, which, surely points towards some technical and organizational challenges that are hampering the growth and adaptation of VPLs in IoT. The following section describes the challenges associated with integration of VPLs with IoT.

Challenges

(i) Extensibility. This is probably the vital problem in the surveyed VPLs. VPLs allow developer/user to perform a limited set of operations (things) easily, but precise edge cases are too far difficult (even impossible) to achieve in practice. These VPLs should give user more power, instead of constricting. This might give an opportunity for the IoT enabled application development process where extensibility is a fundamental need.

(ii) Slow Code Generation. Performance diagnosis is a key part of any developer's testing phase, especially in IoT where lots of devices are engaged into the system. It is always important to detect and solve the underlying problems. But, in case of these VPLs, it seems that they work on leaky abstractions, resulting in slow code generation which is nearly impossible to optimize by a developer.

(iii) Integration. Developers live in Integrated Development Environments (IDEs) and simulators world. If the IDEs and simulators are poor in effort and performance, they can make lives miserable! Hence, VPLs and IDEs (design editor) should be designed together for leverage ease of programming environment which is necessary for IoT.

(iv) Standard Model. This is another serious challenge in VPLs of IoT; there is no existing way to have a global standard model such as "mental model," used to give explanation of every human's thought process on working of "anything." Different service professionals like scientists, electrical engineers, mathematicians, IT industry programmers, and so on are taught and trained how to model the problem statement in different manner, for example, an electrical engineer who would be well qualified and able enough to predict and feel what an electrical system does by just looking at that very circuit diagram. Existing VPLs for IoT do pave the environment to the developer in many ways. Hence, there is a strong need of a standard modular structure or method so that a well-trained developer would be able to build any sort of applications on different genre of platforms.

(v) Abstraction. VPLs are designed for better presenting and working with the existing "abstractions," a metaphor that is used to let the developer manipulate something complex in terms of logic, for example, a function lets programmer present logical operations in form of mapping from an input section to an output section.

(vi) User Interface. VPLs in IoT may be broken into three broad categories of programming tools for different situations such as (a) nonprogrammers (naïve users) to perform basic automation tasks; (b) program learning environments, where it is not feasible to have typing or structuring of the program; and (c) advanced data-flow aggregator; it is well modeled by appropriate data-flows between self-contained logical boxes that essentially mimic the physical world.

4. Conclusion

At this end, we may finally seek for pros and cons of VPLs implied into IoT summarized as follows:

Pros: easy to "visualize" the programming logic (e.g., flow chart), good for naïve users to get associated with the concept of interactions among the logical structures, meant for rapid development of IoT products, less burden over handling "syntax error," and portable on a tablet (or hand held smart phone/device) or in situations where no physical keyboard is present.

Cons: sometimes large amount of time is spent over designing small IoT applications. For example, programming for blinking an LED with Arduino requires lot of graphical interconnections.

Despite VPLs' huge facilities and limited disadvantages, IoT seems to be suitably getting empowered by smooth entanglement and promotion with promising reduction in physical-digital interface.

This paper presents state-of-the-art survey in 13 existing VPLs being used for IoT application development. Popular electronic research databases such as IEEE Xplore, Google Scholar, Web of Science, Science Direct, and Springer Link and IoT specific web blogs (Postscapes, Internet of Things-Wired UK, IoT analysis and commentary-O'Reilly Radar, Internet of Things-The Guardian, and Internet of Things Council) are searched in detail while finding these VPLs. A comparative study has been performed between theses VPLs based on open source and proprietary mode of procurement. Analysis is done on four sections on each VPL that includes (1) programming environment, (2) licensing, (3) project repository, and (4) supported platforms. Presented results show a trendy inference towards implications for Arduino and Raspberry Pi based hardware prototyping boards. Most of the VPLs are presently being hosted in their self-repository. Choice of the programming environment is more or less equally distributed among desktop and web versions. Licensing of such VPLs is somewhat aggregated in and around GPL but other specific types are also under the coverage. Out of the selected issues, poor user interface, slow code generation, lack of standardized model, and absence of abstraction layer seem to resist the growth of VPLs in present time.

However, naïve as well as expert researchers in collaboration with the company incorporations may join to work for the improvement of present issues to facilitate upcoming start-ups (i.e., industry-academia) while paving a prosperous future in this field.

Conflicts of Interest

The author declares that there are no conflicts of interest regarding the publication of this paper.

References

[1] B. Jost, M. Ketterl, R. Budde, and T. Leimbach, "Graphical programming environments for educational Robots: open Roberta—yet another one?" in *Proceedings of the 16th IEEE International Symposium on Multimedia (ISM '14)*, pp. 381–386, IEEE, Taichung, Taiwan, December 2014.

[2] Scratch—Imagine, Program, Share, https://scratch.mit.edu.

[3] Pur Data, "Get Pure Data for Windows/Mac/Linux," January 2017, https://puredata.info/.

[4] Unreal Engine 4, https://www.unrealengine.com/.

[5] VisSim, http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/.

[6] CiMPLE, January 15, http://cimple.software.informer.com/.

[7] IBM Cognos Analytics, https://www.ibm.com/software/in/analytics/cognos/platform/.

[8] E. Fleisch, "What is the internet of things—an economic perspective," Auto-ID labs white paper, https://www.alexandria.unisg.ch/60578/1/AutoID%20-%20What%20is%20the%20Internet%20of%20Things%20-%20An%20Economic%20Perspective%20-%20E.%20Fleisch.pdf.

[9] H.-L. Truong and S. Dustdar, "Principles for engineering IoT cloud systems," *IEEE Cloud Computing*, vol. 2, no. 2, pp. 68–76, 2015.

[10] S. W. Kum, J. Moon, T. Lim, and J. I. Park, "A novel design of IoT cloud delegate framework to harmonize cloud-scale IoT services," in *Proceedings of the IEEE International Conference on Consumer Electronics (ICCE '15)*, pp. 247–248, Las Vegas, Nev, USA, January 2015.

[11] A. Celesti, M. Fazio, M. Giacobbe, A. Puliafito, and M. Villari, "Characterizing cloud federation in IoT," in *Proceedings of the 30th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA '16)*, pp. 93–98, Crans-Montana, Switzerland, March 2016.

[12] A. Taherkordi and F. Eliassen, "Scalable modeling of cloud-based IoT services for smart cities," in *Proceedings of the 13th IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops '16)*, Sydney, Australia, March 2016.

[13] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, "Integration of agent-based and Cloud Computing for the smart objects-oriented IoT," in *Proceedings of the 18th IEEE International Conference on Computer Supported Cooperative Work in Design (CSCWD '14)*, pp. 493–498, May 2014.

[14] F. Wortmann and K. Flüchter, "Internet of things: technology and value added," *Business and Information Systems Engineering*, vol. 57, no. 3, pp. 221–224, 2015.

[15] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business and Information Systems Engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[16] L. Columbus, "Roundup Of Internet of Things Forecasts And Market Estimates," 2015, http://www.forbes.com/sites/buiscolumbus/2015/12/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2015/#60f94aab48a0.

[17] Y. Chen and G. D. Luca, "VIPLE: visual IoT/robotics programming language environment for computer science education," in *Proceedings of the IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW '16)*, pp. 963–971, Chicago, Ill, USA, May 2016.

[18] NEUL, http://www.getnoodl.com/eula.