Ontology-Driven Automation of IoT-Based Human-Machine Interfaces Development

Konstantin Ryabinin(✉), Svetlana Chuprina, and Konstantin Belousov

Perm State University, Bukireva Str. 15, 614990 Perm, Russia
kostya.ryabinin@gmail.com, chuprinas@inbox.ru, belousovki@gmail.com

Abstract. The paper is devoted to the development of high-level tools to automate tangible human-machine interfaces creation bringing together IoT technologies and ontology engineering methods. We propose using ontology-driven approach to enable automatic generation of firmware for the devices and middleware for the applications to design from scratch or transform the existing M2M ecosystem with respect to new human needs and, if necessary, to transform M2M systems into human-centric ones. Thanks to our previous research, we developed the firmware and middleware generator on top of SciVi scientific visualization system that was proven to be a handy tool to integrate different data sources, including software solvers and hardware data providers, for monitoring and steering purposes. The high-level graphical user SciVi interface enables to design human-machine communication in terms of data flow and ontological specifications. Thereby the SciVi platform capabilities are sufficient to automatically generate all the necessary components for IoT ecosystem software. We tested our approach tackling the real-world problems of creating hardware device turning human gestures into semantics of spatiotemporal deixis, which relates to the verbal behavior of people having different psychological types. The device firmware generated by means of SciVi tools enables researchers to understand complex matters and helps them analyze the linguistic behavior of users of social networks with different psychological characteristics, and identify patterns inherent in their communication in social networks.

Keywords: Ontology engineering · Internet of Things · Human-machine interaction · Tangible interfaces · Firmware · Middleware · Scientific visualization · Visual analytics

1 Introduction

Universal computer input devices like keyboard and mouse build the habitual environment for user and normally suit everyday needs in terms of human-machine

The reported study is supported by Ministry of Education and Science of the Russian Federation, State Assignment No. 34.1505.2017/4.6 (Research Project of Perm State University, 2017–2019).

J. M. F. Rodrigues et al. (Eds.): ICCS 2019, LNCS 11540, pp. 110–124, 2019.
https://doi.org/10.1007/978-3-030-22750-0_9

interaction. However specific cases may require specific input hardware and specific software reactions to enable needed user experience. Moreover, real-world experience shows that the today's special cases are the common ones in the future. Nowadays high-level tools adapting the factory-assembled input devices to the specifics of particular task or hardware and software infrastructure are often missing.

When it comes to virtual reality applications, the user's eyes are covered with the head-mounted display, so it becomes very uncomfortable to use keyboard and special hardware control elements (like Oculus Touch) are required. Different vehicle simulators require special hardware control panels including steering wheel and pedals to provide full immersion and enable training function. Some content generation tasks also require special tools, for example, the digital painting requires drawing tablets to increase ergonomics. Special environment conditions may also require space saving, so touchscreens and trackpads are also very popular nowadays. Next, it is often required to bring the benefits of the Internet of Things (IoT) to legacy systems, to combine various old edge devices and sensors with smart ones, or to put the human into the IoT ecosystem. In our opinion, in the near future, putting human centric innovation to drive a composition of IoT systems, in which their constituent systems are individually discovered, becomes the everyday task.

In this context, Internet of Things become a promising branch of computer science and engineering. The technologies and design patterns developed for IoT can be applied to the human-machine interaction bringing it to the new customization level in a unified form. Different sensors provide the user with quite unlimited possibilities to control the computers via so-called tangible interfaces [8], and in the same time provide the automation systems with the same amount of possibilities to monitor the human activity. The problem here is the user's qualification: complex interconnections, sensor-based networks and device making normally require knowledge in both electronics and programming. In contrary, high-level software tools and handy interoperable IoT modules can be easy building blocks ready to be used even by the beginners. The leaders of IoT industry already provide particular solutions for typical use cases like, for example, the organization of Smart Home. But there is still a lack of high-level tools for building arbitrary IoT devices and middleware to turn them into human-machine interface (HMI). Thereby not only the legacy systems can get new HMI by means of IoT, but also the IoT systems themselves can be turned into human-centric ones.

We propose ontology-driven approach to build smart systems, which can automatically generate the firmware for IoT devices and the middleware to connect these devices with third-party software in adaptable way. Thereby this system is a way to automate the creation of software part of IoT-based HMI. Regarding the hardware part we suggest using particular electronic components meeting the price/quality balance. To demonstrate the proposed approach we present the new capabilities of SciVi platform[1] to tackle the problems mentioned above.

[1] https://scivi.tools/.

As follows from the description of our system presented in [17], the SciVi platform was initially intended to be the scientific visualization tool adaptable to different types of software solvers and input data formats. Our innovation presented in this paper is using SciVi platform as IoT-based HMI generator without changing the SciVi core. In this case IoT devices play the solvers' role. Moreover, the SciVi architecture allows using its modules separately to solve wide range of tasks, not limited by scientific visualization. For example, if third-party system already has its own visualizer, SciVi can be used as the middleware to provide new HMI capabilities including gesture-based ones.

2 Key Contributions

In this paper we describe the ontology-driven smart system, which enables automation of IoT-based HMI development. The following key results of conducted research are presented.

1. The ontology-driven approach is proposed to automate the firmware and middleware generation for the cases when IoT devices are intended to be used as a part of HMI to the third-party software.
2. The proposed approach is implemented within a smart system SciVi-Middle, which is built upon the adaptive scientific visualization toolset SciVi we developed previously.
3. The recommendations are made for choosing IoT-based HMI hardware with respect to price/quality ratio.
4. The discussed approach is used to tackle real-world problems of social network users behavior analysis.

3 Related Work

The essence of the IoT is the constant monitoring of the environment and allowing humans the remote management in order to ensure the symbiosis of real and digital worlds [7]. Being the basis of ubiquitous computing [7], IoT provides great possibilities to monitor the human activity [21]. This, in turn, opens up the new horizons of human-machine interaction utilizing more specific actions than traditional pushing of buttons. As stated in [7], "in an IoT environment, the human and computer interaction happens through objects containing embedded technology to sense or interact with their internal state and external environment". Human presence (detected by passive infrared sensor), gestures (detected by some kind of motion capture system, like for example [9]), voice and even physical condition (detected by wearable medical sensors) could be turned into the commands for the computing systems whereby multimodal HMI can be provided [24]. Moreover, IoT ensures so-called tangibility of HMI [8], enabling seamless coupling between physical objects and virtual data. As a result, smart systems can be created, which respect human skills and knowledge, as well as the dynamic user behavior [7].

However, achieving the desired user experience within IoT ecosystem is a challenging task. It is necessary to integrate diverse independent components into a one-stop solution, saving functionality and reliability of individual building blocks and ensuring seamless interoperability [10]. To tackle this challenge, both high-level user interface (UI) frameworks and middleware frameworks are demanded.

Middleware is a "glue" to stick together the existing programs and devices that were not originally designed to be connected. Detailed survey of modern IoT middleware made by Farahzadi et al. [6] shows the variety of platforms aimed to ensure integration of different hardware into a single ecosystem. The popular middleware helps to efficiently solve the interoperability issues, however the tasks of building versatile and ergonomic UI for this ecosystem still do not have unified solutions. This is why specific circumstances force developers to create related software from scratch.

The middleware should be very flexible, extensible and adaptable to ensure both interoperability of devices and versatile HMI capabilities [6]. One of the most adequate ways to achieve these features is a model-driven approach, when the system providing middleware functionality is governed by some formal model. In this case, the modification of the model will modify the system's behavior, while the code of system's core remains unchanged. If the used model maintains semantics, the system possesses context-awareness and thereby can be treated as intelligent one.

Well-proven semantic models are provided by ontology engineering. Since 2012 ontologies are widely used in IoT domain to tackle different key issues like heterogeneity of devices, interoperability and scalability problems, monitoring tasks, etc. [2]. The main point to use ontologies is their ability to be close to human conceptual level, while understandable to machines [11]. This is why we decided to use ontology engineering as a core methodology in automating the IoT-based HMI development.

In our previous research we studied the ways to automate monitoring and calibration of IoT devices based on ontology engineering methods and scientific visualization tools [20]. We implemented the corresponding capabilities within scientific visualization system SciVi. The current paper describes the next step of SciVi development generalizing its capabilities from visualization system to HMI platform. It was an intention to build HMI platform upon a visualization system, because visualization is considered as an essential part of human-machine interaction [13].

Research work related to the versatile HMI software for IoT has already justified ontology engineering in this field. As stated by Calderon et al., IoT is still too device-centric, but should be more human-centric, and the semantic technologies are one of the clues to achieve this human-centricity [5]. Ontologies were successfully used by Luz et al. to specify the human-machine computation workflow retaining knowledge and semantics of atomic tasks [11]. The system called Perci developed by Broll et al. relies on application model ontologies to compose an application interface and UI [4]. The research by Steinmetz et al.

is dedicated to using ontologies to enhance IoT middleware and personalize the visualization available for end users [23]. Several projects, for example the one by Noguera-Arnaldos et al. [1] or by Petnik et al. [14], utilize ontologies to provide users with natural language interface to IoT devices.

The distinctive features of our approach are the following:

1. We propose the platform for developing HMI *based on* IoT devices, not *to* the IoT ecosystem. Of course, the methods we suggest can be used to build interactive environments like Smart Home, similar to the ones described in [14] and [13]. However, the main intention behind our research was to enable device makers creating the custom gadgets to control third-party software.
2. We propose using cognitive interactive graphics [12] capabilities provided by scientific visualization and visual analytics tools to support HMI development and functioning.
3. We utilize complex extensible ontology-driven mechanism of data processing called semantic filtering [20] to enable customizable conversion of data representations and formats. The SciVi platform ontology-driven capabilities to adapt to specifics of different application domains and to tackle different problems of scientific visualization in a uniform way are proven in a number of use cases [18]. In this paper we describe the new interdisciplinary use case tackling multimodal content management.

4 Proposed Solution

4.1 Background

While researching the scientific visualization methods and means we found out, that there is a lack of high-level tools providing easy-to-use capabilities to integrate with third-party data sources. This was a motivation to create the adaptive multiplatform scientific visualization SciVi. The detailed description of this system and underlying integration mechanisms can be found in our previous work [19]. SciVi relies on ontology engineering methods, its behavior is completely governed by the set of ontologies describing the supported visualization techniques, data processing algorithms and data sources. In addition, SciVi provides the data flow diagram editor enabling the user to describe the visualization task in terms of data processors (nodes) and data links (arcs). The aforementioned capabilities can be easily extended by enrichment of corresponding ontologies. SciVi was designed to adapt to third-party data sources including software and hardware data generators (so-called solvers). In this regard, we were able to use it as a monitoring tool for lightweight electronic devices within IoT ecosystem [20]. In the present work we introduce SciVi-Middle, the smart system built upon SciVi. SciVi-Middle is dedicated to generalize the integration capabilities of SciVi, to enable it not only integrating with IoT devices, but also interconnecting them with third-party software. The main purpose of SciVi-Middle is to allow inexperienced users to build custom HMI for different systems using IoT technologies.

4.2 SciVi-Middle Functioning Principles

The high-level pipeline of using SciVi with IoT devices as solvers is shown in the Fig. 1. The *Adaptation Stage* catches the data from the IoT device and, if needed, converts them to the form suitable for SciVi internal mechanisms. For example, the conversion may involve parsing the network packages, etc. The *Data Filtering Stage* performs custom data processing that user can tune by data flow diagram editor. Being in fact optional, this stage allows different context-aware data transformation by semantic filters described in the SciVi knowledge base. For example, the filtering may reduce sensor noise of IoT device, etc. The *Rendering Stage* is responsible to synthesize and present the image according to the user's settings and data caught. This stage also provides UI to organize optional feedback for the IoT device. For example, this may be calibration functionality.

The usage of SciVi-Middle is depicted in the Fig. 2. As seen, in this case only the "middle" stages of the system's pipeline are utilized. The internal adaptation mechanisms of SciVi are applied as middleware to ensure needed connection of the hardware device to the third-party software. Also, the new *Control Emission Stage* is introduced that is responsible to steer the desired application and catch its feedback. The architecture of SciVi-Middle is shown in Fig. 3.

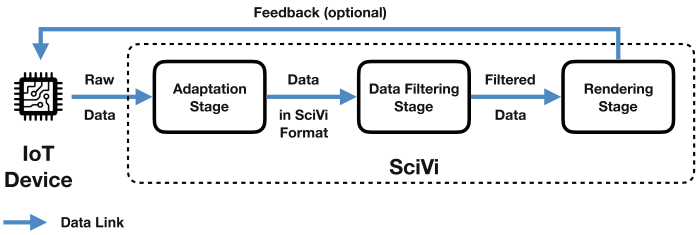

Fig. 1. Functioning pipeline of SciVi with IoT devices as solvers (introduced in the previous work).

The core of SciVi-Middle is its knowledge base that controls the behavior of all the internal modules. It contains 6 ontologies (stored in the standard OWL format) explained below. Let us denote as *Dev* the IoT device we want to use as hardware HMI, and denote as *App* the corresponding third-party application we want to steer by *Dev*.

Programming Languages I/O Structure Ontology (L) takes part in automating the process of integration SciVi-Middle with IoT device and third-party application. It is used if the source code of *Dev* firmware or source code of *App* is available to the user. In this case, the *Integration Module* of SciVi-Middle can automatically build the source code parser to extract the input and output

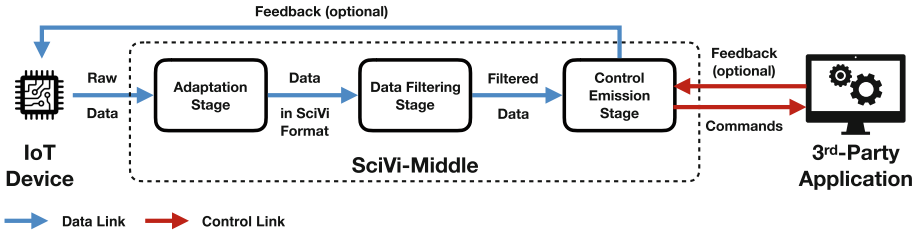

Fig. 2. Functioning pipeline of SciVi-Middle (introduced in the current work).

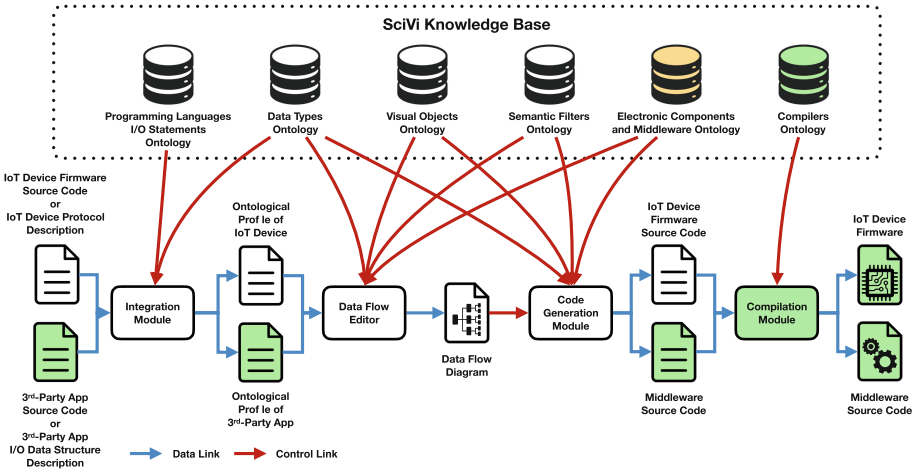

Fig. 3. SciVi-Middle Architecture (SciVi core is white, updated part is yellow, newly introduced parts are green). (Color figure online)

data structures description and thereby compose appropriate descriptions (so-called *Ontological Profiles*) of *Dev* and/or *App*. The corresponding mechanism is described in our previous work [17].

Data Types Ontology (T) describes all the supported data types, for example, numbers (both integer and real), vectors, matrices, quaternions, strings, etc. The description comprises the type name, the values domain, the type cast rules and some service information needed for internal mechanisms (for example, the color of corresponding typed arcs in data flow diagram). Almost every internal SciVi-Middle module requires knowledge stored in this ontology.

Visual Objects Ontology (U) is the most important one for scientific visualization system SciVi, because it describes the available visual objects, graphical scenes and rendering algorithms. But in SciVi-Middle it plays rather accessory role and may be used in some special cases like supplementary monitoring of the *Dev* activity while debugging, or some kind of additional graphical indication for the result IoT-based HMI.

Semantic Filters Ontology (F) provides knowledge about the supported data processing mechanisms. It may be extremely useful, because the raw data from *Dev* may be inappropriate to steer the *App* directly. For example, denoising of sensor data is often required, or, in more specific cases, some complex data transformation is indispensable like fusing the gyroscope and accelerometer measurement results into the representation of orientation. The principles of ontology-driven data filtering are described in details in [19].

Electronic Components and Middleware Ontology (C) is built upon the ontology of electronic components introduced in [20]. It is the most important knowledge base part in the SciVi-Middle. As it is clear from its name, it has two purposes. Firstly, it provides description of different programmable controllers (together with their built-in modules like analog-to-digital converter, etc.), chips, sensors, actuators and so on to give the user semantic overview of the *Dev*. Secondly, it provides description of different programming interfaces (parts of middleware), which may be used to access the *App*. Despite these purposes are quite different, both categories of knowledge are merged into the single ontology, because they require very similar interpretation.

First of all, the ontology C together with U and F governs the *Data Flow Editor*. As mentioned above, this module enables the user to compose visual representation of the task being solved in terms of data flow paradigm. Parsing the ontologies U, F and C the *Data Flow Editor* automatically builds the palette of available data filters, visual objects, graphical scenes, electronic components and middleware parts. User can choose the appropriate items from this palette and add them to the data flow diagram. These items are depicted as nodes having input and output sockets. After that, user can connect sockets building data links and thereby define the concrete processing algorithm. The usage of data flow paradigm to visually program the task solving algorithms within SciVi is described in [19].

Next, the ontologies U, F and C provide necessary knowledge for *Code Generation Module*. This module was first introduced in [20] as the tool generating firmware for IoT devices. In the present work it was enhanced and generalized to enable both firmware and middleware source code generation. This module traverses *Data Flow Diagram* (DFD) received from *Data Flow Editor*. Each node of DFD is linked to the corresponding concept in the ontology U, F or C. This concept is in turn connected to the set of concepts representing corresponding code snippets, function calls, library dependencies, etc. Assembling these building blocks, *Code Generation Module* composes the program source for the firmware and middleware.

It is worth noting, that both *Dev Firmware Source Code* and *Middleware Source Code* are actually optional. In some cases the firmware already exists and SciVi-Middle adapts to it, so no its modification (no generation) is required. The same is applied to the middleware: if the *App* provides necessary interface that can directly connect to the *Dev*, no middleware is created to minimize mediators. One more important point is that both firmware and middleware

source codes, if generated, are made "human-friendly": readable coding style is followed. If the user is skilled enough, he/she can make some changes in this code, for example, implement some additional functionality or just inspect the code in learning purposes.

Compilers Ontology (K) is first introduced in the present work and enables to connect compilers directly to SciVi-Middle to build the firmware/middleware from the generated sources. This simplifies the toolchain needed to get the IoT-based HMI creation task solved. The user can obtain not just source code, but the ready-to-run application. The ontology K describes makefile-templates for building the source and, in case of the firmware, deploying it to the device. The `make` utility is assumed to be used to automate the building process. The makefile is generated by the *Compilation Module* and afterwards make is invoked. As a result, SciVi-Middle provides fully functional toolchain to automate creation of all the corresponding software to organize custom HMI for the *App* based on the *Dev*.

4.3 Key Points of SciVi-Middle Usage

The SciVi-Middle smart system retains the multiplatform portability of SciVi it is based on. It has two versions: desktop application written in C++ using Qt (while some ontology parsing and code generation parts are written in Python) and web application written in Python (server side) and JavaScript (client side). This enables two usage scenarios: standalone application and SaaS (Software as a Service).

Creating the hardware tangible HMI for some application, user should first of all get the device. Our suggestions of hardware components are listed in the below subsection. Here are two variants possible: the device already exists, or user assembles it from scratch. In both cases the firmware for the device either already exists (as a source code or in a binary form), or should be generated. If no generation is required, SciVi-Middle needs ontological profile of the firmware. In case the source code is available, this profile can be created automatically by *Integration Module*. Otherwise, the user should describe the device communication protocol manually using high-level graphical UI provided by *Integration Module*.

The next step is to get the application, which should be controlled via hardware HMI. Again, two variants are possible: the user either has an access to the source code, or not. In case the source code is available, the ontological profile of the application is built automatically. Else, the user should describe input and (optionally) output data structures of the application, or, alternatively, skip this step relying on the built-in middleware components presented in the ontology C. The idea is that the ontology C describes some presets for different use cases, allowing, for example, emulate the mouse movements or certain keyboard buttons. If the required communication is that simple, no special adaptation to the given application is required. So, the ontological profile of the application is optional.

After that, the user should build the data flow diagram describing the actions of IoT-based HMI. For this task high-level visual editor is provided. On this step the user can find out, that there are not enough items in the data flow editor's palette, for example, some needed data filter is missing. In this case SciVi-Middle functionality can be extended by enrichment of the corresponding ontologies without core source code modification. It is worth noting, that the set of available filters, visual objects, electronic components and middleware parts can be personalized: if there are too many of them described in the ontologies U, F and C, some excerpt of these ontologies can be used.

Next step is code generation that takes part automatically according to the data flow diagram composed. On this stage the user can obtain the results and use them for further development, or, alternatively, can proceed to compilation phase. In the latter case, the binaries of the firmware and middleware are created. The user can obtain the ready-to-run middleware (if it was required on previous steps) and deploy the firmware (if necessary) right to the device. The only limitation is, that for now the deploying works for desktop SciVi-Middle only; in case of Web-based version the firmware as well as the middleware can be just downloaded in the binary form.

4.4 Hardware Recommendations

Despite the diversity of microcontrollers, which may be used to assemble IoT devices, now we suggest using ESP8266 with the WiFi module on board. This controller is fairly cheap yet powerful (running at 80 MHz clock frequency and providing 80 Kb RAM) and energy efficient (energy consumption is about 80 mA at 3.3 V). In particular, we propose using WeMos D1 mini – the compact and cheap circuit board containing ESP8266 microcontroller, needed service circuits (like, for example, power and reset circuits) and USB-to-UART interface. This board is very handy in terms of firmware uploading, because it has built-in USB interface. Also, it already contains all needed service circuits to ensure stable work of ESP8266. The ontological description of this microcontroller is already included in the Electronic Components ontology mentioned above.

To simplify the device assembly, we propose using *shields*. Shields are extension circuit boards carrying sensors, displays, plugs for other electrical components (including actuators), etc. Shields require neither soldering nor any special instruments to connect. If they are described in the ontology C, the programming of corresponding firmware become as easy as connecting the components together: the user just adds the related nodes to the data flow diagram and visually builds the data processing chain linking these nodes in an appropriate order.

5 Use Case

The project titled "Socio-Cognitive Modeling of Social Networks Users Verbal and Non-Verbal Behavior Based on Machine Learning and Geoinformation Technologies" aims to identify behavior patterns of Social Network Services (SNS)

users through a comprehensive multiparameter analysis of social, behavioral, psychological and linguistic parameters of the individuals. The information from a SNS user profile, such as gender, age, education, sphere of interests, communities, etc. is considered as social parameters. User preferences such as publications marked as favorite, posts, etc. are considered as behavioral. Psychological parameters are revealed as a result of a psychological survey based on BFI (Big Five Inventory) [16] with an adapted Russian-language version of BFI [22]. Linguistic parameters are found out through linguistic analysis of 19161 automatically collected replicas of 340 users who have passed a psychological survey. As a result of linguistic analysis, 163 categories related to deixis, modality, stylistics, graphic means, etc. were singled out (in particular, the list of language genres included 41 nominations). Main goal of the research is to discover the dependencies between language categories and personality types according to BFI methodology. To achieve this, different statistical methods are used. The research is conducted within information system Semograph [3] that provides automation the research work of applied linguists. The large volume of heterogeneous parameters and the revealed numerous dependencies between them pose the problem of scientific data visualization for analyzing the obtained results of a multiparameter SNS users description. These problems are tackled within SciVi.

First, SciVi provides graph-based interactive visualization of the results[2]. Secondly, to simplify semantic filtering SciVi enables special IoT-based HMI. In particular, SNS users' messages with spatial and temporal deictic semantics were considered as language categories: (1) here/there, (2) right/left, (3) up/down, (4) now/then (and also in past/future) and (5) the speed of movement/change (fast/slow). An additional parameter of the analysis was the representation of one or several selected categories in SNS users' texts. The analysis is very complex because there are 144 variants of filtering available (filtering is exposed by hierarchy of 6 scales, each one having 2 or 3 variants). The advanced visualization tools and appropriate filter management interface are demanded. This is why we suggest to use not only traditional slider-based scales to filter the data, but also gesture interface as an alternative. For this purpose, the 6 categories mentioned above are mapped to gestures as described in the Table 1. Combinations of different gestures are possible. They are interpreted as combinations of their semantics. In case of an open palm, speed of gesture is ignored. In case of a feast, speed is mapped to the corresponding speed semantics.

Table 1. Verbal-gestural correspondences (bold font: **gesture**; italic font: *meaning*).

		Vertical move		**Horizontal move**		**Depth move**	
		Down	**Up**	**Left**	**Right**	**Back**	**Forth**
Hand back	**Up**	*Down*	*Up*	*Left*	*Right*	*Here*	*There*
	Down	*Past*	*Future*	*Past*	*Future*	*Past*	*Future*

[2] https://graph.semograph.org/cgraph/psycho/index.html?lang=en.

To detect gestures, we assembled glove-style wearable device similar to the ones presented in [21] or [9]. The distinctive feature of our approach is to use ontology-based SciVi tools to automate the generation of firmware for the device and middleware to turn this device into HMI. The hand gesture interface enables to apply the filtering with the single gesture instead of searching for the needed options within 6 separated filtering scales. That is why the efficiency of visual analysis is increased: the single gesture takes approximately 1 s, while interacting 6 sliders takes about 11 s.

The Fig. 4 demonstrates two roles of SciVi: scientific visualizer and IoT-based HMI provider. SciViCGraph [20] visualizer is used to establish structures of dominant connections between variables for different states of the graph that represent informants' social parameters, for example, gender. Glove-style IoT device simplifies managing semantic filtering needed to interactive visual analysis of multiparameter SNS users description. That helped the researchers to prove the hypothesis of the impact of gender differences on the behavior of users with the same psychological traits.

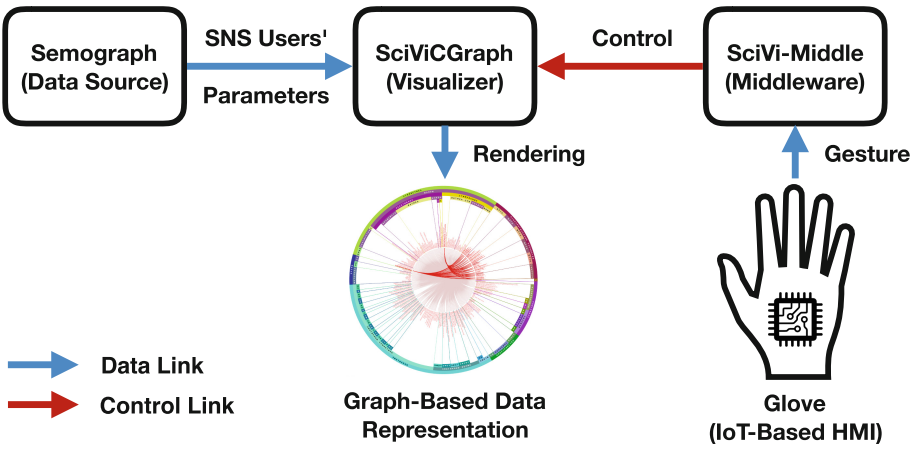

Fig. 4. Roles of SciVi in the research of SNS users' parameters.

6 Conclusion

In this paper, we present the unified high-level solution to build custom IoT-based HMI by means of a self-service smart system that has been developed to automate IoT devices firmware and middleware generation. On the one hand this system allows user to build a new custom device, program it and embed in the existing hardware and software ecosystem. On the other hand our solution is suitable to build the IoT ecosystem from scratch.

Nowadays such kind of solutions may be useful for solving special tasks, which require custom hardware controls to improve ergonomics compared to traditional mouse, keyboard, etc. In the near future, by evolving of virtual and augmented reality tools, the ability to have a hardware and software interface wizard will become more and more demanded. For example, gesture-based interfaces, tangible interfaces, wearable electronics and sensor network based interfaces are already indicated as today's rising trends [15].

The distinctive features of our approach are ontology-driven architecture that enables high-level adaptivity to specifics of different application domains and usage of cognitive graphics (initially based on scientific visualization engine) that enables building complex 2D and 3D monitoring and control graphical interface.

We tested our approach within the real-world interdisciplinary project named "Socio-Cognitive Modeling of Social Networks Users Verbal and Non-Verbal Behavior Based on Machine Learning and Geoinformation Technologies" and successfully improved visual analytics tools to discover correlations between the verbal behavior characteristics and psychological parameters of SNS users. The tools of visual analytics used in the study made it possible not only to streamline the connections between variables related to non-linguistic and linguistic aspects of personality traits and behavior but also to find new behavior patterns of SNS users.

In the future, we plan to create IoT-based healthcare monitoring systems and to adapt the tools described above to the specifics of voice recognition and rendering issues to tackle the problems related to speech input and output of biomedical data.

References

1. Noguera-Arnaldos, J.Á., Paredes-Valverde, M.A., Salas-Zárate, M.P., Rodríguez-García, M.Á., Valencia-García, R., Ochoa, J.L.: im4Things: an ontology-based natural language interface for controlling devices in the Internet of Things. In: Alor-Hernández, G., Valencia-García, R. (eds.) Current Trends on Knowledge-Based Systems. ISRL, vol. 120, pp. 3–22. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51905-0_1
2. Bajaj, G., Agarwal, R., Singh, P., Georgantas, N., Issarny, V.: A Study of Existing Ontologies in the IoT-Domain. arXiv (2017). https://arxiv.org/abs/1707.00112. Accessed 14 Feb 2019
3. Belousov, K., Erofeeva, E., Leshchenko, Y., Baranov, D.: "Semograph" information system as a framework for network-based science and education. In: Uskov, V.L., Howlett, R.J., Jain, L.C. (eds.) SEEL 2017. SIST, vol. 75, pp. 263–272. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-59451-4_26
4. Broll, G., Rukzio, E., Paolucci, M., Wagner, M., Schmidt, A., Hussmann, H.: Perci: pervasive service interaction with the Internet of Things. IEEE Internet Comput. **13**(6), 74–81 (2009). https://doi.org/10.1109/MIC.2009.120
5. Calderon, M., Delgadillo, S., Garcia-Macias, A.: A more human-centric Internet of Things with temporal and spatial context. Procedia Comput. Sci. **83**, 553–559 (2016). https://doi.org/10.1016/j.procs.2016.04.263

6. Farahzadi, A., Shams, P., Rezazadeh, J., Farahbakhsh, R.: Middleware technologies for cloud of things: a survey. Digit. Commun. Networks **4**, 176–188 (2018). https://doi.org/10.1016/j.dcan.2017.04.005
7. Ferrari, M.I., Aquino, P.T.: Human interaction and user interface design for IoT environments based on communicability. In: Amaba, B. (ed.) Advances in Human Factors, Software, and Systems Engineering. AISC, vol. 492. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41935-0_10
8. Ishii, H.: Tangible bits: beyond pixels. In: Proceedings of the 2nd International Conference on Tangible and Embedded Interaction, pp. XV–XXV (2008). https://doi.org/10.1145/1347390.1347392
9. Kumar Mummadi, C., Philips, F., Deep Verma, K., Kasireddy, S., Scholl, P., Kempfle, J., Van Laerhoven, K.: Real-time and embedded detection of hand gestures with an IMU-based glove. Informatics **5**, 28 (2018). https://doi.org/10.3390/informatics5020028
10. Lazarevich, K.: 5 keys to designing great UX for IoT products. IoT For All (2018). https://medium.com/iotforall/5-keys-to-designing-great-ux-for-iot-products-d10bda51842e. Accessed 14 Feb 2019
11. Luz, N., Pereira, C., Silva, N., Novais, P., Teixeira, A., Oliveira e Silva, M.: An ontology for human-machine computation workflow specification. In: Polycarpou, M., de Carvalho, A.C.P.L.F., Pan, J.-S., Woźniak, M., Quintian, H., Corchado, E. (eds.) HAIS 2014. LNCS (LNAI), vol. 8480, pp. 49–60. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07617-1_5
12. Nechaev, Y.I., Degtyarev, A.B., Boukhanovsky, A.V.: Cognitive computer graphics for information interpretation in real time intelligence systems. In: Sloot, P.M.A., Hoekstra, A.G., Tan, C.J.K., Dongarra, J.J. (eds.) ICCS 2002. LNCS, vol. 2329, pp. 683–692. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46043-8_69
13. Nuamah, J., Seong, Y.: Human machine interface in the Internet of Things (IoT). In: 12th System of Systems Engineering Conference, pp. 1–6 (2017). https://doi.org/10.1109/SYSOSE.2017.7994979
14. Petnik, J., Vanus, J.: Design of smart home implementation within IoT with natural language interface. IFAC-PapersOnLine **51**, 174–179 (2018). https://doi.org/10.1016/j.ifacol.2018.07.149
15. Poh, M.: 8 Next-Generation User Interface That Are (Almost) Here. Hongkiat (2018). https://www.hongkiat.com/blog/next-gen-user-interface/. Accessed 14 Feb 2019
16. Goldberg, L.R.: The development of markers for the big five factor structure. Psychol. Assess. **4**, 26–42 (1992). https://doi.org/10.1037/1040-3590.4.1.26
17. Ryabinin, K., Chuprina, S.: Development of ontology-based multiplatform adaptive scientific visualization system. J. Comput. Sci. **10**, 370–381 (2015). https://doi.org/10.1016/j.jocs.2015.03.003
18. Ryabinin, K., Chuprina, S.: Using scientific visualization tools to bridge the talent gap. Procedia Comput. Sci. **51**, 1734–1741 (2015). https://doi.org/10.1016/j.procs.2015.05.376
19. Ryabinin, K., Chuprina, S.: High-Level toolset for comprehensive visual data analysis and model validation. Procedia Comput. Sci. **108**, 2090–2099 (2017). https://doi.org/10.1016/j.procs.2017.05.050
20. Ryabinin, K., Chuprina, S., Kolesnik, M.: Calibration and monitoring of IoT devices by means of embedded scientific visualization tools. In: Shi, Y., et al. (eds.) ICCS 2018. LNCS, vol. 10861, pp. 655–668. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93701-4_52

21. Sawasaki, N., Ishihara, T., Mouri, M., Murase, Y., Masui, S., Nakamoto, H.: Front-end device technology for human centric IoT. Fujitsu Sci. Tech. J. **52**, 61–67 (2016)
22. Shchebetenko, S.: Reflexive characteristic adaptations explain sex differences in the big five: but not in neuroticism. Personality Individ. Differ. **111**, 153–156 (2017). https://doi.org/10.1016/j.paid.2017.02.013
23. Steinmetz, C., Rettberg, A., Ribeiro, F.G.C., Schroeder, G., Soares, M.S., Pereira, C.E.: Using ontology and standard middleware for integrating IoT based in the industry 4.0. IFAC-PapersOnLine **51**, 169–174 (2018). https://doi.org/10.1016/j.ifacol.2018.06.256
24. Wu, J., Grimsley, R., Jafari, R.: A robust user interface for IoT using context-aware bayesian fusion. In: IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks, pp. 126–131 (2018). https://doi.org/10.1109/BSN.2018.8329675