End-User Development of Voice User Interfaces based on Web content

Gonzalo Ripa [1], Manuel Torre[1], Sergio Firmenich[1,2], Gustavo Rossi[1,2]

[1] LIFIA, Facultad de Informática, Universidad Nacional de La Plata
[2] CONICET, Argentina
{sergio.firmenich,gustavo}@lifia.info.unlp.edu.ar

Abstract. Voice Assistants, and particularly the latest gadgets called smart speakers, allow end users to interact with applications by means of voice commands. As usual, end users are able to install applications (also called skills) that are available in repositories and fulfill multiple purposes. In this work we present an end-user environment to define skills for voice assistants based on the extraction of Web content and their organization into different voice navigation patterns. We describe the approach, the end-user development environment, and finally we present some case studies based on Alexa and Amazon Echo.

Keywords: Voice Assistant, End-User Development, Web content

1 Introduction

The World Wide Web has become the main information source and service platform. In parallel, speech recognition algorithms and technologies around it have experienced strong advances in the last ten years, reaching a broad consumption by part of end users. Nowadays, we can see new kinds of gadgets that allow the access to content and functionalities already offered in the Web, but under another user interaction mode based on voice interaction. This is the case of voice assistants in the form of smart speakers such as Amazon Echo or Google Home. As any other smart device, these gadgets allow users to install applications (also called skills) that offer specific information services [1]. Some smart speakers' applications are related to the command of smart devices, or even a smart devices mashups by the use of platforms for IoT such as IFTTT [2] or Node-Red [3]. However, other kind of smart speaker applications are more focused on reading and interacting with information and services already published by existing Web applications, such as reading news from a news portal, or asking for a product's price from Amazon.com.

Suddenly, a process is ongoing regarding how Web applications owners move the access to their content and services to these voice-based user interaction devices. For instance, the booking site Expedia.com offers an Amazon Echo Skill that lets end users search prices for accommodation and flights. A similar process happened when smart phones emerged, and Web applications started to deliver native mobile applications for them. Nevertheless, different to this case where users are able to visit the Web

site from the mobile Web browser, when a native application does not exist (in the case of smart speakers) there is not a generic way to access content and services not delivered by a native application or skill.

Web applications play a very relevant role in the users' daily life; we use them for reading news, to consume different services, for working, and even for interacting with smart devices in the Web of Things. In spite of the progress on Model Driven Engineering [4] and Multi-Modal User Interfaces [5], a large majority of Web sites are not developed with these technologies and specifications, therefore delivering device-specific applications (e.g. for providing voice access) is usually expensive.

This paper aims to fill the existing gap between the available smart speakers' applications and the preferred online Web services and contents that users consume on a daily basis by browsing the Web. We propose an end-user development environment by which end users may create their own extensions for voice assistants based on the abstraction and extraction of Web content and services that they are accustomed to use.

The paper is organized as follows. Section 2 presents a background on different aspects related to this approach. Section 3 introduces our approach and presents the rationale underlying our End-User Development environment, which is described in Section 4. Section 5 explains how the case studies used such as examples through the paper were developed for Amazon Echo. Finally, we give some conclusion and future works on Section 6.

2 Background and related works

This section presents a necessary background in different concepts and technologies related to our approach, before introducing it in Section 3. In this section, we also discuss about different works related to our intents.

2.1 Voice User Interfaces

Conversational agents and Virtual Assistants are not a new concept. Already on 1960, Licklider established the interest of "talking with computers" as one dimension to contemplate in human-computer interaction [6]. At that moment, speech production was more easily doable by electronic systems, but speech recognition had severe problems. Speech recognition algorithms evolved fast in the last decades, and we could appreciate research works for conversational agents (also known as conversational interfaces) almost twenty years ago [7] [8]. Nowadays, VUIs (Voice User Interface) are deployed among diverse kind of devices and interaction, such as smart phones, smart speakers, etc.

Although VUI started to be broadly used with their inclusion in smart phones (for instance, Siri in iPhone[1]) the emergence of gadgets such as Google Home or Amazon Echo are changing their daily use and pervasiveness. These smart speakers allow users to start some conversational interaction with a voice command expressed in natural language, such as "*Alexa, tell me the news*", in the case of Alexa service from

[1] Siri, https://www.apple.com/es/siri/, last accessed 3/14/2019

Amazon. Smart speakers are delivered with a set of base capabilities, for instance regarding the time and the weather, or other question-answer VUI that consumes vendors' services, play music, read news, etc.; new capabilities (also known as "skills") may be installed from repositories. In this way, other possible user tasks such as home automation, travel plan, online shopping, alternative information access, etc., may be added by installing third-party skills.

Currently, and just for analyzing one case, the Alexa Skill repository is organized in categories and offers more than 50.000 skills[2], almost doubling the number of skills available at the end of 2017 [9].

Table 1 lists the most relevant categories, the number of skills per category, an example of skill per category, and some samples of commands for this sample skill.

Table 1. Table captions should be placed above the tables.

Skill Category	Amout of Skills	Skill Example	Skill Sample commands
Business & Finance	over 1.000	Marketplace	*"Alexa, what's my Flash Briefing?", "Alexa, what's in the news?"*
Communication	over 1.000	Mastermind	*"Alexa, Ask Mastermind to text <someone>", "Alexa, ask Mastermind to ring my phone"*
Education & Reference	over 8.000	Couriosity	No direct commands, this skill offer aleatory content that end user may skip.
Games & Trivia	over 10.000	Twitch	*"Alexa, ask Twitch for followed channels", "Alexa, ask Twitch to play Monstercat"*
Lifestyle	over 7.000	Sleeptracker	*"Alexa ask Sleeptracker how I slept last night"*
Movies & TV	619	MDb's What's On TV Briefing	*"Alexa, what's my Flash Briefing?"*
Music & Audio	over 6.000	Connect Control for Spotify	*"Alexa, ask Connect Control to play on device 2"*
News	over 4.000	The Washington Post	*"Alexa, ask Washington Post for headlines"*
Shopping	153	Opening Times	*"Alexa ask Opening Times for Tesco Redruth Extra"*
Smart Home	over 1.000	Smart Life	*"Alexa, set hallway light to 50 percent"*
Sports	over 1.000	PGA Tour	*"Alexa, ask PGA TOUR for the leaderboard."*
Travel & Transportation	808	Madrid Transport	*"Alexa, open Madrid Transport" "incoming buses at 70"*
Weather	663	Temperature Now	*"Alexa, Temperature Now"*

[2] Alexa skill repository: https://www.amazon.com/alexa-skills/b?ie=UTF8&node=13727921011, accessed February 20th 2019.

As the reader may note, the range of services offered by skills is very broad, and it is also remarkable that for some categories there are more than 4.000 ones (e.g News). However, this is not surprising if we consider that for a same purpose we have a great variety of Web sources and services to achieve it. Consequently, it is straightforward to establish the possibility of creating new kind of skills using publicly available Web content and services (either in the form of Restful APIs or directly parsing and extracting the desired Web content).

Beyond this quantitative analysis, a recent study [10], shows that smart speakers users (the study was made with Google Home's users) use skills (in order of relevance, i.e., from most used to less used skills) related to Music, Information, Automation, Smalltalk, Alarm, Weather, Video, Time, Lists, Others. This lets us to do a more qualitative analysis related to the kind of skills users prefer. One more time, we can appreciate that for most of these categories there are several Web applications counterparts from where end users could read information or complete some business process using a device supporting normal Web browsing. Although some skills for automation of IoT devices could not satisfy this condition, it is clear that a very broad range of skills could be, or even are, based on existing Web contents and functionalities.

The possibility of creating personalized voice commands is also relevant and has already been studied in the context of multi-model user interfaces [11]. However, although this personalization system offers some kind of flexibility, end users are not able to manage the complete specification of VUI by their own.

In this paper, we investigate how Web contents can be extracted, processed and used (as responses) by VUI applications, particularly for smart speakers. Our approach, involves a set of tools that let end users without programming skills to be the ones that can create these VUI specifications using preferred custom web content sources.

2.2 Managing existing and third-party Web content

The idea of information extraction we use in this approach is similar to some techniques for Web Scraping [12]. Web scraping is the process of non-structured (or with some weak structure) data extraction, usually emulating the Web browsing activity. It is usually used to automate data extraction in order to obtain more information by processing it.

A common end-user driven technique for information extraction is the annotation of Web content. Some Web sites already tag their contents allowing other software artifacts (for instance a Web Browser plugin) to process those annotations and improve interaction with that structured content. A well-known approach for giving some meaning to Web data is Microformats [13]. Some approaches leverage the underlying meaning given by Microformats, detecting those objects present on the Web page and allowing users to interact with them in new ways. According to [14], only 5,64% among more than 40 million Web sites provide some kind of structured data (Microformats, Microdata, RDFa, etc.). This reality raises the importance of empowering users to add semantic structure when it is not available.

Several approaches let users adding structure to existing contents to ease the management of relevant information objects. For instance, Atomate it! [15] offers a reactive platform that could be set to the collected objects by means of rule definitions. Then

the user can be informed when something interesting (such as a new movie, or record) is added, edited or removed.

Some End-User Development approaches arose to empower users to solve their particular needs by themselves. For instance, MashMaker [16] allows extracting widgets with their properties, and later inserting these widgets in other Web pages n order to modify the application. Another work proposes the structuring and extraction of client-side data models to create personal Web sites that run purely on client-side, i.e. the end user's Web browser [17]. SearchAPI allows end users without programming skills to create search APIs by visually selecting the UI parts of Web applications search engines. In this way, the domain objects that an application offers can be searched by emulating the user interaction [18]. Similar approaches have arisen under the technique called Web augmentation, and it still a promising technology for end-user development [22]. However, to the best of our knowledge there are not end-user development approaches for developing entire VUI specifications by reusing existing Web content.

3 VUI Specification by end users and based on Web contents

In this section, we present our approach to define VUI by parsing Web content. First we present our approach in a glance, and Section 3.2 explains in details different underlying aspects.

3.1 The approach in a nutshell

The base of our end-user development approach for VUI is three-fold:

1. A mechanism for allowing users to select and define Web content blocks. For this purpose, we use Web content annotation and definition by means of visual tools and simple configuration. In the remaining of this paper, we call them content blocks.
2. A way to specify how these content blocks should be used in a VUI and how this VUI must behave. We found that flowcharts are a suitable to model VUI behavior structure; nodes represent a specific content block and connections represent how these contents must be organized and read.
3. An interpreter that processes a VUI specification, and obtain the Web pages' DOM dynamically, then provides them and finally give the response to the user with the extracted text content.

Our main idea is that end users can design flowcharts using the content blocks they previously created. The process starts by defining content block, which they can be used later in the flow editor to compose the VUI, to finally use these specifications from a native application on the voice-enabled smart device. Figure 1 depicts an overview of our idea, where the VUI specification (based on a flowchart) includes the content block A (from the Web Page A), and two other content blocks from Web Page B, one content block named B (that corresponds to a collection of related Web contents) and other named C. Imagine that the user wants to create a VUI for news based on the The New York Times' Web site. This user wants to include some elements from the site´s home (Blocks 1, 2, 3, 4 and 5 from the image 2.a). These content blocks would be extracted

individually, such as the elements A and C in the generic mockup example from Figure 1. However, this user also wants all the news for a specific news section from The New York Times (such as the highlighted one in the right image in Figure 2); our approach allows defining a set of siblings elements to compose a content block, which would behave as the B elements in Figure 1.

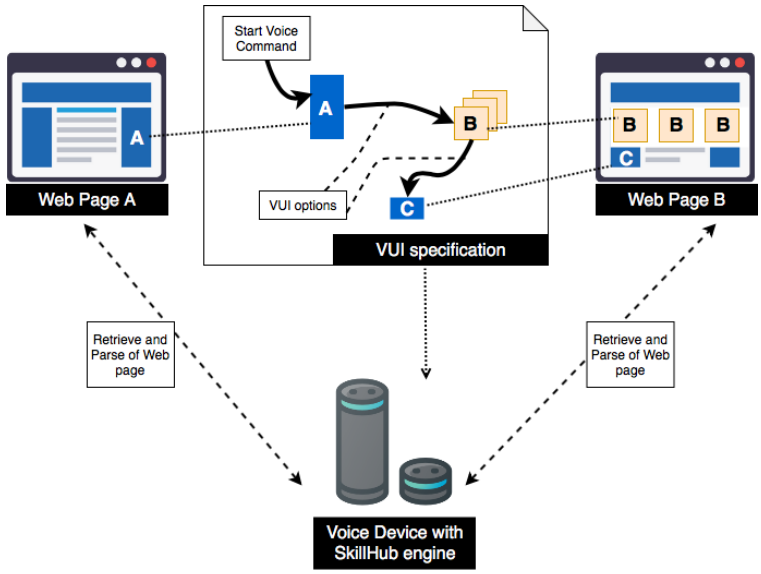


Fig. 1. Our VUI interpreter processes VUI specifications that are flows defining how parts of Web pages must be read in front of voice interaction.

3.2 Rationale: from web user interfaces to voice interfaces

In this section we present the four dimensions that defines our EUD environment for VUI.

Definition of content blocks

We foresee two ways to define content blocks. One of them is based on the individual selection of each useful part, such as Figure 2.a shows. In this case, the user must select the UI element (a DOM element) to create it's corresponding content block. The other way is to contemplate a set of UI elements as a whole content block that includes a collection of elements, such as Figure 2.b depicts.

Usually, Web applications expose in their UI a representation of domain objects such as news, products, articles, etc. This means that on the Web client-side a user could recreate a simple domain model based on the attributes presented in the UI. For instance, for the news presented in Figure 2.a, the attributes tittle, summary, date and author could be obtained from the UI. The same happens if we look for products in Amazon, whose UI presents name of the product, price, description, etc. The annotation process by which a content block is defined may consider this semantic specification, or be simpler and more direct and just consider the whole DOM element as a content

block. In this last case, by parsing the target DOM element it is possible to decompose it to detect different parts relevant to the VUI (anchors, text, etc.) automatically.

Another important aspect is whether the content block is navigable or not. It is very common in a Web site to present excerpts of information for a given item and offering to navigate to the specific Web page corresponding to that item by clicking a link; this is the case of the news presented in Figure 2.a. This navigation option to obtain further information about a content block will be also considered in our approach. Finally, content blocks must be categorized in order to allow flexibility when the VUI behavior is defined. In this way, voice commands such as "ask for ***main news***", "ask for ***weather information***", etc., can be defined.

(a) Different parts of the same Web page (b) Sibling elements from the same Web page

Fig. 2. Web content blocks: (a) unique items selection (b) sibling items selection.

A sample skill could be designed for using the content blocks from Figure 2.b, which is based on a set of siblings element sharing a topic. This skill could read the user the "Topics news" when he pronounces these work as a voice command. The skill may respond y reading the title of the first news, and ask to the end user if he wants to listen more about it or just to continue with the following:

- **User's Command voice:** "Topic news"
- **Amazon Echo:** "women's hockey rivals prepare for the olympics by playing each other again and again, ***do you want to listen more about this news?***" (this answer first tell the first news from the topic, and then ask to the user if he wants to listen more).
- **User's Command voice:** "yes".
- **Amazon Echo:** "BOSTON — Three days after the United States women's hockey team lost to Canada, 5-1, in an exhibition game here on Oct. 25, USA Hockey unexpectedly added Cayla Barnes, an 18-year-old freshman at Boston College, to its roster. ***Do you want to continue listening more about this news?***"

(a) News' Web page (b) Search results in a news portal

Fig. 3. Web content blocks: (a) items details (b) search result items.

Access to content blocks

The Web is based on the concept of navigation among resources accessible by a universal resource location (URL). With this in mind, a specific Web site content can be retrieved if the URL is known. When facing a dynamic URL that changes in relation to published content or user session, navigation is another strategy to reach the Web content, given the Web site's home. However, when there are large sets of data in a Web site, search engines become essential to reach relevant information items. With this in mind, our approach considers three ways to access Web content:

- Direct Access: given an URL (which can be static or based on an API-based URL that allows to change some parameters values), it is possible to retrieve the Web page. This method is useful for getting the current state of a Web site that offers frequently updated information such as news, weather, etc. Figure 2, either (a) or (b), could be accessed in this manner.
- Navigation: when the desired content cannot be access by a predefined URL, navigating through the links that come with the Web page retrieved is possible. Navigation is also important for retrieving more information for a content block. For instance, reading the details for a main news may imply to follow the link that allows end users to navigate from the Web site's home to the specific news' web page, whose URL could hardly be known beforehand. Imagine that the user wants to know more about the news represented by the Element 3, in Figure 2.a. Then, a Web page similar to the presented in Figure 3.a would be retrieved, where this news is presented. In cases like this, navigation is used to reach the target content block.

- Search: in cases where the VUI requires querying a Web application for specific information, the automated use of search engines could be used. For instance, if only elements related to a specific domain are required (for instance, news related to "Venezuela", as Figure 3.b shows), then to emulate how the user would search it on the Web application could be useful. This method for Web content access is also relevant in the case of e-commerce, accommodation and flights Web sites, etc.

Despite how the target Web page that contains the desired content is retrieved, once it is obtained it is necessary to parse it and extract the content block. For this goal, each content block has a template extractor, which is defined by end users using visual selection and annotation tools. These annotations belong to the VUI definition and (among other aspects) contains the XPath expressions to extract a specific information element given a retrieved Web page.

Order in which the same voice command reads several content blocks

As we mentioned, this approach proposes using flow diagrams to arrange how content blocks are disposed in the VUI, because to respond a voice command, a sequence of content blocks will be read as a response. Once the content blocks are defined, it is important to define an order in which they will be read and under which voice interactions. For example, for the content blocks in Figure 2.a, the voice command could be "Read today's news" and the order in which the news must be read may be (Block1, Block2, Block3, Block4, Block5), in which each block number corresponds to the numbers in figure 2.a, or any other the end user defines.

Configuration of the VUI's behavior

As we said, in our approach a flow diagram defines the main structure of VUI responses. Besides the established order, different aspects of the VUI behavior must be defined:

1. How to read a content block: when the user says a command, the VUI will respond using one or a set of content blocks. However, which parts or properties to read for each of these content blocks may be different in distinct use scenarios. For instance, the user may be interested on reading just the main title or the complete content block for a news, or in a generic way, a specific collection of the semantic properties defined for that content block. Furthermore, if the block contains a navigable element, then it would be possible to offer deeper information that could be extracted by retrieving the Web page defined in the content block's link, etc.
2. How to continue to the next content block: beyond how to read a specific content block, when it is finished, there are different possibilities to continue with other related ones. This is part of the definition of the VUI behavior, in which the end user must be able to define among different options: read following block without asking, read following block without asking but pronouncing a predefining text, ask to the user if s/he want to continue, etc.

With these two aspects, we aim to give support to behavior variability for the proposed VUI. However, since it can be tedious and error prone to define each of these aspects for each element in the VUI flow (contemplating both nodes and arrows), we propose to use as a default option some VUI patterns. A pattern template defines the

transversal behavior to manage content blocks (1) and the transitions among them (2). However, to allow end users to customize their VUI and better support variability, they may change the pattern-based behavior for both a particular content block and for a transition to another content block. So far in our approach, a VUI pattern requires JavaScript programming skills to be defined, basically these are state machines for define the conversational behavior.

4 SkillMaker: A Web browser-based environment for VUI Specification

In this section, we present SkillMaker, our EUD environment, through an example. We first present the tool for defining extraction templates for content blocks, and later the editor of VUI based on flow diagrams using these content blocks. The whole environment is deployed as a Web browser extension (in particular as a Google Chrome plug-in).

4.1 Contents Blocks definition

We use content annotation as the method to define a content block [19]. The process starts when the user decides to define a content block for the current Web site, which is done by clicking the main button of the SkillMaker Web extension – point (1) in Figure 4 –. The result of this is that the DOM elements are highlighted when the pointer is over them. When the user chooses a specific DOM element, he may drag and drop it – point (2) in Figure 4 –into the extraction template definition box – point (3) in Figure 4 –.

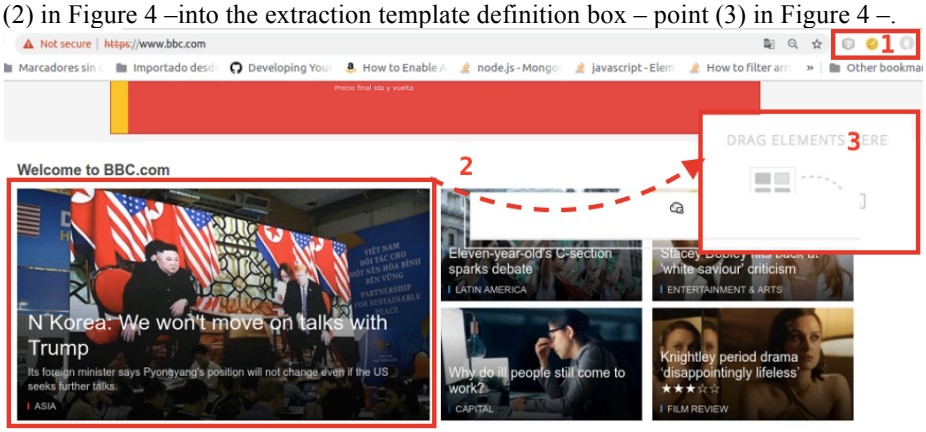


Figure 4. Web content selection for defining a content block

Once a DOM element is selected, the annotation process starts by adding the semantic of the sub-DOM elements, such as Figure 5 shows. A detailed view is shown in Figure 6.a, where the confirmation for the title property can be seen.

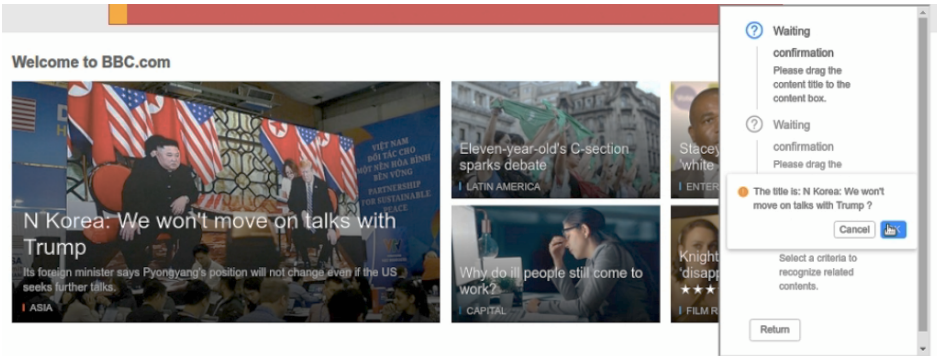
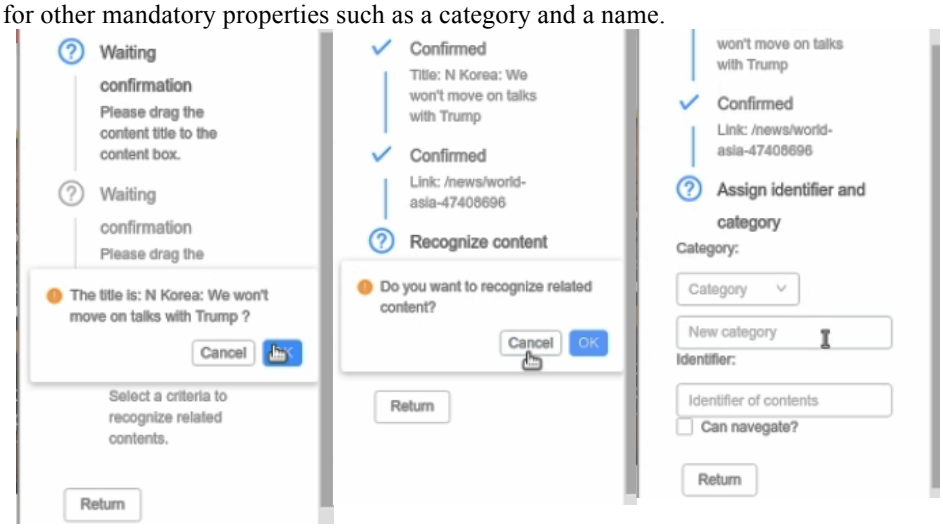


Figure 5. Definition of a Web content block

The process continues (Figure 6.b) by asking the end user if related content (basically sibling elements) should be considered, in order to support content blocks such as the one in Figure 2.a. Figure 6.b also shows that the tool detects navigation links for the selected DOM element automatically. In this way, the end user may consider this navigation (see "Can navigate?" checkbox in Figure 6.c). Figure 6.c shows the edition form for other mandatory properties such as a category and a name.



a. Semantic attributes edition b. Related content edition c. Final block edition

Figure 6. Edition views of different steps of the block definition process

When the end user confirms the creation of this content block, the tool stores it and offers to open the flow editor, which is explained next. Otherwise, the user may continue with the creation of content blocks for the same Web page or any other.

4.2 VUI definition and deployment through examples

The flow editor, also deployed in the same Web extension, has access to the content blocks defined by the end user, which can be dragged and dropped into the diagram editor's canvas, as Figure 7 shows. In this example, the content block selected is the one representing all siblings elements from Figure 2.b.

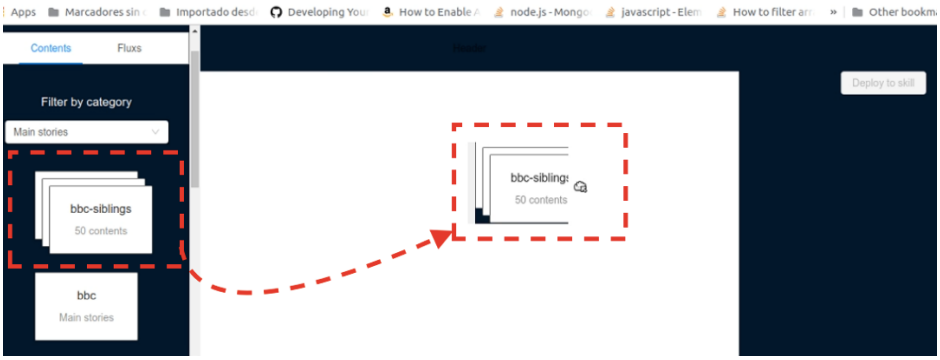


Figure 7. Flow for read a set of sibling content blocks

After adding several more blocks into the canvas, and also some links among them, the flow looks like the one in Figure 8. In these case, several content blocks representing the main news from different portals.

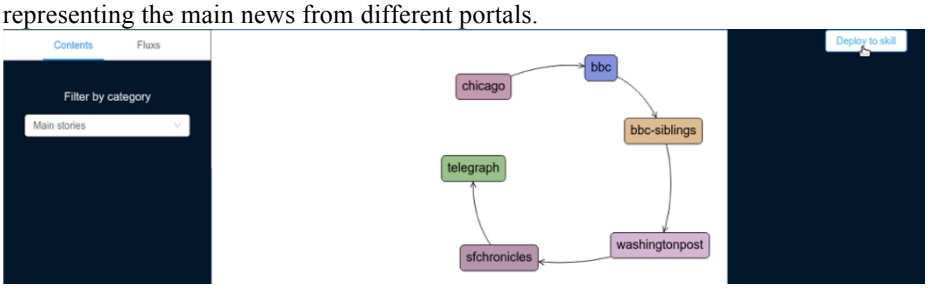


Figure 8. A flow for reading the main news from each news portal

When the user clicks on the "Deploy to skill" button, the editor opens a modal window (Figure 9) asking the user: 1) a name for the skill which will be used as a voice command to be answered with this flow, 2) the content reading pattern to use in the corresponding response. This last option relates to the Configuration of VUI behavior issue presented in previous section.

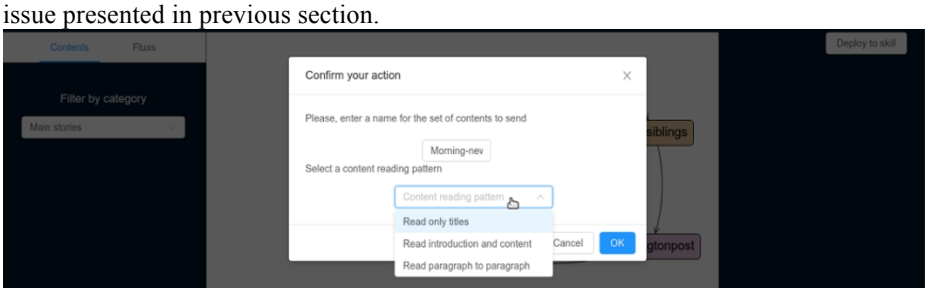
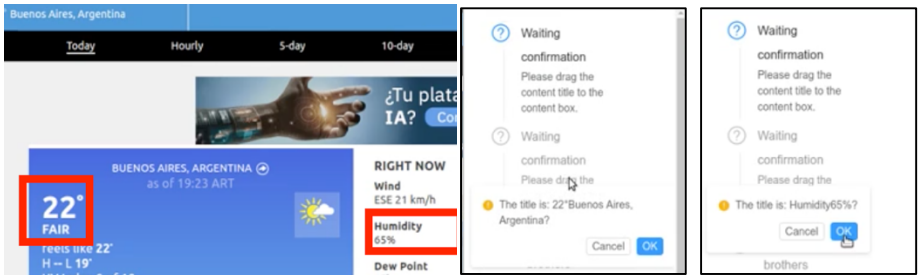


Figure 9. Editing skill name and content reading pattern.

The skill designed in the flow from Figure 8 uses the "Read only titles" pattern. In this example, 6 content blocks "main news" were defned for 6 news portals (Chicago, BBC, Washington Post, Telegraph, sfchronicles). A sample interaction with this VUI is the following:

- **User's Command voice:** "Main News headlines"
- **Amazon echo:** "N Korea We won't move on talks with Trump, ***next news…***"

A second case study was based on existing and frequently used skills, such as those in the weather domain. This skill is defined to answer with the temperature and the humidity from Buenos Aires city, when the user says "Weather in Buenos Aires".



a. Web page highlighting Web contents b. Temperature block c. Humidity block

Figure 10. Content blocks definition for the Weather skill

The skill was built using information from weather.com. Figure 10 shows the blocks definition process. Two content blocks are defined, one for temperature and a second one for humidity. Figure 11 shows the corresponding flow. Since the interesting content were detected as titles, we used the "Read only titles" pattern. A possible conversation excerpt is the following:

- **User's Command voice:** "Buenos Aires weather"
- **Amazon Echo:** "22, humidity 65%"

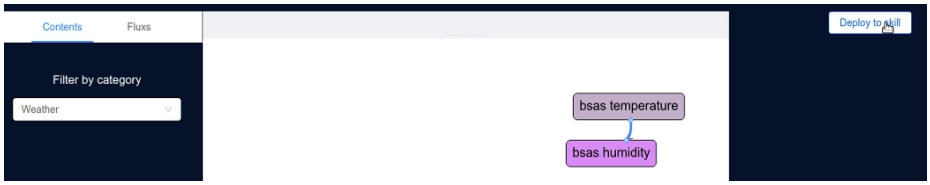


Figure 11. VUI definition for the Weather skill

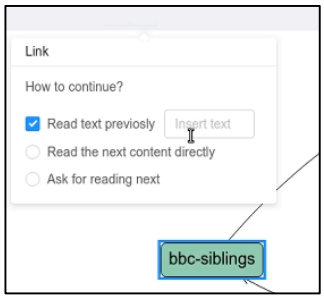
End-User Development concerns

We mentioned before that variability is a key aspect in EUD environments; therefore, the general rules to read content blocks and the links used by the pattern may be replaced by manually editing how to read these elements. For the case of the content blocks, Figure 12.a shows the available options, which range from read everything, read only the titles property, and whether the VUI must ask the user before reading the content block or not.

Figure 12.b shows the configuration for the links. In this case, the available options are to read a particular text before starting with the content block, read the block directly without asking, ask for reading the next content block (which has the same impact that set it up for the following block's configuration).

It is interesting to mention that these options came from the analysis of several examples that were useful to define the expressivity of the approach (further details are not included for the sake of space). However, new kinds of controls can be easily programmed.

a. Editing VUI options for a Node



b. Editing VUI options for a Link

Figure 12. Editing VUI options for replacing patterns rules

Our environment takes into account particularly the debugging concern described by Ko. [20] in the context of end-user software engineering approaches. This is because our approach is based on third-party Web content. The extraction template defined with our environment have references to DOM elements expressed in xPath. If a Web page changes its underlying DOM structure, these xPaths expressions may not work anymore. Although there are ways to make them more robust [21], it still is possible that a substantial change in the target Web page's DOM breaks the references. In this sense, the environment provides visual feedback in the editor when a particular content block seems to have a broken reference, as Figure 13 shows; these blocks have a red background in the menu. By clicking on one of them, the procedure for defining an extraction template starts in the corresponding Web page. If the corresponding Web page is not found, the user may define a new content block in any other Web site and store it with the same name.

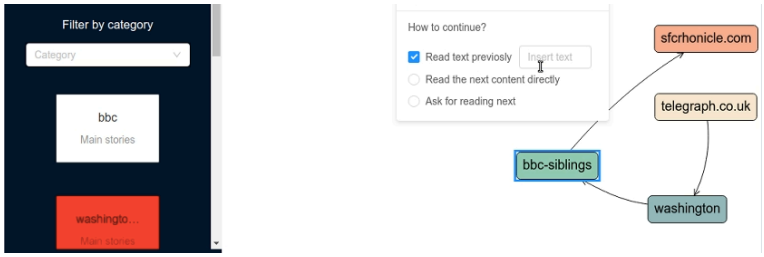


Figure 13. Content block with missing or broken DOM references

5 Case Studies

Al the examples presented in the paper were used for defining case studies achieving real interaction. We have carried on these case studies using Amazon Echo, which is based on Alexa. For these case studies, we create SkillHub, an Amazon Echo skill that

includes our Javascript-based interpreter for the VUI created with SkillMaker. SkillHub and SkillMaker are synchronized so when a new VUI flow is created, it is automatically available on the Amazon Echo. SkillHub also allows end users to ask for the content of a specific category of block. For instance, if for some block the user defines the category "temperature", SkillHub will read the current content from the Web page. VUI specifications made with SkillMaker are stored in JSON format and interpreted in this format by SkillHub. This Amazon Echo skill allows us to prove the VUI created with SkillMaker in a real scenario. We have developed the "Main news" presented in Figure 8 and also the "Weather skill" shown in Figure 11. The process for defining these cases studies is quick and trivial.

6 Conclusion and future works

VUI are being increasingly used to allow communication with smart devices. These devices usually allow end users to install third-party skills to support new behaviors. In this paper, we presented an end-user development approach to allow end users to create their own VUI-based skills for using their preferred Web sources for information and services. The creation of VUI based on Web content could be an interesting way for users to gain more control while interacting with devices.

We discussed the rationale and mechanics to transform Web content into VUI, which consists in extracting content blocks and arranging them in flow diagrams that will be interpreted for answering a voice command. We also showed our EUD environment, including the extraction template for content blocks and SkillMaker, our EUD tool to create VUI based on content blocks. The development time was very short; it just took some minutes for defining content blocks and the VUI for using them in SkillMaker. As a proof of concept we developed SkillHub, an Amazon Echo skill implementing our approach. We used SkillHub to interact with the voice commands defined using SkillMaker.

References

1. Zhang, N., Mi, X., Feng, X., Wang, X., Tian, Y., & Qian, F. (2018). Understanding and mitigating the security risks of voice-controlled third-party skills on amazon alexa and google home. *arXiv preprint arXiv:1805.01525*.
2. IFTTT and Amazon Alexa, https://ifttt.com/amazon_alexa, last accessed 2019/3/13.
3. Rajalakshmi, A., & Shahnasser, H. (2017, September). Internet of Things using Node-Red and alexa. In *2017 17th International Symposium on Communications and Information Technologies (ISCIT)* (pp. 1-4). IEEE.
4. Brambilla, M., Cabot, J., & Wimmer, M. (2017). Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering*, *3*(1), 1-207.
5. Elouali, N., Rouillard, J., Le Pallec, X., & Tarby, J. C. (2013). Multimodal interaction: a survey from model driven engineering and mobile perspectives. *Journal on Multimodal User Interfaces*, *7*(4), 351-370.
6. Licklider, J. C. R. (1960). Man-computer symbiosis. *IRE transactions on human factors in electronics*, (1), 4-11.

7. Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjálmsson, H., & Yan, H. (1999, May). Embodiment in conversational interfaces: Rea. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 520-527). ACM.
8. Kadous, M. W., & Sammut, C. (2004, August). InCa: A mobile conversational agent. In *Pacific Rim International Conference on Artificial Intelligence* (pp. 644-653). Springer, Berlin, Heidelberg.
9. White, R. W. (2018). Skill discovery in virtual assistants. *Communications of the ACM*, *61*(11), 106-113.
10. Bentley, F., Luvogt, C., Silverman, M., Wirasinghe, R., White, B., & Lottrjdge, D. (2018). Understanding the Long-Term Use of Smart Speaker Assistants. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, *2*(3), 91.
11. Kurniawati, E., Celetto, L., Capovilla, N., & George, S. (2012, January). Personalized voice command systems in multi modal user interface. In *2012 IEEE International Conference on Emerging Signal Processing Applications* (pp. 45-47). IEEE.
12. Ferrara, E., De Meo, P., Fiumara, G., & Baumgartner, R. (2014). Web data extraction, applications and techniques: A survey. Knowledge-Based Systems, 70, 301-323.
13. Khare, R., & Çelik, T. (2006, May). Microformats: a pragmatic path to the semantic web. In Proceedings of the 15th international conference on WWW (pp. 865-866). ACM.
14. Bizer, C., Eckert, K., Meusel, R., Mühleisen, H., Schuhmacher, M., & Völker, J. (2013). Deployment of rdfa, microdata, and microformats on the web–a quantitative analysis. In The Semantic Web–ISWC 2013 (pp. 17-32). Springer Berlin Heidelberg.
15. Van Kleek, M., Moore, B., Karger, D. R., & André, P. (2010, April). Atomate it! end-user context-sensitive automation using heterogeneous information sources on the web. In Proceedings of the 19th international conference on World wide web (pp. 951-960). ACM.
16. Ennals, R., Garofalakis, M. Mashmaker : Mashups for the masses (demo paper). In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD'2007).
17. Firmenich, S., Bosetti, G., Rossi, G., & Winckler, M. (2017, May). End-user software engineering for the personal web. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (pp. 216-218). IEEE.
18. Bosetti, G., Firmenich, S., Fernandez, A., Winckler, M., & Rossi, G. (2017, June). From Search Engines to Augmented Search Services: An End-User Development Approach. In *International Conference on Web Engineering* (pp. 115-133). Springer, Cham.
19. Firmenich, S., Bosetti, G., Rossi, G., Winckler, M., & Barbieri, T. (2016, June). Abstracting and structuring web contents for supporting personal web experiences. In *International Conference on Web Engineering* (pp. 77-95). Springer, Cham.
20. Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., ... & Rosson, M. B. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys (CSUR)*, *43*(3), 21.
21. Aldalur, I., & Diaz, O. (2017, June). Addressing web locator fragility: a case for browser extensions. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems* (pp. 45-50). ACM.
22. Aldalur, I., Winckler, M., Díaz, O., & Palanque, P. (2017). Web Augmentation as a Promising Technology for End User Development. In *New Perspectives in End-User Development*(pp. 433-459). Springer, Cham.