IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 15, 2018, accepted November 21, 2018, date of current version January 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2883500

New Approach for Conversational Agent Definition by Non-Programmers: A Visual Domain-Specific Language

LUIS RODRÍGUEZ-GIL[1], JAVIER GARCÍA-ZUBIA[2], (Senior Member, IEEE), PABLO ORDUÑA[1], (Member, IEEE), AITOR VILLAR-MARTINEZ[3], AND DIEGO LÓPEZ-DE-IPIÑA[3]

[1]LabsLand-C. Gordóniz, 48002 Bilbao, Spain
[2]Faculty of Engineering, University of Deusto, 48007 Bilbao, Spain
[3]DeustoTech-Deusto Foundation, 48007 Bilbao, Spain

Corresponding author: Luis Rodríguez-Gil (luis@labsland.com)

ABSTRACT Intelligent tutors and conversational agents (CAs) have proven to be useful learning tools. They have potential not only as stand-alone devices but also as integrable components to enrich and complement other educational resources. For this, new authoring approaches and platforms are required. They should be accessible to non-programmers (such as most teachers) and they should be integrable into current web-based educational platforms. This paper proposes a new approach to define such agents through a visual domain-specific language based on Google Blockly (a scratch-like language). It also develops a web-based integrable authoring platform to serve as a prototype, describing the requirements and architecture. To evaluate whether this novel approach is effective, a multi-stage experiment was conducted. First, participants learned to use the prototype authoring platform through an interactive tutorial. Second, they created a CA with a specific domain model. Times and performance were measured. Finally, they answered a standardized usability questionnaire (UMUX) and a purpose-specific survey. Results show that participants were able to learn to use the domain-specific language in a short time. Moreover, the purpose-specific survey indicates that their perception of the approach (and its potential) is positive. The standardized questionnaire indicates that even in its prototype stage, its usability is satisfactory.

INDEX TERMS Visual programming languages, customizable systems, conversational agents, intelligent tutoring systems, online learning, online labs.

I. INTRODUCTION

Conversational agents (CAs) and intelligent tutoring systems (ITSs) have existed for decades [1], [2]. Those two types of systems have different characteristics. The former focus on representing a human, often featuring a virtual body. The latter provide a learning environment, are often task-based, and try to resemble human tutoring to leverage its advantages [3]. They have aspects in common, and some ITSs include or are based on CAs [1].

Throughout the years, significant research efforts have been dedicated to these systems. Traditionally, one of the aspects that have drawn more attention is their natural language processing (NLP) capabilities. For example, this was the focus of Eliza [4], one of the first and most influential CAs. It relied on a simple pattern-matching algorithm to create the illusion of intelligence. Other researchers have experimented with Embodied CAs [5], [6], which aim to increase believability by also having a virtual animated body. Attention has also been directed towards approaches for building systems with comprehensive domain models, and powerful authoring tools [7], [8].

CAs have many potential applications. One of those is education (e.g., tutoring or question answering [9]). They also have applications in other fields: they can provide customer service, information, or act as a virtual companion or website tour guide [10]. Interest in them keeps growing. However, they can be expensive to create; and domain experts do not always have programming experience. Effective authoring tools are important to reduce those obstacles [7], [11].

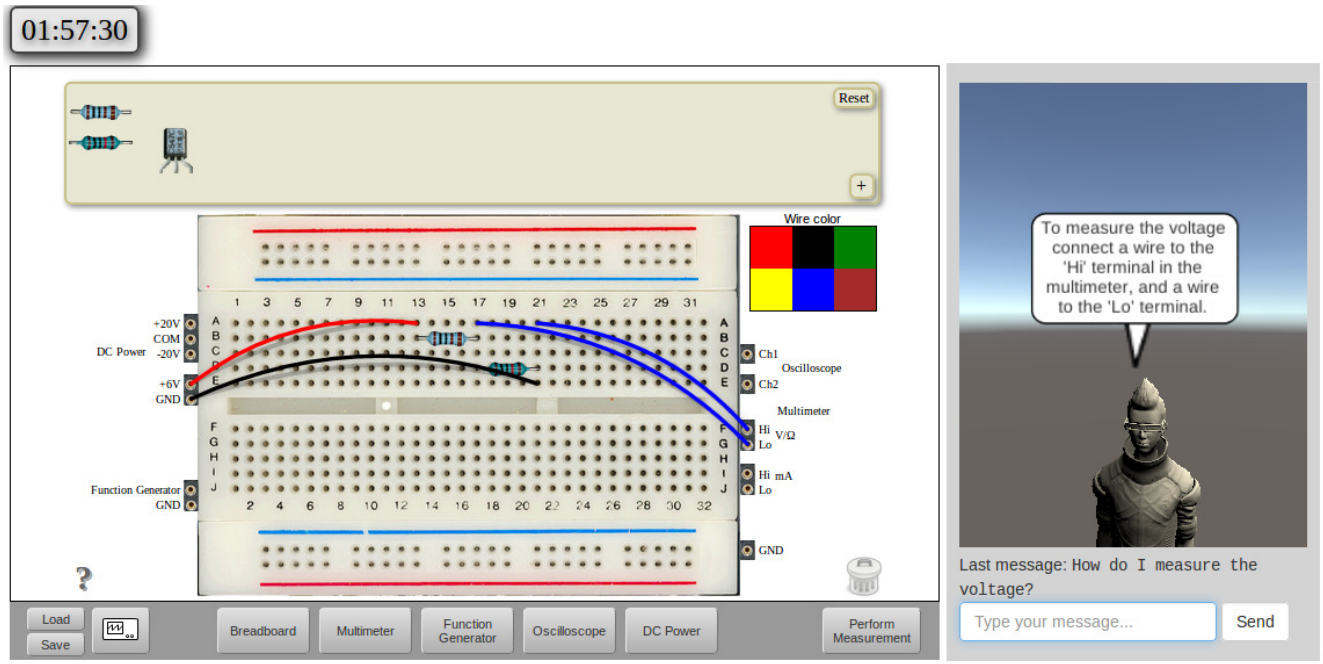

FIGURE 1. Embedding an agent designed by a non-programmer into a VISIR remote lab.

In this work we describe a novel approach for the creation of web-based CAs and we present the architecture of a web-based authoring platform that uses it. Although the main example use case are remote lab guides, the approach and agents are intended to be general-purpose. Its main goal is to be used by non-programmers. The approach relies on a simple Domain-Specific Language (DSL) based on Google Blockly [12] —a library for creating visual block programming languages that has been successfully used by non-programmers, including children [13]— and are designed to create domain-independent agents that may be integrated into external platforms. The main novelty in this approach is the use of a non-programmer oriented visual DSL for the definition of the agent behavior.

Though this work can be applied for the authoring of many types of CA, the main use case that will be described throughout this work is the creation of web-embeddable guides and educational agents. Throughout the last years, the use of educational technologies such as online courses, MOOCs [14], and online laboratories [15]–[17] has increased, and the trend is expected to continue. Some research works and large initiatives, such as the European project Go-Lab, suggest that for the wide and successful adoption of these tools it is important that teachers are able to provide customized experiences for their students [18]–[22]. In this context, educational agents that are customizable by non-programmers and easily embedded into educational contents (e.g., online exercises or online labs) to provide guidance may add significant value. This is the intended main practical application of this scheme. Figure 1 illustrates this with an example: a non-programmer-defined CA accompanies an instance of the VISIR [23] remote lab, in order to provide specific guidance for a practice.

This contribution focuses on providing the means for the general non-programming public (such as most students, or most teachers) to create and customize their own agents. Agents with improved NLP capabilities, natural conversations or high believability would be supported by the extensible nature of the proposed scheme, but are beyond the scope of this contribution. Non-programming users define the knowledge domain of the agent through a Google Blockly based DSL. This approach and these agents are intended to be most useful for use cases where the number of rules is not large, the user input is scope-specific and a well-defined conversational output is preferred. The authoring platform that is proposed in this work is designed to create relatively simplistic agents. However, Google Blockly has been used as a programming language, and it is designed to be extensible. As such, future systems that use this approach could provide arbitrarily advanced functionality through additional custom *blocks*, which in Google Blockly are akin to functions in traditional programming languages.

This paper is organized as follows: Section II describes in more detail the relevant state of the art on customizable CAs, authoring tools, and languages oriented for non-programmers. Section III analyzes the requirements for the proposed approach and the authoring platform. Section IV presents the proposed approach. Section V details the platform architecture. Section VI describes the methodology of the study that has been conducted to evaluate how easily such an authoring approach can be learned, how usable it is, and what the user perception is. Section VII describes the

results of the study, organized around the research questions. Section VIII discusses these results. Section IX summarizes the conclusions. Finally, Section X proposes future lines of work and potential applications.

A. PURPOSE AND RESEARCH QUESTIONS

The purpose of this work is to propose a novel approach to enable non-programmers to define custom CAs through the use of a Visual DSL; to create a prototype web-based authoring platform that implements that approach and that satisfies real-world technical requirements; and to answer the following research questions:

1) How easy to learn is the proposed approach? Can non-programmers learn it in a reasonable time?
2) How usable is the proposed approach?
3) Is the proposed approach perceived as valuable and intuitive?

II. RELATED WORK

A. AGENT AUTHORING TOOLS AND NON-PROGRAMMERS

Creating agents can take significant time and effort. Authoring tools can be leveraged to create agents faster and more efficiently, reducing the amount of time and resources required [8], [24]. Research efforts are dedicated towards their development and improvement. For example, research is being conducted on authoring tools for collaborative ITS and CA environments [8], [25], [26], on integrating ITSs into serious games [27], on developing authoring models such as Authoring by Tutoring [28], on making mobile authoring tools [29], and on making authoring tools available to non-programmers [27], [30]–[33]. This last goal is, indeed, the main focus of this work.

Past research efforts have led to various authoring tools and frameworks which aim to reduce the skills and amount of time required to create tutors. Tools such as CTAT [34], which is a well-known set of tools that relies on the Jess rule-language; frameworks such as GIFT [35] or other authoring tools such as TDK [36]. Different models are available to specify the ITSs and CAs, with varying levels of power and complexity, and different advantages and disadvantages. For example, CTAT's cognitive tutor creation relies on XML and Jess rules.[1] It is used by systems such as [30]. It is powerful, but, though it does indeed require no programming experience, it tends to require considerable IT expertise. Specifically, it requires knowledge of computer languages such as the aforementioned ones, and, generally, it involves using relatively complex environments such as the Flash IDE or Eclipse. Other tools rely on specific XML-based languages [33]. CTAT's *example-tracing* tutor creation requires no JESS knowledge and is simpler [37]. Works such as [31] rely on conversation trees. In this case, the non-programming authors build a tree structure, specifying for each branch a selection of responses among which the end-user can choose. Other systems rely on NLP and other techniques to extract information from different sources and automatically generate an ITS or a CA [38], [39].

In this work we propose a novel model to define such an agent. A domain specific language (DSL) that relies on a Visual Programming Language specifically designed for non-programmers has been created. Through it, even those users who are not experienced in IT can specify rules and customize the behavior and domain-model of the agent.

B. VISUAL PROGRAMMING LANGUAGES

Visual programming languages are those that are based on graphical elements. Examples of those elements are puzzle blocks, in the case of Google Blockly [12] and Scratch [40]; or block diagrams, in the case of LabView [41]. They rely on drag-and-drop and other spatial actions instead of being based on text. As an example, Figure 2 shows the Scratch environment, oriented for children.

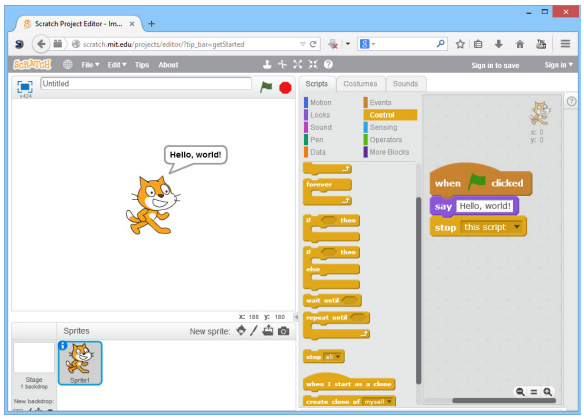

FIGURE 2. The Scratch visual programming language and environment.

Visual programming languages have proven their usefulness in certain domains such as education [42]: Google Blockly or Scratch have been designed to be intuitive and easily understood [43], even by very young students. Google Blockly has been used to teach them programming [44], [45]. It is particularly fit for these tasks [43], because, among other reasons:

- It is intuitive: students can see the blocks that are available and check whether they connect.
- Students do not need to remember a syntax and adhere to it before they start getting results.
- No syntax errors: if the blocks fit, then the program is valid.[2]

Google Blockly is Open Source. It is based on blocks that the user can drag-and-drop to connect to each other and nest them. Once the blocks are arranged, Blockly can generate executable code for text-based languages (e.g., JavaScript, Python, and other languages), so that it can be run. JavaScript is the one most commonly used, because then it can run straightaway in the browser. Although it is

[1] http://ctat.pact.cs.cmu.edu

[2] This does not necessarily imply that the program does what the user expects. It only implies that it can be translated into a text-based language with no syntax errors.

originally designed to teach young students programming, it is also designed to be completely customizable, including the blocks, the connections, the generated code and the looks. For that reason, it is particularly effective for creating visual DSLs. It has been used for purposes as varied as educational robots scripting [46], smart home applications [47] and business processes modeling [48].

III. REQUIREMENTS

This section describes, first, the general requirements for the proposed approach. Second, it describes additional, more specific requirements for the prototype authoring platform. Later sections of this paper (see Section VII) will evaluate whether these requirements are met. The main design goals for the proposed approach are the following:

- **Low skill requirements**: Non-programmers should be able to use it.
- **Simple and predictable output**: The CA creators should be able to easily predict the output, which should be well-defined.
- **Integrable into other resources**: The produced agents are meant to be integrated into other educational resources.

Additionally, the authoring platform that implements the proposed approach should meet the following practical and technical requirements:

- **Web-based and mobile-friendly**: The tools and the produced conversational agents should be fully web-based, thus available anywhere.
- **Integrable agents into external systems**: As previously described, the generated agents are not meant to be stand-alone. Instead, they are meant to be integrated into other systems such as Learning Management Systems or Remote Labs, as an aide. It should be technically possible to integrate them with minimal intervention from the external systems.

The proposed approach is designed to be extensible. Advances in related fields could be leveraged to provide advanced features for the produced agents (e.g., speech recognition, realistic conversations). Though those capabilities are beyond the scope of this work, the authoring platform and the visual language could be extended to make use of advanced Natural Language Processing (NLP) techniques. The current proposed version focuses on being able to produce predictable, known and controlled output, without requiring data sets. It does not aim for the agents to be particularly powerful from an Artificial Intelligence point of view, but aims for simplicity instead.

In the next subsections, the aforementioned goals and requirements are described in more detail.

A. LOW SKILL REQUIREMENTS

Domain experts do not necessarily have programming or advanced IT skills. So that they can create and customize the agents, it is important that the authoring tool be easy to use and intuitive. It should avoid requiring knowledge of programming or of computer languages (such as XML, JSON or AIML) that are not necessarily easy to understand for the general public.

B. SIMPLE AND PREDICTABLE OUTPUT

The design goal of some CAs is to be as natural as possible. Such an agent should be able to provide varied and often unexpected output, like a human would. Some agents, such as Tay by Microsoft, even rely on previous conversations to learn new answers. This is, however, not the goal of the model proposed in this work. Natural agents are not without drawbacks. For instance, the agent by Microsoft, which has since been shut down, is known to have learned to be racist, utter profanities, and, in short, behave in a way that was particularly unexpected and undesirable for its creators.[3] The model proposed in this work explicitly sacrifices believability so that it can be easier to define, the output can be predictable and more easily understood, and no data sets are required.

C. INTEGRABLE INTO OTHER RESOURCES

The CAs produced by the authoring platform are not meant to be used stand-alone. They are meant to be integrated into other educational contents, such as online labs. Those can be relatively complex virtual labs, as in the case of the PhET simulations [49]. Or even online interfaces to real equipment, as in the case of remote labs [15], [50]. Online labs are often designed to satisfy many use-cases, which makes it hard for students to use them without guidance. In this case, for instance, a CA designed by a non-programmer would be able to provide context-specific information, provide non-intrusive help when needed, and enhance the lab experience.

D. WEB-BASED AND MOBILE FRIENDLY

The goal of the described architecture is to be accessible to as many people as possible. For that purpose, it is very important that it is based on standard web technologies and that it supports mobile devices. This is critical for education: nowadays many schools and students rely on mobile and tablet devices [51]–[54]. This implies that the architecture must only rely on web standards, and avoid certain proprietary technologies, such as Adobe Flash or Java Applets. Though common in the past, they no longer have wide support and have significant security implications [55].

This is also important for the platform to be deployable easily in different networks and environments. School and institutional networks are often restricted. Relying on non-HTTP ports, or on plugins or technologies which require administrator privileges to deploy, would make the system unacceptable to many potential users.

E. INTEGRABLE AGENTS INTO EXTERNAL SYSTEMS

As previously described, the produced CAs are meant to be integrable into other educational resources. From a technical point of view, those external resources are typically

[3] http://money.cnn.com/2016/03/24/technology/tay-racist-microsoft/

web-based platforms (such as Moodle instances or online labs) hosted by external providers. To be able to be used effectively, a key feature is that the produced agents should be able to be integrated into those resources without requiring intervention from the administrators of the external system (As an example, see Figure 1).

IV. PROPOSED APPROACH

One of the main focuses for the proposed approach is to allow non-programmers to create and customize CAs. Therefore, as described, it relies on a visual DSL. We have designed the DSL specifically for this purpose, and it is one of the contributions of this work. It relies on the Google Blockly library. Google Blockly was initially oriented towards children, and can thus be used to create intuitive and easy to use languages. This tool is described in more detail in later sections. It is oriented towards extensibility. The functionality of Google Blockly, and thus, the functionality of the DSL, depends on the *blocks* they provide. A *block* is, in that way, akin to a function in a standard programming language.

The non-programming authors, who are the domain experts, define the behavior of the CAs through that visual DSL. This DSL is fully based on a set of custom blocks that we have created. The base block is a *Conversation Node* block. The non-programming authors attach conditions and actions to them, which also take the form of blocks. A very simple condition block, for instance, would be a *words detected in input* block. A very simple action block, for instance, would be a *say* block. The variety, power, and complexity of action and condition blocks would vary depending on the purpose, needs, audience and platform, and can be tailored and extended easily.

With no extensions, the language and the prototype authoring platform are designed to be simplistic. This makes the agents most useful for creating narrow-scope agents, in which the inputs of the end-user are predictable and in which the author wants fine-grained control over the output of the agent. However, it makes them unsuitable for realistic human-like conversations.

One of the main motivations of this work is to enable non-programmers (such as teachers or even students) to create and integrate bots in educational content such as MOOCs, online courses, and remote [56], [57], virtual [49] or hybrid labs [58]. For this purpose, the approach is also designed for straightforward interaction with these systems. The bots themselves, as the prototype platform shows, can be integrated along with that content. Also, the language is built to support straightforward interaction. An example is the *raise event* block. It can be used, for instance, for proposing and evaluating teacher-defined practical exercises. Thus, teachers could create an agent that proposes an exercise, integrate it into a generic virtual lab, and have their students solve the particular exercise using that generic lab. The agent would describe the exercise, provide guidance, evaluate the response, and raise a particular event when the exercise is solved.

Although not explored in detail in this work, this scheme is suitable for Embodied CAs (ECAs) [59], because the described DSL, with the proper blocks, would allow the author to control accurately the behavior and actions of the virtual character.

A. VISUAL DOMAIN-SPECIFIC LANGUAGE

In this section we detail the visual DSL and its basic blocks. However, as previously described, the system is designed to be extensible. Depending on the platforms, target, and advances in the state of the art, specific platforms that implement it may easily extend it.

1) BASE CONVERSATION NODE

The basic block is the *Conversation Node*. This block establishes the conditions under which the node will be triggered, and the actions that will be executed when the conditions are met. Figure 3 shows an example with two base conversation nodes, each of which contains condition and action blocks. Users can attach or remove actions and conditions by simply dragging-and-dropping them.

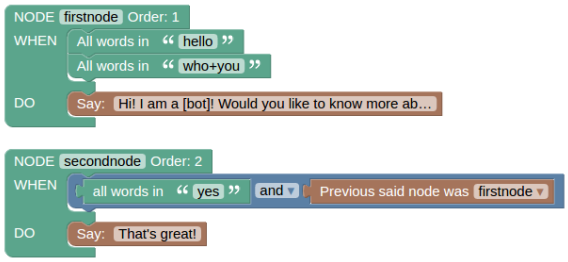

FIGURE 3. Example of two base nodes with condition and action blocks within.

2) CONDITION BLOCKS

- **Contains-words condition block:** *Contains-words* is a basic condition block. It triggers the conversation node when the user's input contains all of the specified words. More than one word can be specified by separating them with plus signs or with spaces, as shown in Figure 3.
- **Logic condition block:** The logic condition blocks let users combine conditions with logical operators. An example is the *AND* block. They can be nested to build complex conditions.
- **Previous-node block:** The previous-node block can be used to specify that the node should only be triggered when the last triggered node was a specific one. That is a simple way to handle, for instance, responses to yes/no questions without requiring the use of contexts, or more complicated schemes.
- **Default condition block:** There is a special block that can be used to specify nodes that will be triggered by default when no other node is triggered. This block can be used to specify default responses. Another way is to simply rely on the ordering of the blocks.

3) ACTION BLOCKS

- **Say block:** The most basic action block is the *say* block. In the case of the prototype authoring platform, it will simply display the text in a speech bubble above the virtual agent's head.
- **Raise-event block:** Raises a specific custom event. It is intended to be captured by the external system the CA is integrated into. The actual purpose of this will depend upon the external system, which can provide specific events. Those can, for example, enable teachers to propose custom exercises through the CA system.

4) NODE ORDERING

Each node is ordered vertically, and the order is explicitly shown as a number. This defines the priority of a specific node. Users can rely on ordering to trigger different nodes depending on whether the higher-priority nodes conditions are met or not. Thus, when two different nodes have conditions that partially overlap (for example, one node is to be triggered when the user says ''who are you'' and the other when the user says something with ''you''), the less restricting ones will normally be lower-priority and act as context-specific defaults.

B. STRENGTHS AND WEAKNESSES

The main strengths of the proposed approach are the following:

- **Simplicity**: Non-programmers can learn and use it easily.
- **Fine-grained control**: Agent authors can control exactly what the bot says and when.
- **Power**: Despite its simplicity, Blockly can support a fully-fledged programming language. With extended blocks, it can provide as much power as one. Alternative approaches that are oriented for non-programmers such as example-tracing tend to be less powerful.
- **Integration**: The approach can be integrated easily into other systems. In more automated systems, or in systems on which the author does not have a fine-grained control over the output, it can be less obvious how to interact with other systems reliably. Likewise, it makes it suitable for controlling a virtual agent, and thus the behavior of an ECA.
- **Predictability**: Authors can easily predict the exact output of the bot under given conditions. Thus, for a given question, they can know exactly if the bot will be able to answer, and how; avoiding surprises and indeterminacy.
- **No data or training required**: The system does not rely on corpora, datasets, or training.
- **Language-agnostic**: While systems that rely on NLP techniques and data corpora tend to be tailored towards a specific language (such as English), the described approach, without specialized blocks, is language-agnostic.

And the main weaknesses would be the following:

- **Limited scope**: This approach is not suitable for the creation of agents with a large body of knowledge. Too many nodes would be time-consuming to create manually.
- **Realistic language**: Without advanced NLP blocks, it is not suitable for generating realistic conversations. For this purpose, the intended predictability is also a disadvantage: being unpredictable would add realism.
- **No automation**: No automatic learning or information extraction functionalities are present.

As a consequence of this, these agents are suitable for these cases in which the authors want to obtain a narrow-scope agent, in a context in which they can predict the input, and want fine-grained control over the results. This is the case of several educational contexts, such as a remote lab, in which the range of possible questions and interactions is limited and can be predicted by the domain expert (the teacher) and in which the said expert does not need (or even want) the students to have broad-scope conversations with the agent. This is also an advantage for these contexts in which authors may want to use different languages, because the approach is language-agnostic. It is also important to remark that the approach is intended to lead to embeddable agents, not standalone ones (for which realistic conversations would probably be more important).

V. PLATFORM ARCHITECTURE

The architecture is fully web-based and it relies on certain key technologies to provide the required features while maintaining platform independence. This section will describe the proposed architecture, which has been designed, developed and evaluated. To better understand how the authoring platform looks and works in practice, a short screencast is supplied as an online Multimedia Material ('authoring.mp4').

To describe the architecture it is important to remark the two different perspectives:

- The **authoring tool**: For the authors, who normally will be domain experts.
- The **conversational agent component**: Which will be integrated into an external system, such as an LMS or a remote lab, and implement the agent defined by the author.

A. KEY TECHNOLOGIES

1) GOOGLE BLOCKLY

Google Blockly[4] [12], which was briefly described earlier, is an Open Source visual programming language created by Google. It is originally intended to teach basic programming to young students. They can first learn basic programming concepts, such as logic, variables and loops in Blockly, and once they understand them they can move to the normally less-intuitive text-based languages.

[4]https://developers.google.com/blockly/

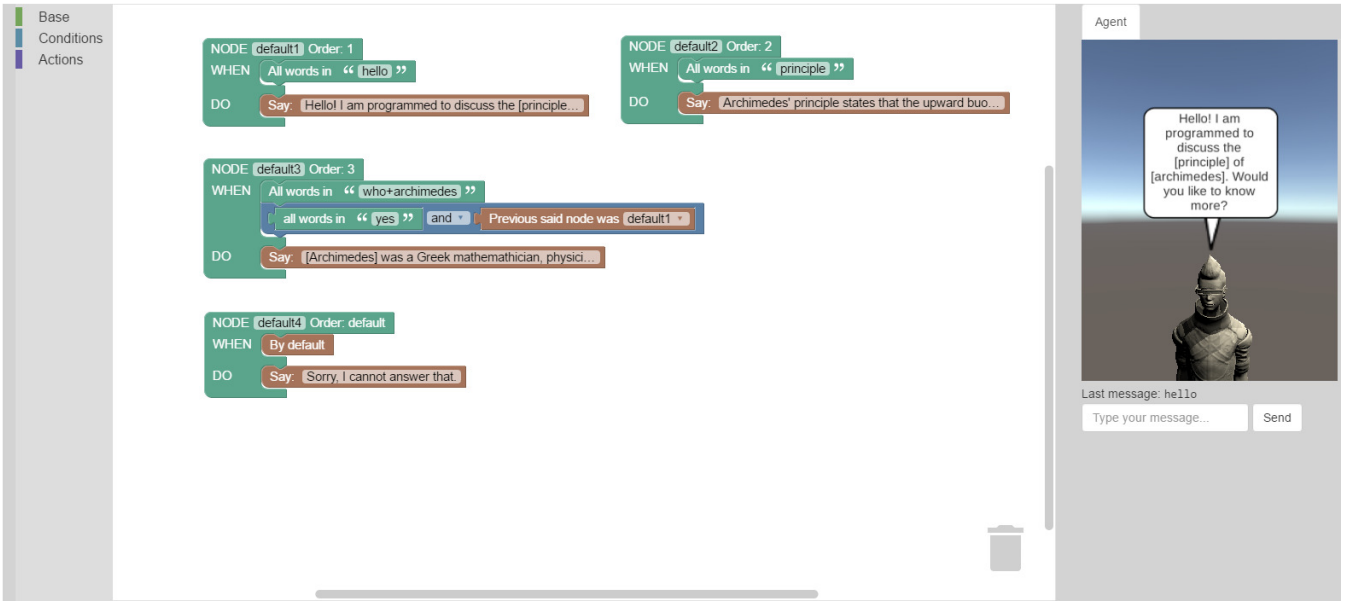

FIGURE 4. Authoring Tool's main view.

Google Blockly is designed to be customized and modified easily. Completely new blocks and code generators can be developed, so it is well-suited to create visual domain-specific programming languages. In this work, we have designed and developed such a language so that non-programming users can very easily be able to define the logic of their CAs, which will be described in the next sections.

2) UNITY AND WEBGL

Unity3D[5] is a very popular 3D application creation tool. The 3D applications created in Unity can be exported to different platforms. One of those are browsers, through WebGL [60]. WebGL is a standard by the Chronos group: a web-based graphical API that makes accelerated 3D graphics available on the Web. Though it is not part of the HTML5 standard itself, it has strong ties to it and is supported by every major browser, including mobile ones. Although SVG or Canvas can also be used for web-based graphics, WebGL is more powerful; it supports full 3D acceleration, shaders and relatively low-level access to the graphics card.

Thus, using Unity and WebGL for the CAs we can get agents that:

- Are fully 3D and can rely on advanced shaders and animation and be in a virtual environment.
- Are fully web-based and cross-platform, being deployable even in mobile and tablet devices.

B. AUTHORING TOOL

As described in earlier sections, the goal of the authoring tool is to enable users, including those without programming or IT knowledge, to create and customize their own CA relying on a visual domain-specific language that has been designed and implemented for that purpose. A screen capture of the main view of the Authoring Tool can be observed in Figure 4. An overview of the architecture of the authoring system is depicted in Figure 5.

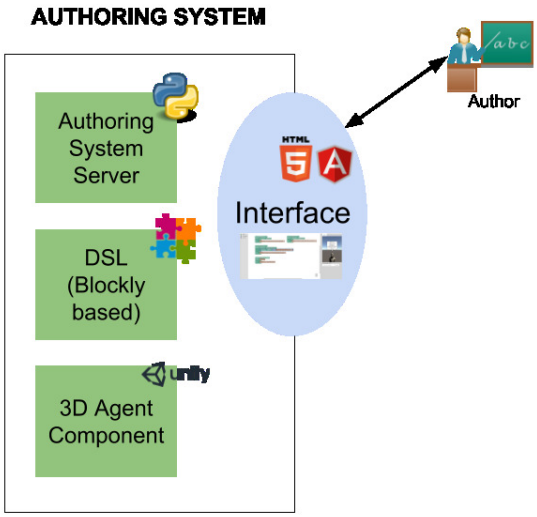

FIGURE 5. Authoring system architecture overview.

Several components form the authoring tool. The authors —who will generally be non-programming domain experts— interact with it through the *Interface*, which is a web-application and which is thus supported in all the main browsers and in tablets. The authoring tool web-app as a whole is managed by the *Authoring System Server*, which is, essentially, the server-side. The agent authors use the *Google Blockly based visual DSL* to create and customize the agents (thus defining the domain model). This component is also responsible for generating the JSON specification (see Figure 6) from the visual code, and forwarding it to either the server (for storage) or to the 3D agent component.

[5]https://unity3d.com

```
{
    "formatVersion": 2,
    "conversationNodes": [

        {
            "node": "default1",
            "order": 1,
            "on": [
              "hello",
              "hi"
            ],
            "do": [
              { "say": "Hello, I can explain to you the basic [electrical c
            ]
        },

        {
            "node": "default2",
            "order": 2,
            "on": [
              "electrical components",
              { "$and": ["yes", { "last_node": "default1" }] }
            ],
            "do": [
              { "say": "You can use [resistors], [capacitors] or [diodes]."
            ]
        },

        {
            "node": "default4",
            "order": 3,
            "on": [
              "bye",
              "goodbye"
            ],
            "do": [
              { "say": "You can use [resistors], [capacitors] or [diodes]."
            ]
        },
```

FIGURE 6. Example of the advanced JSON view for an agent. This is transparent to the user and normally not visible.

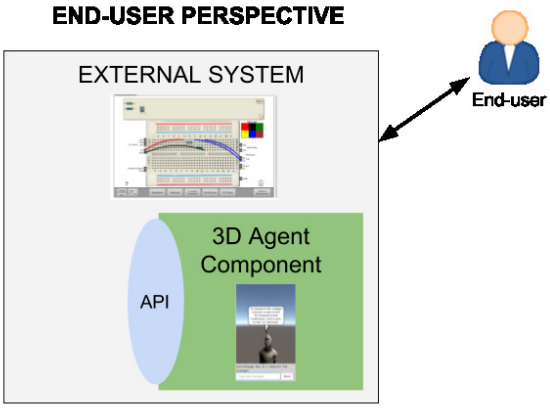

FIGURE 7. The end-user perspective.

The *3D agent component* within the Authoring System is there so that the authors can test their agents in real-time and check whether they function as they expect.

C. END-USER PERSPECTIVE

From the perspective of the platform described in this work, there are two different kind of users.

- **Authors**: They will use the authoring tool to create and customize conversational agents. They are not expected to have programming or IT knowledge, but can be considered domain-experts.
- **End-users**: They will use the CAs created and customized by the author-users. They will not necessarily have programming or IT knowledge either.

The CAs created through this platform are purposefully simplistic, and are not meant to be used as stand-alone intelligent tutors. Instead, they are integrable and add value to external educational systems, such as LMSs or remote labs. The end-user perspective is summarized in Figure 7 and an example can be observed in Figure 1. The end-users interact with an external system, which could be for instance a web-based remote lab. The 3D agent component would then be displayed as an *iframe* within that remote lab. The remote lab would forward interactions to the component, which implements the agent previously defined by an author-user. The interactions are forwarded through a bidirectional JavaScript API based on *window.postMessage*, which will be described in more detail in later sections. The purpose of this API is to guarantee that the agent component can be integrated into other systems, including those that are hosted in a different domain than the actual tool.

D. COMPONENTS

This section describes in a lower-level detail the key components of the platform.

1) GOOGLE BLOCKLY BASED DSL

Throughout this work we have developed a Blockly-based DSL based on rules to enable the users (non-programmers) to easily define the behavior of the agent. The language relies on a set of custom blocks that internally generate JSON code (transparently to the users, unless they choose otherwise). Then, the CA engine interprets that JSON code to behave in the way that the non-programmer user defined.

The language for this implementation, which is meant to act as an example implementation for the model, is purposefully designed to be very simplistic. It is based on trigger-nodes. Users specify the *conditions* under which the node will be triggered, and the *actions* that will take place once the node is triggered. The most common and relevant action is to *say* a specific sentence, though the system itself supports other actions.

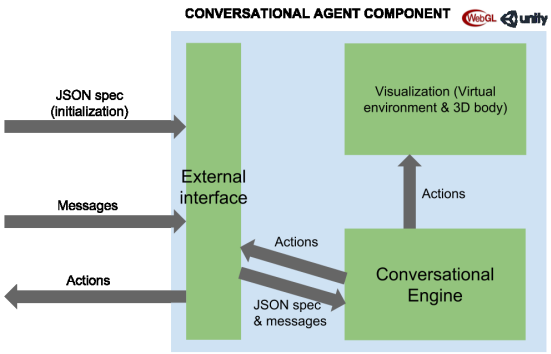

FIGURE 8. The Conversational Agent component architecture.

2) INTEGRABLE 3D TUTOR COMPONENT

The 3D tutor component itself, as previously described, is created in Unity3D and exported to WebGL (JavaScript). Its architecture is depicted in Figure 8. This component contains the conversation engine that receives the JSON DSL — which is essentially the knowledge model— and the input from the user, and chooses the appropriate actions. It also provides a virtual 3D environment with a 3D body for the tutor,

which can speak according to the engine and carry out other actions.

3) INTEGRATION

The architecture described in this work is meant to be integrable into other systems such as Learning Management Systems, virtual laboratories and remote laboratories. For that purpose, as described in Section II, the 3D tutor component has been created in Unity3D to be exported into WebGL and integrated anywhere as an HTML5 *iframe*. It contains the CA engine that interprets the provided JSON, receives input from the user, and runs the specified actions when appropriate. It communicates through the JavaScript *window.postMessage* API instead of communicating through JavaScript directly so that it can avoid cross-domain issues that would arise when hosting the 3D tutor component into a different domain's *iframe*.

E. ARCHITECTURE VALIDATION

In order to validate the architecture and to conduct further experiments, the DSL, the Conversational Agent engine, and the Authoring Tool have all been implemented. Then, it has been verified that the main technical and functional goals are met. The tools can indeed be used to create 3D conversational agents using the proposed visual blocks-based DSL. The tools, both to author and to use the agents, are all fully web-based. The agents can be successfully integrated into external web contents. The proposed language-level features are all supported by the implemented DSL, and the blocks can be extended easily so new features could be added. These implementations of the architecture have been used to conduct the user experiments that are described in the following sections.

VI. METHODOLOGY

To evaluate the proposed approach and to explore the research questions that were specified in Section I-A, user tests were conducted. For this, the following software platforms were created:

- The prototype web-based *authoring platform* whose architecture was described in Section V.
- A web-based *testing platform* to lead the study participants through several predefined stages, during which they use the authoring platform, and data is collected.

In the final stage, participants were asked to fill a standardized UMUX questionnaire (that measures usability), and a custom survey to evaluate their perception of the authoring platform and of the proposed approach.

A. THE TESTING PLATFORM

A web platform was created to lead participants through the stages depicted in Figure 9. First, in the *example* stage, they see and use a resulting CA for the first time. Second, in the *tutorial* stage, they learn how to use the authoring tools through an interactive tutorial. Third, in the *challenge* stage, they are given a topic and several constraints and are asked to create a CA. Fourth, in the *survey* stage, they provide feedback about their experience.

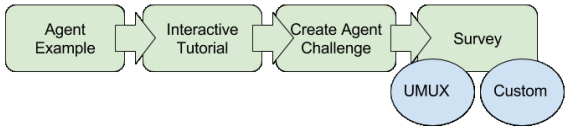

FIGURE 9. Experiment stages that the participants follow.

A short screencast is provided as an online Multimedia Material ('experiment.mp4'), briefly demonstrating the testing platform and how users go through its various stages. It may be noteworthy that in the video the stages are completed very fast: real first-time users will need to read through the text, make more errors, and take much more time (as the time measurements in Section VII show).

B. STAGES

1) EXAMPLE STAGE

The goal of this stage is to familiarize the participant with the CAs. This specific CA speaks about the laws of thermodynamics. However, the content itself is not important, because in practice it will depend on each agent's author. To ensure that the participant uses the agent sufficiently, we measure the different agent responses that are triggered. Once 5 different responses are triggered, the participants may continue to the next step.

2) TUTORIAL STAGE

The goal of this stage is to teach the test subject to create CAs using the prototype authoring tools and the visual DSL. This is challenging, because even though the visual DSL is designed to be intuitive, time is constrained and concepts such as CAs, computer languages, or logical conditions are novel to most participants. The system automatically evaluates whether what they are doing is right or wrong, provides tips, and suggests the next step to take. Users cannot skip to the next step until their current step works as intended.

3) CHALLENGE STAGE

In the challenge stage, participants apply what they have learned to create a CA. We provide a topic —in this case, a museum guide— and several constraints. This way, we ensure that the resulting data is comparable and that they use several types of visual blocks, including complex ones. In this stage they can still see a 'cheatsheet' with examples, but they receive no guiding or tips. They only receive feedback about whether the constraints are currently met or not. A screenshot of this stage can be seen in Figure 10.

4) SURVEY STAGE

In this last stage the participants fill a 9-question survey, which is actually split in two different parts. The first part is a standardized UMUX (Usability Metric for User Experience) survey [61], [62]. UMUX is a four-items Likert scale to assess an application's perceived usability. It is designed to measure the three dimensions of usability defined by the ISO 9241-11 standard [63] (effectiveness, efficiency and

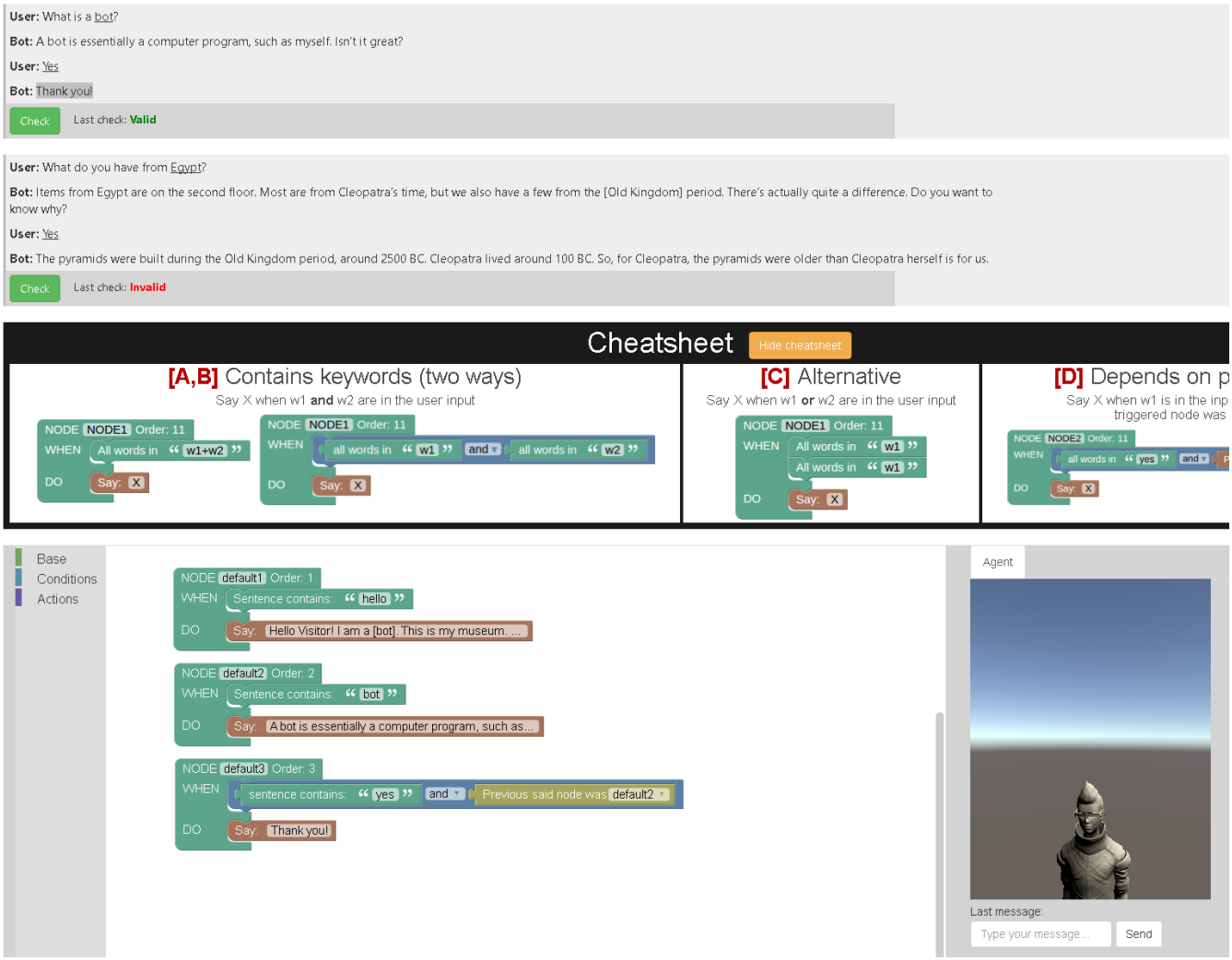

FIGURE 10. Screenshot of the Challenge stage of the experiment.

satisfaction) and to provide similar results to the longer ten-items System Usability Scale (SUS) [64], [65], but with less questions. Through this questionnaire the usability of the authoring platform prototype and the visual DSL that it relies on is measured. The second part has six questions, which are more specific, and are designed to evaluate the perception of the participants and their satisfaction with the approach and the authoring tools. All participants were Spanish-speaking, so the original questionnaires were in Spanish. The translated survey questions are detailed in Table 1.

C. PARTICIPANTS

32 first-year students from the University of Deusto, in Spain, took part in the study. 14 of them were from the campus in San Sebastian and were enrolled in a Business Administration degree. 18 of them were from the campus in Bilbao and were enrolled in an Electronics Engineering degree. Participation was voluntary. The main intended audience of the authoring platform are non-programming users with some technology experience. Therefore, the students met the criteria. They played the role of content creators. The students were non-programmers and all were familiar with technology such as graphical interfaces and web browsing. All were native Spanish speakers.

D. PROCEDURE

Two group sessions were held in total, one for each campus. The first one had 14 of the participants, while the second had 18. Each student had access to a computer. An hour of time was allocated for each session. During the session, students were briefly introduced to the topic of CAs and the purpose of the authoring platform. Then, they were directed to the tutorial platform and provided some general advice (such as making sure to follow the instructions at each stage very precisely, because the system is automated and they would be unable to advance otherwise). The participants then went through each of the stages, directed by the testing platform, and at their own pace. The interactive tutorial of the platform is designed to be self-sufficient, but the participants were allowed to ask questions and they received directions when

TABLE 1. UMUX and custom questionnaires.

Standard UMUX questionnaire	
Q1	*The Authoring Tool's capabilities meet my requirements* 7 point Likert-type scale (1 to 7), from *Strongly Disagree* to *Strongly Agree*
Q2	*Using the Authoring Tool is a frustrating experience* 7 point Likert-type scale (1 to 7), from *Strongly Disagree* to *Strongly Agree*
Q3	*The Authoring Tool is easy to use* 7 point Likert-type scale (1 to 7), from *Strongly Disagree* to *Strongly Agree*
Q4	*I have to spend too much time correcting things with the Authoring Tool* 7 point Likert-type scale (1 to 7), from *Strongly Disagree* to *Strongly Agree*
Custom questionnaire	
Q5	*Non-programmers would be capable of creating bots with a Visual Language such as the one we used* 7 point Likert-type scale (1 to 7) from *No teacher could do it* to *Yes, everyone could do it*
Q6	*It would be useful to integrate bots into other educational content (such as online courses; to support the student)* 7 point Likert-type scale (1 to 7), from *Strongly Disagree* to *Strongly Agree*
Q7	*It can be useful to integrate bots into some educational systems (as support for online courses; or virtual or remote labs)* 7 point Likert-type scale (1 to 7), from *Strongly Disagree* to *Strongly Agree*
Q8	*I found technical problems during the experiment* 7 point Likert-type scale (1 to 7), from *Strongly Disagree* to *Strongly Agree*
Q9	*Please, make any observation, comment or complaint* Free text response

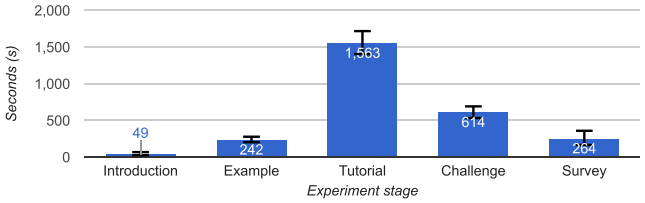

FIGURE 11. Average time in seconds that participants spent in each stage (n=32). Error bars show 95% confidence intervals.

they got stuck in a particular step or came across a potential technical issue.

VII. RESULTS

The results of this research are organized around the Research Questions enumerated in Section I-A.

A. HOW EASY TO LEARN IS THE PROPOSED APPROACH?

The testing platform measured the time that the participants spent in each stage. The average times in seconds are shown in Figure 11. The error bars show the 95% ($\alpha = 0.05$) confidence intervals. The sum of the average times for each stage is 2732 seconds for all stages (45.5 minutes).

The tutorial stage, as expected, took the longest: 1563 seconds (26.05 minutes) on average. The fastest participant took 1112 seconds (18.5 minutes) while the slowest took 2920 seconds (48.67 minutes). The challenge stage took 614 seconds (10.23 minutes) on average. The fastest participant took 293 seconds (4.88 minutes) while the slowest took 1183 seconds (19.71 minutes).

B. HOW USABLE IS THE PROPOSED APPROACH?

As previously described, the evaluation of the authoring platform usability has relied on a conventional UMUX questionnaire. As [61] explain, odd items are scored as $[score-1]$ and even items are scored as $[7-score]$. The preliminary UMUX score is thus in the 0-24 range. The sum for each participant is then divided by 24 and multiplied by 100 to convert it to the 0-100 SUS-like scale. The mean score is then calculated.

The UMUX mean score for the 32 participants is 73.31. The standard deviation is 15.62, with a 95% ($\alpha = 0.05$) confidence interval half-width of 5.41 ([67.89 − 78.72]). According to [65], this score is in the *acceptable* range and can be considered "good", though not "excellent". However, considering that the platform is a research prototype, and that the participants had a short time to learn to use the relatively powerful visual language, the result can be considered satisfactory.

TABLE 2. Results of the custom survey Q5-Q8 (n=32).

	Question (1-7 points Likert-type scale)	Mean	S.D.
Q5	*Non-programmers would be capable of creating bots with a Visual Language such as the one we used*	5.78	1.008
Q6	*It may be useful to integrate bots into some educational systems [...]*	6.00	0.916
Q7	*The visual block-based language to program the bots is intuitive*	5.94	1.076
Q8	*I have come across technical issues during the experiment*	3.62	2.196

C. IS THE PROPOSED APPROACH PERCEIVED AS VALUABLE AND INTUITIVE?

The second part of the survey consisted of 5 questions. The first four, for consistency, use a Likert-scale with 7 points. However, unlike the UMUX ones, they are specific and are considered independently rather than as a metric. The last question is an invitation to freely comment anything. The results of this questionnaire are summarized in Table 2. They show that users overwhelmingly believe that everyone would be capable of creating bots with such a visual language ($\bar{x} = 5.78$), that it would be useful to integrate bots into educational content ($\bar{x} = 6.00$), and that the proposed visual DSL is intuitive ($\bar{x} = 5.94$).

Only a few of the participants chose to respond to the free-comments question (Q9), which was optional. Most of the participants left the question blank. However, feedback from the few that did answer was remarkably positive. The comments (translated) were:

- "It is extremely easy to use. Very intuitive and simple."
- "It's a very good idea and I hope it moves forward!"
- "I would remark how intuitive it is"
- "Very good"

VIII. DISCUSSION

This work has proposed an approach to create and customize CAs using a visual DSL that is friendly for non-programmers. An authoring platform has been created to showcase the approach and to study its effectiveness, and a user study has been conducted.

A. THE PROPOSED APPROACH

The experience and the results suggest that, indeed, the approach can be a useful way to define CAs, especially for non-programmers. This does not necessarily mean that it is the best approach for all cases, but it may be the most appropriate for some of them. Alternative approaches such as using a text-based formal language (such as AIML or XML) may yield more flexibility, but in exchange can generally be expected to be harder to learn. On the contrary, approaches such as example-tracing may be easier to learn than the proposed visual DSL, but are (at least if not combined with other approaches) significantly less powerful. Thus, using a visual DSL can be considered a compromise between simplicity and power that may be very appropriate for some domains.

An additional advantage of this approach is that its capabilities and complexity can be tailored freely. In such a visual language, those depend on the *blocks* that are provided by the authoring platform. Thus, the platform can offer blocks that are more or less complex depending on the needs and audience, and advances in related areas could be leveraged easily. For instance, some advanced functionalities that may be integrated easily into the blocks system would be:

- Text-to-speech: Integrated into the *say* block, using an API such as Google's.
- Speech-to-text: Integrated into the *keywords-recognition* block, using an API such as Google's,[6] and automatically providing the API with clues about the expected words to increase accuracy.
- Blocks to control the behavior of the 3D agent to leverage advances in Embodied Conversational Agent (ECA) research.
- Blocks to raise evaluation events so that the domain-expert can create, propose and evaluate customized exercises for online laboratories without requiring collaboration from the author of the lab.

[6]https://cloud.google.com/speech/

B. AUTHORING PLATFORM PROTOTYPE

In general the results have been satisfactory. Usability, according to the UMUX questionnaire and the previously mentioned thresholds, can be considered good enough, though there is still some room for improvement. The perception that users had of the platform was very positive (and generally, higher than their usability perception).

From a technical perspective, the technology choices were appropriate. Google Blockly was chosen as a base for the visual DSL, and it has met the requirements. It has been stable, easy to extend and intuitive enough for the participants to learn to use it in a very short time and with very little help. Similarly, the choice of Unity and WebGL as a rendering engine to implement the CA's interpreter engine and ECA has been satisfactory. It allows it to be web-based and can be run, as was the goal, in almost any browser, and thus meets the universality, security and deployability goals that we had set in section II.

Though exploring it in detail was not the focus of this work, it is also noteworthy that the interactive tutorial could be improved. Although it was effective (participants were able to learn how to create agents in a short time) some participants found it frustrating. Not allowing users to skip to the next step until they have done the current one properly is an effective means to guarantee that the learning goals are achieved, but at the same time can be found frustrating if they are unable to find the mistake in a short time, or if they don't receive appropriate feedback. Ways to improve this issue could be explored.

IX. CONCLUSIONS

To create effective educational CAs, domain knowledge is required. This work has proposed a novel method to define such agents that is both simple —accessible to non-programmers— and expressive —capable of supporting complex rules and being extended with additional *blocks*—. It has also described an authoring platform that leverages that method to allow non-programmers to define their own agents. These agents are embeddable into external educational content, without requiring explicit collaboration from the content's creator.

The results of the study suggest that a visual DSL based on Blockly is indeed a promising way to define CAs for non-programmers. Even though the platform is still a prototype, the participants of the study were able to learn to use them fast, and created their own CA. Additionally, the standard UMUX usability test suggests that the authoring tools prototype and the visual DSL that it relies on have satisfactory usability already. With additional work it can be expected to produce better results. The specific survey, as well, shows that participants are very satisfied with their use of the tool, and believe that it has significant potential. The approach itself (using a visual DSL to allow non-programmers to create or customize CAs) seems promising. Although the visual DSL is expressive, non-programmers can learn and understand it in

a short time. An additional advantage of this approach is that the visual language could be extended with custom blocks that provide additional capabilities and that leverage advances in related areas.

X. FUTURE WORK

In the future certain improvements and modifications to the CA model could be investigated:

- The language processing system is currently very simplistic and based on keywords. It might be worthwhile to find ways to apply more advanced NLP techniques to the CA engine, and to check whether the new capabilities are useful enough, taking into account the potential cost in simplicity.
- The set of potential actions of the engine is currently limited. It might be worthwhile to add new synchronized animations or 3D interactions, relying on the existing literature about embodied CAs.

Apart from those lines of research that mainly involve improvements to the CA model and engine, certain potential applications that relate to online laboratories and that could be explored are worth mentioning:

- As a non-programmer customizable intelligent tutor for an online lab, which may guide students and provide specific advice on its usage, expectations and results.
- Additionally, as a non-programmer customizable intelligent tutor which also offers automatic evaluation capabilities. Thus, the teachers could integrate a tutor into the online lab of their choice, design the problem description and the expected output, and have the intelligent agent automatically evaluate students. This would be particularly useful because virtual and remote labs often just offer an open environment but not a customized practice session or experience, and because in distance education the teacher often has to evaluate a large number of students, and tools which make this easier can have a significant impact.

REFERENCES

[1] B. D. Nye, A. C. Graesser, and X. Hu, "Autotutor and family: A review of 17 years of natural language tutoring," *Int. J. Artif. Intell. Educ.*, vol. 24, no. 4, pp. 427–469, 2014.
[2] J. Cassell *et al.*, "Embodiment in conversational interfaces: Rea," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 1999, pp. 520–527.
[3] K. Vanlehn, "The behavior of tutoring systems," *Int. J. Artif. Intell. Educ.*, vol. 16, no. 3, pp. 227–265, 2006.
[4] J. Weizenbaum, "ELIZA—A computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, 1966.
[5] S. Kopp, L. Gesellensetter, N. C. Krämer, and I. Wachsmuth, "A conversational agent as museum guide—Design and evaluation of a real-world application," in *Proc. Int. Workshop Intell. Virtual Agents*. Kos, Greece: Springer, 2005, pp. 329–343.
[6] T. W. Bickmore, D. Utami, R. Matsuyama, and M. K. Paasche-Orlow, "Improving access to online health information with conversational agents: A randomized controlled experiment," *J. Med. Internet Res.*, vol. 38, no. 1, pp. 2057–2060, 2016, doi: 10.2196/jmir.5239.
[7] V. Aleven, J. Sewall, O. Popescu, M. van Velsen, S. Demi, and B. Leber, "Reflecting on twelve years of ITS authoring tools research with CTAT," in *Design Recommendations for Intelligent Tutoring Systems*. Orlando, FL, USA: U.S. Army Research Laboratory, 2015, pp. 263–283.
[8] J. K. Olsen, D. M. Belenky, V. Aleven, N. Rummel, J. Sewall, and M. Ringenberg, "Authoring tools for collaborative intelligent tutoring system environments," in *Proc. Int. Conf. Intell. Tutoring Syst.* Honolulu, HI, USA: Springer, 2014, pp. 523–528.
[9] A. Kerry, R. Ellis, and S. Bull, "Conversational agents in e-learning," in *Applications and Innovations in Intelligent Systems XVI*. Cambridge, U.K.: Springer, 2009, pp. 169–182.
[10] V. L. Rubin, Y. Chen, and L. M. Thorimbert, "Artificially intelligent conversational agents in libraries," *Library Hi Tech*, vol. 28, no. 4, pp. 496–522, 2010.
[11] V. Aleven, B. M. Mclaren, J. Sewall, and K. R. Koedinger, "A new paradigm for intelligent tutoring systems: Example-tracing tutors," *Int. J. Artif. Intell. Educ.*, vol. 19, no. 2, pp. 105–154, 2009.
[12] R. Doe. (2017). *Google Blockly—A Visual Programming Editor*. Accessed: Jan. 2017. [Online]. Available: http://code.google.com/p/blockly
[13] J. Trower and J. Gray, "Blockly language creation and applications: Visual programming for media computation and bluetooth robotics control," in *Proc. 46th ACM Tech. Symp. Comput. Sci. Educ.*, 2015, p. 5.
[14] L. Pappano, "The year of the MOOC," The New York Times, 2012.
[15] J. Ma and J. V. Nickerson, "Hands-on, simulated, and remote laboratories: A comparative literature review," *ACM Comput. Surv.*, vol. 38, no. 3, p. 7, 2006.
[16] S. Dormido, "Control learning: Present and future," *Annu. Rev. Control*, vol. 28, no. 1, pp. 115–136, 2004.
[17] J. E. Froyd, P. C. Wankat, and K. A. Smith, "Five major shifts in 100 years of engineering education," in *Proc. IEEE*, vol. 100, pp. 1344–1360, May 2012, doi: 10.1109/JPROC.2012.2190167.
[18] T. de Jong, S. Sotiriou, and D. Gillet, "Innovations in STEM education: The go-lab federation of online labs," *Smart Learn. Environ.*, vol. 1, no. 1, p. 3, 2014.
[19] M. J. Rodríguez-Triana *et al.*, "Rich open educational resources for personal and inquiry learning: Agile creation, sharing and reuse in educational social media platforms," in *Proc. Int. Conf. Web Open Access Learn. (ICWOAL)*, 2014, pp. 1–6.
[20] D. Gillet, T. De Jong, S. Sotirou, and C. Salzmann, "Personalised learning spaces and federated online labs for STEM education at school," in *Proc. IEEE Global Eng. Educ. Conf. (EDUCON)*, Mar. 2013, pp. 769–773.
[21] S. Govaerts *et al.*, "Towards an online lab portal for inquiry-based STEM learning at school," in *Proc. Int. Conf. Web-Based Learn.* Kenting, Taiwan: Springer, 2013, pp. 244–253.
[22] L. Rodriguez-Gil *et al.*, "OpenSocial application builder and customizer for school teachers," in *Proc. IEEE 14th Int. Conf. Adv. Learn. Technol. (ICALT)*, Jul. 2014, pp. 31–33.
[23] M. Tawfik *et al.*, "Virtual instrument systems in reality (VISIR) for remote wiring and measurement of electronic circuits on breadboard," *IEEE Trans. Learn. Technol.*, vol. 6, no. 1, pp. 60–72, Jan./Mar. 2013.
[24] J. Lester, B. Mott, J. Rowe, and R. Taylor, "Principles for pedagogical agent authoring tools," in *Design Recommendations for Intelligent Tutoring Systems: Authoring Tools and Expert Modeling Techniques*. Ann Arbor, MI, USA: Robert Sottilare, 2015, p. 151.
[25] J. K. Olsen, D. M. Belenky, V. Aleven, and N. Rummel, "Intelligent tutoring systems for collaborative learning: Enhancements to authoring tools," in *Proc. Int. Conf. Artif. Intell. Educ.* Memphis, TN, USA: Springer, 2013, pp. 900–903.
[26] R. Kumar and C. P. Rose, "Architecture for building conversational agents that support collaborative learning," *IEEE Trans. Learn. Technol.*, vol. 4, no. 1, pp. 21–34, Jan./Mar. 2011.
[27] C. Ray and S. Gilbert, "Bringing authoring tools for intelligent tutoring systems and serious games closer together: Integrating GIFT with the unity game engine," in *Proc. AIED Workshops*, vol. 7, 2013, p. 37.
[28] N. Matsuda, W. W. Cohen, and K. R. Koedinger, "Teaching the teacher: Tutoring simstudent leads to more effective cognitive tutor authoring," *Int. J. Artif. Intell. Educ.*, vol. 25, no. 1, pp. 1–34, 2015.
[29] M. Virvou and E. Alepis, "Mobile educational features in authoring tools for personalised tutoring," *Comput. Educ.*, vol. 44, no. 1, pp. 53–68, 2005.
[30] V. Aleven, R. Baker, Y. Wang, J. Sewall, and O. Popescu, "Bringing non-programmer authoring of intelligent tutors to MOOCs," in *Proc. 3rd ACM Conf. Learn. Scale*, 2016, pp. 313–316.
[31] H. C. Lane, M. G. Core, M. J. Hays, D. Auerbach, and M. Rosenberg, "Situated pedagogical authoring: Authoring intelligent tutors from a student's perspective," in *Proc. Int. Conf. Artif. Intell. Educ.* Springer, 2015, pp. 195–204.

[32] K. R. Koedinger, V. Aleven, N. Heffernan, B. McLaren, and M. Hockenberry, "Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration," in *Proc. Int. Conf. Intell. Tutoring Syst.* Maceió, Brazil: Springer, 2004, pp. 162–174.
[33] T. Bickmore and L. Ring, "Making it personal: End-user authoring of health narratives delivered by virtual agents," in *Proc. Int. Conf. Intell. Virtual Agents*. Springer, 2010, pp. 399–405.
[34] V. Aleven, B. M. McLaren, J. Sewall, and K. R. Koedinger, "The cognitive tutor authoring tools (CTAT): Preliminary evaluation of efficiency gains," in *Proc. 8th Int. Conf. Intell. Tutoring Syst. (ITS)*. Berlin, Germany: Springer, 2006, pp. 61–70.
[35] R. A. Sottilare, K. W. Brawner, B. S. Goldberg, and H. K. Holden, "The generalized intelligent framework for tutoring (GIFT)," U.S. Army Res. Lab. Hum. Res. Eng. Directorate, Orlando, FL, USA, Tech. Rep., 2012, doi: 10.13140/2.1.1629.6003.
[36] S. B. Blessing, "A programming by demonstration authoring tool for model-tracing tutors," in *Authoring Tools for Advanced Technology Learning Environments*. Springer, 2003, pp. 93–119.
[37] V. Aleven *et al.*, "Example-tracing tutors: Intelligent tutor development for non-programmers," *Int. J. Artif. Intell. Educ.*, vol. 26, no. 1, pp. 224–269, 2016.
[38] B. A. Shawar and E. Atwell, "A chatbot as a novel corpus visualization tool," in *Proc. LREC*, 2004.
[39] J. Feng, S. Bangalore, and M. Rahim, "WebTalk: Mining Websites for automatically building dialog systems," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand. (ASRU)*, Dec. 2003, pp. 168–173.
[40] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch programming language and environment," *ACM Trans. Comput. Educ.*, vol. 10, no. 4, p. 16, 2010.
[41] G. W. Johnson, *LabVIEW Graphical Programming*. New York, NY, USA: McGraw-Hill, 1997.
[42] J.-M. Sáez-López, M. Román-González, and E. Vázquez-Cano, "Visual programming languages integrated across the curriculum in elementary school: A two year case study using 'Scratch' in five schools," *Comput. Educ.*, vol. 97, pp. 129–141, Jun. 2016.
[43] W.-H. Kuan, C.-H. Tseng, S. Chen, and C.-C. Wong, "Development of a computer-assisted instrumentation curriculum for physics students: Using LabVIEW and Arduino platform," *J. Sci. Educ. Technol.*, vol. 25, no. 3, pp. 427–438, 2016.
[44] F. Kalelioğlu, "A new way of teaching programming skills to K-12 students: Code.org," *Comput. Hum. Behav.*, vol. 52, pp. 200–210, Nov. 2015.
[45] D. Kumar, "Digital playgrounds for early computing education," *ACM Inroads*, vol. 5, no. 1, pp. 20–21, 2014.
[46] I. Iturrate *et al.*, "A mobile robot platform for open learning based on serious games and remote laboratories," in *Proc. 1st Int. Conf. Portuguese Soc. Eng. Educ. (CISPEE)*, 2013, pp. 1–7.
[47] M. Á. Serna, C. J. Sreenan, and S. Fedor, "A visual programming framework for wireless sensor networks in smart home applications," in *Proc. IEEE 10th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process. (ISSNIP)*, Apr. 2015, pp. 1–6.
[48] J. Wiriyakul and T. Senivongse, "A visual editor for language-independent scripting for BPMN modeling," in *Proc. 12th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, Jul. 2015, pp. 156–161.
[49] C. E. Wieman, W. K. Adams, and K. K. Perkins, "PhET: Simulations that enhance learning," *Science*, vol. 322, no. 5902, pp. 682–683, 2008.
[50] M. Kaluz, J. Garcia-Zubia, M. Fikar, and L. Cirka, "A flexible and configurable architecture for automatic control remote laboratories," *IEEE Trans. Learn. Technol.*, vol. 8, no. 3, pp. 299–310, Jul. 2015.
[51] H. Crompton, D. Burke, K. H. Gregory, and C. Gräbe, "The use of mobile learning in science: A systematic review," *J. Sci. Educ. Technol.*, vol. 25, no. 2, pp. 149–160, 2016.
[52] L. J. Couse and D. W. Chen, "A tablet computer for young children? Exploring its viability for early childhood education," *J. Res. Technol. Educ.*, vol. 43, no. 1, pp. 75–96, 2010.
[53] S. N. Şad and Ö. Göktaş, "Preservice teachers' perceptions about using mobile phones and laptops in education as mobile learning tools," *Brit. J. Educ. Technol.*, vol. 45, no. 4, pp. 606–618, 2014.
[54] D. G. de la Iglesia, J. F. Calderón, D. Weyns, M. Milrad, and M. Nussbaum, "A self-adaptive multi-agent system approach for collaborative mobile learning," *IEEE Trans. Learn. Technol.*, vol. 8, no. 2, pp. 158–172, Apr. 2015.
[55] McAfee, "McAfee labs threats report: May 2015," Intel Secur., Santa Clara, CA, USA, Tech. Rep., May 2015.
[56] J. Chacon, H. Vargas, G. Farias, J. Sanchez, and S. Dormido, "EJS, JIL server, and LabVIEW: An architecture for rapid development of remote labs," *IEEE Trans. Learn. Technol.*, vol. 8, no. 4, pp. 393–401, Oct./Dec. 2015.
[57] D. Lowe, S. Murray, E. Lindsay, and D. Liu, "Evolving remote laboratory architectures to leverage emerging Internet technologies," *IEEE Trans. Learn. technol.*, vol. 2, no. 4, pp. 289–294, Oct. 2009.
[58] L. Rodriguez-Gil, J. Garcia-Zubia, P. Orduna, and D. Lopez-de Ipina, "Towards new multiplatform hybrid online laboratory models," *IEEE Trans. Learn. Technol.*, vol. 10, no. 3, pp. 318–330, Jul./Sep. 2016.
[59] J. Cassell, "Embodied conversational interface agents," *Commun. ACM*, vol. 43, no. 4, pp. 70–78, 2000.
[60] C. Marrin, "WebGL specification," Khronos WebGL Working Group, Tech. Rep., 2011.
[61] K. Finstad, "The usability metric for user experience," *Interacting Comput.*, vol. 22, no. 5, pp. 323–327, 2010.
[62] M. I. Berkman and D. Karahoca, "Re-assessing the usability metric for user experience (UMUX) scale," *J. Usability Stud.*, vol. 11, no. 3, pp. 89–109, 2016.
[63] International Organization for Standardization, 9241-11:1998, 1998, "Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Guidance on Usability."
[64] J. Brooke, "SUS-A quick and dirty usability scale," *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.
[65] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Int. J. Hum.–Comput. Interact.*, vol. 24, no. 6, pp. 574–594, 2008.

LUIS RODRÍGUEZ-GIL received the dual degree in computer engineering and industrial organization engineering in 2013, the M.Sc. degree in information security in 2014, and the Ph.D. degree in computer science from the University of Deusto in 2017. During his Ph.D., he co-founded the LabsLand remote labs company, on which he is currently a full-time as its CTO. Since 2009, he has been a part of the WebLab-Deusto Research Group, collaborating in the development of the WebLab-Deusto RLMS. Throughout this time, he has authored various peer-reviewed publications and contributed to several open source projects.

JAVIER GARCÍA-ZUBIA (M'08–SM'11) received the Ph.D. degree in computer science from the University of Deusto, Spain. He is a Full Professor with the Faculty of Engineering, University of Deusto. He is the Leader of the WebLab-Deusto Research Group. His research interest is focused on remote laboratory design, implementation, and evaluation.

PABLO ORDUÑA (M'05) received the degree in computer engineering and the Ph.D. degree from the University of Deusto in 2007 and 2013, respectively. During his Ph.D., he was a Visiting Researcher twice for six weeks each, in the MIT CECI in 2011 and UNED DIEEC in 2012. He has also attended two programs for entrepreneurship training at Singularity University: Global Solutions Program and Launchpad. Since 2004, he has also been involved in the WebLab-Deusto Research Group, leading the design and development of WebLab-Deusto, and a Later Researcher and the Project Manager of the MORElab (DeustoTech Internet) until 2017. He is the CEO at LabsLand (spin-off of the WebLab-Deusto project), and an external collaborator at DeustoTech.

AITOR VILLAR-MARTINEZ received the B.Sc. and M.Sc. degrees in telecommunication engineering in telecommunication engineering from the University of Deusto in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with a focus on working and researching about remote laboratories and their technologies. Since 2014, he has collaborated with several university projects, such as the Smart Moto Challenge in 2015 and 2016 editions. In 2017, he started working at LabsLand (spin-off of the University of Deusto and the WebLab-Deusto Project) as a Hardware/Software Developer.

DIEGO LÓPEZ-DE-IPIÑA received the Ph.D. degree from the University of Cambridge in 2002. Responsible for several modules in the B.Sc. and M.Sc. degrees in computer engineering, he is interested in pervasive computing, IoT, semantic service middleware, open linked data, and social data mining. He is an Associate Professor and a P.R. of the MORElab Group and the Director of the DeustoTech Internet Unit, and of the Ph.D. Program within the Faculty of Engineering, University of Deusto. He is taking and has taken part in several big consortium-based research European (IES CITIES, MUGGES, SONOPA, CBDP, GO-LAB, and LifeWear) and Spanish projects, and has more than 70 publications in relevant international conference and journals, including more than 25 JCR-indexed articles.