Developing a Mobile System for Natural Language Grammar Acquisition

Marina Purgina Maxim Mozgovoy Vitaly Klyuev

The University of Aizu
Aizuwakamatsu City, Japan
{d8172102, mozgovoy, vkluev}@u-aizu.ac.jp

***Abstract*—The architectural and user interface patterns of mobile applications are well established for most popular software types. However, it is still challenging to design a mobile application for a use scenario beyond typical daily tasks. In this paper, we describe the challenges and design decision of mobile WordBricks software — a virtual lab-like environment for natural language grammar acquisition. The flexibility of natural language grammar and complexity of visual representation of syntactical word relationships as well as specific pedagogical requirements required flexible system design decisions. We base the system on a combination of dynamic GUI elements creation and XML description of graphical scene contents. The system was successfully tested in a real classroom environment, and proved demonstrated high flexibility and maintainability.**

Keywords—mobile-assisted language learning, intelligent systems, virtual labs, Android application

I. INTRODUCTION

Nowadays, computer-assisted language learning (CALL) instruments are very popular and widespread. It is generally recognized that the appearance of many tools for language learning as well as the powerful electronic dictionaries greatly enriches the practice of teaching and learning [1].

However, most software tools in this area are in fact the electronic versions of textbooks and the already existing technologies of language learning. For example, according to PC Magazine [2] most of popular CALL applications have the following capabilities: lessons with multimedia content, word-based memory games, sets of flashcards, online tutoring, and pronunciation training.

This situation greatly differs from the use of technology found in STEM (Science, Technology, Engineering, Mathematics) education. Modern educational software goes far beyond digitalized learning materials and virtual laboratories are no longer a novelty [3]. For example, the instruments that enable users to perform scientific experiments on a computer screen without necessity to setup a real lab are emerging in the fields of physics and chemistry: Open Source Physics or ChemCollective. In programming, there are systems like Jeliot3 and JFLAP that visualize computer programs and demonstrate algorithms in the form of easily understandable interactive animations. These tools were already successfully tested on several generations of students.

Accordingly, appreciating the positive effects of these systems on the learning, as well as increasing the motivation of students, we started to develop WordBricks system [4], which is an attempt to introduce 'virtual lab-like experience' into language learning.

The system focuses on grammar acquisition, a topic rarely explored in CALL instruments beyond the level of traditional quizzes and similar exercises.

The main aim of the software is to let the user to combine words and phrases into grammatically correct constructions, thus explore the possibilities of natural language grammar. This system can be used in two modes. In the free mode, WordBricks simply gives the user a chance to experiment with any words and word combinations to see which constructions are admissible in the target language. In the lesson mode, WordBricks displays a set of predefined constructions that have to be combined by the user into correct sentences.

The design of WordBricks was greatly inspired by Scratch [5], a system for visual programming, aimed at beginners. In Scratch, individual elements of a computer program are represented with colorful blocks that can be connected together if and only if the resulting structure is syntactically correct.

While developing WordBricks, we concentrated on the following challenges:

- The system should reliably identify admissible word combinations and thus serve as a solid aid to the learners.
- The system should be intuitive and easy to use. It should be consistent with pedagogical goals. It should be extensible and adaptable in order to incorporate new assignments and use cases.
- Due to the growing popularity of mobile platforms and mobile-assisted language learning (MALL), the system should be available on a mobile platform, and support simple over-the-air update capabilities to reflect changes in course materials.
- The system should be flexible enough to support a large variety of natural language constructions and thus be relatively easily adaptable for new natural languages.

In this regard, it should be noted that the current version of WordBricks is a working prototype with limited functionality. Nevertheless, preliminary testing of WordBriks in real classrooms at the University of Aizu showed that the students who used WordBricks scored higher on the exam tests [7]. The testing was implemented with the following procedure. Two sets of pre- and post-tests were used to identify differences of participants' English grammar skills. The pre- and post-test were executed before and after each lesson, covering units 69 and 70 of the course textbook [6] (see Table I).

TABLE I. DESCRIPTIVE STATISTICS OF THE PRE- AND POST-TESTS

Unit	Test	Group	*Number of participants*	*Mean (M)*	*Standard deviation (SD)*
69	Pre-test	WB	10	15.90	4.43
		Control	11	15.18	5.04
	Post-test	WB	10	24.20	4.02
		Control	11	21.00	5.80
70	Pre-test	WB	10	4.20	2.57
		Control	11	6.00	2.72
	Post-test	WB	10	11.60	2.84
		Control	11	9.18	4.17

Note: WB = WordBricks (experimental) group

Descriptive statistics of the tests from the chapter 69 indicate that the experimental (WB) group and the control group had similar mean values in the pre-test. In the post-test, the WB group performed a little better than the control group. Unlike chapter 69, WordBricks users scored lower than the control group in the pre-test of chapter 70. However, they scored higher than the control group in post-test of chapter 70.

These experiments were performed with small groups of students, but they met our expectations, so we are planning to extend classroom evaluation.

II. SYSTEM DESCRIPTION

We designed the architecture of the mobile application by following the process suggested in [8]. Android was the operating system of choice for WordBricks due to its wide availability and openness [9].

A. WordBricks Package Structure

WordBricks is developed using the Java programming language. It relies on the standard functionality of the Android framework. Thereby, the application consists of the following components:

- Java classes that are subclasses of the main Android SDK classes (`View`, `Activity`) and Java classes that have no Android SDK ancestors, i.e. helper classes for implementation of the application logic;
- the Android Manifest file;
- application resources and XML definitions of application GUI layouts;
- exercise descriptions (XML files).

In terms of handling relationships between GUI and a logic supporting GUI, the application architecture follows Model-View-ViewModel architectural pattern [10].

B. Implementation of core functionality

In addition to "virtual lab" experience, WorkBricks is intended to contribute to the overall gamification of the study process. Therefore, we tried to visualize the grammar through plain and simple forms as much as possible. From a technical point of view the above problem can be logically divided into two sub-tasks: visualization of syntactic forms and semantic description of grammar exercises.

a) Visualizing syntactic forms. Android application GUI is a tree of instances of `View` subclasses, i.e. GUI widgets [11]. The `View` class is from the Android framework and it is used for all Android GUI widgets. In the system, every element of a sentence is represented with one brick of a certain shape and color having a set of connectors for other sentence elements. These elements may vary from one exercise to another, and may consist of individual words or punctuation symbols, or arbitrary phrase fragments. According to the Android framework, `Brick` objects displayed on WordBricks screen are defined as subclasses of the View class. The GUI tree is normally defined with XML layout files, and at the runtime expanded automatically into the tree of corresponding objects. However, in our case our custom `View` of `ViewBrick` is created and added to the existing GUI layer at runtime. This allows the user to create and delete them at any time.

Brick width is calculated in accordance with a set of parameters. These parameters are word length, empty connectors and non-empty connectors if they are presented. Therefore, brick width is constantly changing in the process of sentences construction, as illustrated in Fig. 1.

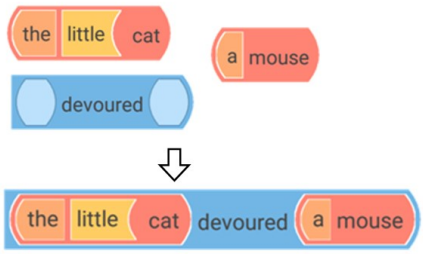

Fig. 1. Connecting bricks into a sentence

b) Describing grammar exercises semantics. As mentioned above, the content of each individual brick varies from exerciese to exercise. Therefore, each exercise needs its own XML-defined set of word bricks. In other words, each exercise is specified by an XML file with an independent 'vocabulary' of bricks. In fact, this specification of the bricks is a description of the semantics of words and semantic description of its use. The polysemy nature of words can be defined with multiple XML sections.

For example, the bricks “ball” as a noun and as an adjective (“ball game”) can be described with XML as follows:

```
<brick word=" ball"
    part_of_speech="Noun phrase"
    case = "common"
    person = "third"
    number = "singular" >
    <item type="brick connector"
        value="Determiner"/>
    <item type="brick connector"
        value="Adjective phrase"/>
    <item type="text" value=" ball"/>
</brick>

<brick word="ball"
    part_of_speech ="Adjective phrase">
    <item type="text" value="ball"/>
</brick>
```

In this example, the brick contains one word form. Its attributes require to form a brick view, as well as a further compilation of sentences. A list of connectors also is in place with the attributes.

The user can select words from the vocabulary exercises and add them to the screen (item (1) on the Fig. 2). Descriptions of the corresponding bricks are retrieved from the XML exercise file (item (2) on the Fig. 2). The descriptions are then parsed and stored in the class `Brick` instance (item (3) on the Fig. 2). After that, a new element of `BrickView` class is added and displayed on the screen (item (4) on the Fig. 2).

D. Design of WordBricks GUI

The Start Activity (item (5) on the Fig. 2) is a list of chapters of the textbook (we used the textbook [6]). After selecting the exercise, the user goes to the main screen of the application. The user interface of the main Activity (item (6) on the Fig. 2) is a standard Navigation Drawer (item (7) on the Fig. 2) recommended from the official Google documentation [12]. It is used for switching fragments of the working area (item (8) on the Fig. 2). In WordBricks, the Drawer on the left part of the screen allows the user to switch between the exercises.

Working area of the Blackboard is implemented through a Fragment class from the Android framework.

New bricks can be added using the floating action button, located in the bottom right corner of the application screen.

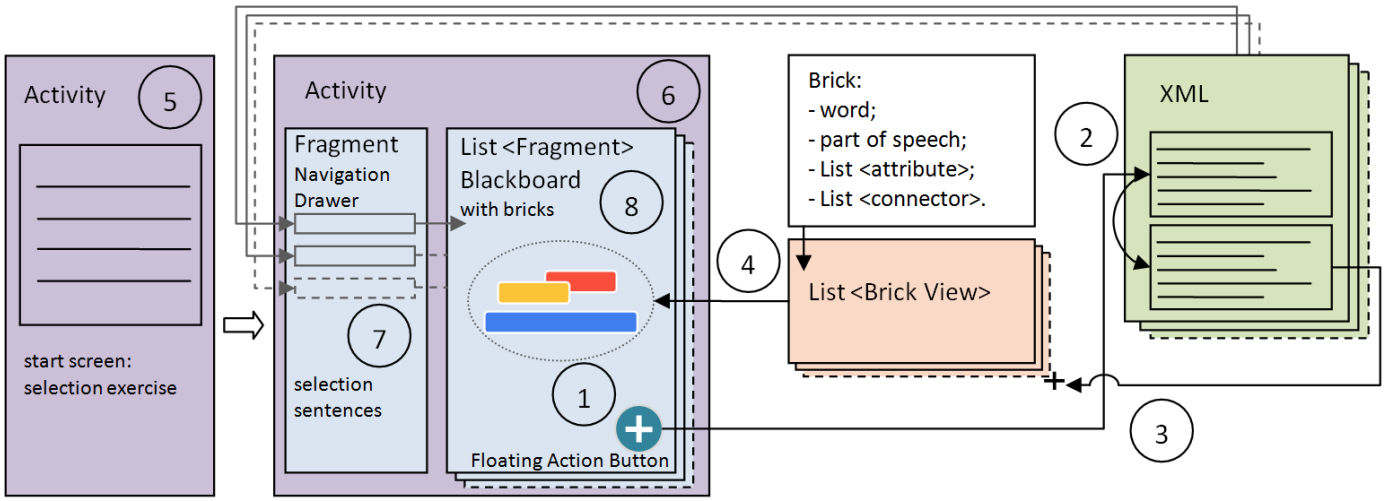

Fig. 2. Scheme of the structure and interaction of components of WordBricks

When the user attempts to insert a brick into a connector, the system checks whether the connector’s part of speech and the brick’s part of speech do match. If parts of speech are the same, attributes are being checked next. If the attribute list of the connector includes the attribute list of the brick, then the brick is inserted into the connector successfully. Moreover, it does not matter if the brick has a dependent brick or not. The order of brick connection actions is also not important.

C. Interactions between WordBricks components

The scheme of the general structure and interaction of components of the application in a simplified form is shown in Fig. 2. The main steps of interaction are as follows.

Editing or deleting a block is performed by double-clicking on the block. Also, this can be done via the Action Bar.

a) Blackboard: The user of the WordBricks can move bricks and create sentences by employing a drag-and-drop interface. Bricks can be connected in any order. GUI contains color hints and pop-up information about words at the bottom of the screen for convenience of the user.

b) Responsive design: Sizes of bricks are calculated dynamically according to the screen resolution. Thus, the bricks look normally on the screens of any screen size and density. Also, the application has an ability to zoom the screen of the Blackboard for individual user settings. However, an horizontal orientation is preferred since a relatively long sentences hardly fit the screen.

III. Discussion

Designing a virtual lab for language learning is a very challenging task. There is no generally accepted way of teaching languages: numerous distinct strategies coexist, and they greatly vary in methods and materials, and emphasize various aspects of language use. The materials must be adapted to the learners' proficiency level, and might be specifically tailored for the needs of the native speakers of a particular language. The structure of natural languages is also very diverse, and might greatly vary, causing difficulties both for learners and MALL software developers. Furthermore, there is no established way of using virtual lab-like experimental software, since most teaching methods rely on traditional learning activities, such as reading, writing, speaking and listening, and assume that learner feedback is provided via teacher-student interaction.

Therefore, the architecture of a virtual language lab must be flexible enough to support a variety of pedagogical needs and language structures. It is nearly impossible to anticipate all potential use cases and needs of a particular educational setup. So we have to aim for maximum flexibility and language independence. In the present version of WordBricks this goal is accomplished by encoding all brick features in XML documents. Brick shapes, content, and configuration of connectors are fully defined in XML. These definitions represent semantics of words and semantics of usage in the sentences. In its turn, XML files are currently crated manually, but we believe that some part of this work can be automated with natural language processing software.

A number of technical and pedagogical challenges is caused by the distinctive features of mobile platforms. The small mobile screen cannot display the complete set of brick attributes, so we had to intentionally hide some of them, and make the remaining attributes easy to see and understand. We also had to support numerous possible user actions via limited tap interface, and ensure proper auto-positioning and sizing of bricks.

Preliminary experiments show that the users appreciate our efforts, and speak in favor of the current design decisions. However, we should note that the ready experimental data is still very limited. We tested the system in the classroom using a number of textbook exercises, and thus we cannot yet prove the robustness of WordBrick on large blocks of educational materials, complex and diverse grammatical phenomena, and effectiveness of the software for teaching. These topics will be addressed in our upcoming research initiatives.

IV. Conclusion

Designing a sound experimental MALL system is a challenging endeavor, given the variety of use cases, natural language complexity, and the ongoing evolution of the software to respond to changing user needs. Proper software architecture is a notable aid in this work, reducing the need of rewriting code and fine-tuning it to adapt to new use scenarios. Our current experiments show that the chosen approach supports nearly all natural language elements we had to implement for the classroom use, and ensures flexibility and adaptability of WordBricks.

Acknowledgement

WordBricks project is supported by the JSPS KAKENHI Grant #25330410.

References

[1] G. Stanley and S. Thornbury, *Language learning with technology: Ideas for integrating technology in the language classroom*, 2013.

[2] J. Duffy, "The Best Language-Learning Software for 2015," *PC Magazine*, 02 Sep, 2015, http://www.pcmag.com/article2/0,2817,2381904,00.asp.

[3] V. Potkonjak, M. Gardner, V. Callaghan, P. Mattila, C. Guetl, V. M. Petrović, and K. Jovanović, "Virtual laboratories for education in science, technology, and engineering: A review," *Computers & Education*, vol. 95, pp. 309–327, 2016.

[4] M. Mozgovoy and R. Efimov, "WordBricks: a virtual language lab inspired by Scratch environment and dependency grammars," *Human-centric Computing and Information Sciences*, vol. 3, no. 1, pp. 1–9, 2013.

[5] M. Resnick, B. Silverman, Y. Kafai, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, and J. Silver, "Scratch," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.

[6] R. Murphy, *English Grammar in Use, 4th Ed*: Cambridge University Press, 2012.

[7] M. Park, M. Purgina, and M. Mozgovoy, "Learning English Grammar with WordBricks: Classroom Experience," in *Proceecings of the 2016 IEEE International Conference on Teaching and Learning in Education*, 2016.

[8] Salazar, Felix Javier Acero and M. Brambilla, "Tailoring Software Architecture Concepts and Process for Mobile Application Development," in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, New York, NY, USA: ACM, 2015, pp. 21–24.

[9] B. A. Lucini, T. Hatt, C. Gardner, and B. Pon, *Mobile platform wars*: GSMA Intelligence, 2014.

[10] J. Smith, "WPF Apps With The Model-View-ViewModel Design Pattern," *MSDN Magazine*, Feb. 2009.

[11] Google Inc. and the Open Handset Alliance, *API Guides: UI Overview.* Available: http://developer.android.com/guide/topics/ui/overview.html (2016, Mar. 26).

[12] Google Inc, *Material Design Specification. Patterns – Navigation.* Available: https://www.google.com/design/spec/patterns/navigation.html (2016, Mar. 26).