International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:9, No:1, 2015

Models and Metamodels for Computer-Assisted Natural Language Grammar Learning

Evgeny Pyshkin, Maxim Mozgovoy, Vladislav Volkov

***Abstract*—**The paper follows a discourse on computer-assisted language learning. We examine problems of foreign language teaching and learning and introduce a metamodel that can be used to define learning models of language grammar structures in order to support teacher/student interaction. Special attention is paid to the concept of a virtual language lab. Our approach to language education assumes to encourage learners to experiment with a language and to learn by discovering patterns of grammatically correct structures created and managed by a language expert.

***Keywords*—**Computer-assisted instruction, Language learning, Natural language grammar models, HCI.

I. Introduction

COMPUTING environments affect learning process in a wide range of disciplines from humanities to technical sciences. As people get exposed to computer and software technology that surrounds us in everyday life, the term *"computer-assisted"* sounds like a dissonance. Let's take an area of computer-assisted language learning (CALL) as an example. In [1] the authors sarcastically note that in contrast to computer-assisted learning, "we do not refer to 'ballpoint pen-assisted writing' or 'car-assisted traveling'", and that the common definitions of CALL merely state the use of computers as the most important trait of this phenomenon. Despite all dramatic changes in computer facilities since the time when the basic CALL concepts were established , Beatty still concedes that current CALL is "an *amorphous or unstructured* discipline, constantly evolving both in terms of pedagogy and technological advances in hardware and software" (italics ours) [2].

The development of computer-based learning environments is intended to improve the following characteristics of the language learning process:

- **Learning Performance**: A process of delivering new knowledge and skills should reduce the time learners spend and facilitate access to educational resources.
- **Learning Efficiency**: Knowledge and skills acquired by learners should be long-lasting. Learners should be able to focus on topics that are especially important for them.
- **Accessibility**: Learners should be able to access new materials and interactive instruments that might be unavailable or hardly available without computer technology.
- **Flexibility**: Learners should be able to access learning tools in any time and from any location.
- **Organization**: Teachers should be able to distribute study materials easier; distance learning techniques should be available to enforce communication with language experts and other learners.
- **Motivation**: As a result of aforementioned advantages learners become better motivated to continue their studies and to be deeper involved into the process.

The present role of computer technologies in education is mainly focused on extending the boundaries of the classical learning process: computers should become integrated and (in a sense) invisible components of a learning system.

The maturity of consumer technologies affected strongly the CALL development especially in ways of communication activities implementation:

- Technological innovations encourage teachers and learners to communicate in the ways never available before.
- In addition to (or often instead of) specialized CALL software like tutorials, games, simulators or problem solvers, CALL uses general consumer communication tools and applications (which are not kind of teaching software).
- Authentic materials (created by native speakers) are easier accessible via computer communication: to check the phrase correctness learners rather use Google instead of language tutorials or dictionaries.

Garrett concluded that the new demands on language education constitute a powerful set of reasons to rethink grammar CALL [3]. Despite the fact that one of primary goals of language learning is to improve learners' communicative competence, learning grammar is still an essential part of language apprehension. In written language, learning a language grammar is one of areas where a typical student activity is limited to following inflexible learning tutorials, while having little or no ways to experiment with the language in a similar way as students do in natural and technical sciences [4]. We know many examples of virtual labs implemented for the academic courses in physics [5], [6], chemistry [7]-[9], medicine [10], [11], control systems [12]-[14], etc. There are many reasons to use such sorts of labs instead of using real equipment: virtual labs require less space, they can be easily installed or deployed, often they can be accessed remotely, they are safe in regards to user health and equipment integrity, they can be easily reconfigured, etc. In case of language learning, we can consider any system that

Evgeny Pyshkin is with St. Petersburg State Polytechnical University, 21 Polytechnicheskaya st., St. Petersburg, 195251 Russia (phone: +7-812-297-4218; fax: +7-812-297-6780; e-mail: pyshkin@ icc.spbstu.ru).

Maxim Mozgovoy is with the University of Aizu, Tsuruga, Ikki-machi, Aizu-Wakamatsu, Fukushima, 965-8580 Japan (phone: +81-242-37-2664, e-mail: mozgovoy@u-aizu.ac.jp).

Vladislav Volkov was with St. Petersburg State Polytechnical University as a master degree student (vladislav.volkov@yadumay.ru).

supports independent student activities with machine-provided feedback as a similar "virtual lab" that can encourage freer experimentation with language elements and partially substitute teachers.

II. Related Work

Virtual labs can also support problem-based learning (PBL). Within this educational philosophy, instead of doing artificial exercises, students are encouraged to work on real subject domain problems. Interestingly, one of the first PBL implementations was in medicine education: students worked on problems from the domain of real clinical experience and tried to find proper knowledge to solve practical tasks [15]. In [16] the authors cite the example of the assisted lung ventilation control system: the students had to design the system by using modeling tools, and experimented with real medical equipment where the control system has been embedded. Thus, PBL was proposed as one of ways to bridge the gap between learning environments and real-life projects. We believe that for a case of language learning one of reasons to improve CALL technology is to bridge the similar gap between classroom language study and real-life communication.

Let's reconsider an obvious idea to visualize basic language grammar constructions in a "teachable" way. In language tutorials such visual models are mostly "static": learners are unable to use them interactively. Why not to provide learners with a possibility to manipulate such constructions (in order to do exercises or to experiment with them freely)? We think that visual grammar-based constructions simplify learning process for beginners. To decrease complexity of exercises, it is possible to use certain techniques of producing easy-to-read materials [17]. It is also advisable to base learner-oriented grammar constructions on restricted dictionary, simplified in order to educate people with preliminary low literacy. As noted by Robin, in language teaching the best solutions aren't those that implement some methods better, but those that conduct the learning process in a learner's own style [18].

One of the possible approaches to create a "virtual language lab" is explored in the experimental system WordBricks described in our earlier works [1], [4]. WordBricks is a virtual language playground inspired by Scratch programming environment [19]. The Scratch approach combines several important concepts that simplify teaching programming. They include visual flowchart-style code representation, event-based and multithreaded execution model, and, what is the most important in our case, the absence of error messages [20]. We can draw an analogy with LEGO bricks: individual parts can be connected only in a restricted number of possible ways: all possible combinations of brick linking can be easily found with trial and error (see Fig. 1).

WordBricks (see Fig. 2) follows the same idea: while the users are free to experiment with any language structures, the environment makes it impossible to create ungrammatical language constructions. Furthermore, open (but directed in the right way) experiments are possible without traditional grammar checking technologies [4].

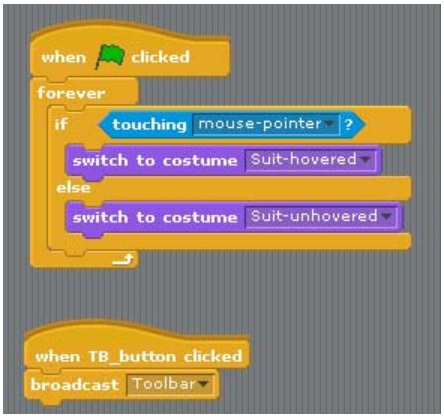

Fig. 1 Scratch visual elements

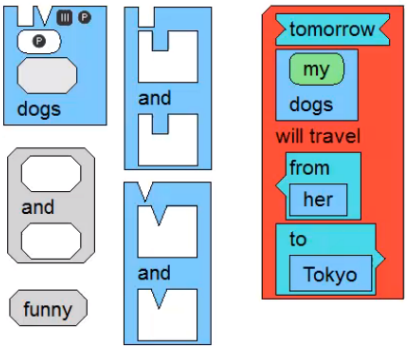

Fig. 2 WordBrick: elements and structures

While being quite a simple instrument, WordBricks still has to address a number of important problems, relevant to many CALL systems. Perhaps, the simplest variation of a "virtual language lab" is represented with spell- and grammar-checking software. A student can type literally any sentences and check whether they are considered grammatical. In practice, it turns out that general-purpose programs, such as the built-in MS Word's grammar checker, are not designed for CALL: they miss many mistakes and do not provide a comprehensive feedback. Specialized tools, such as Robo-Sensei [21], have to limit possible user inputs to certain known cases in order to provide precise explanations to students. So conventional grammar checking instruments are still unable to reliably identify the errors in the texts written by the beginners, and their feedback generation capabilities are not sufficient for students. The WordBricks projects started as an attempt to answer the following question: *what kind of valuable functionality can be reliably implemented with the current state-of-the art technologies?* Since natural language processing methods are not reliable enough (this is especially true for ungrammatical sentences, created by students), they had to be abandoned. This means that the students have to be *forced* to create grammatically correct phrases from the very beginning. Scratch environment clearly demonstrates the viability of this approach in programming: the possibility to avoid syntax errors helps novice programmers to concentrate on code design. However, we believe that another important

aspect of such visual programming is underestimated: by looking at shapes and connectors, the students understand the inner structure of code.

While the syntax of computer programs is often intentionally designed to be simple, natural language constructions are more challenging to analyze, and thus their understanding needs to be supported with learning aids. The views on teaching grammar structures in the classroom differ, but still certain simplified schemes and explanations are almost universally used in language learning. So, in case of WordBricks, there are two major advantages of using shaped Scratch-like bricks: 1) the system does not need to rely on natural language processing; 2) the students are exposed to the visual representation of sentence structure, which presumably contributes to their mental model of language.

The analysis of existing computerized language learning environments (like PLATO project [22] or Athena Language Learning Project [23]) leads us to the belief that the goal of creating a CALL system is twofold: first, to support learning process, and second, to gather research data necessary for further improvements of language learning concepts, and for the design of future language learning tools. In the context of supporting the language grammar learning process, we examine visual models that we consider relevant and useful both to overcome difficulties that novices have while learning languages and to implement learning environments in order to have a possibility to get feedback from students and teachers after experimenting with them.

III. Structures for Modeling Language Sentences

Language grammar is a study of how words combine to form sentences and what structural relationships in a language are. When language grammar constructions are explained in school, typically no formal models of syntax like constituency or dependency grammars are used. Instead, some typical structures for different phrases are demonstrated. This approach agrees with the constructivist views on language education, and partially inherits the way children learn their first language using almost no direct grammar rules, but deducing them implicitly during the learning process [24].

Conventional school practices are similar to the use of naïve pedagogical grammars (as noted as far as in 1991 by Fum et al. [25]). Such grammars comprise the knowledge derived from textbooks and teacher experience. We believe that computer technologies can enrich this process by allowing students to experiment with words and structures. We believe that computer assisted language learning tools are aimed to provide an environment where the learners are building their own mental models of the language they learn step by step. In many disciplines related to languages (including software engineering and programming), students' capability to create a clear mental model of a studied concept (e.g. of a natural or a programming language) has crucial importance. As mentioned by Milne and Rowe for the case of programming, the absence of such a mental model is considered as one of usual difficulties in learning programming [26]. As well as for the case of software engineering, in natural language education visualization of data and control structures is one of the known ways to overcome such difficulties.

We propose the concept of a learning system aimed to support basic scenarios of grammar teaching and learning by using visual modeling of language grammar structures. We pay special attention to deal with teacher's and learner's views that differ. As a language expert, a teacher can create new models representing grammar constructions, and edit the models taken from the knowledge base. Then the models are used to create annotated examples and problems for students. Here are the examples of typical study problems:

- Having the structure definition, construct examples by using dictionary words (they can be organized in groups such as objects, actions, properties, and so on).
- Construct proper sentences by using most words from the bag of words with or without explicit reference to a grammar pattern.
- Select the correct forms for missing words in the given sentence.
- Find grammar mistakes, and correct the sentence.

A. Teacher's Perspective: Metamodel, Model, Implementation, and Examples

While lexical categories may significantly differ in different languages, there are many universal categories [27]. We tried to express the common elements of grammar learning constructions as Fig. 3 shows.

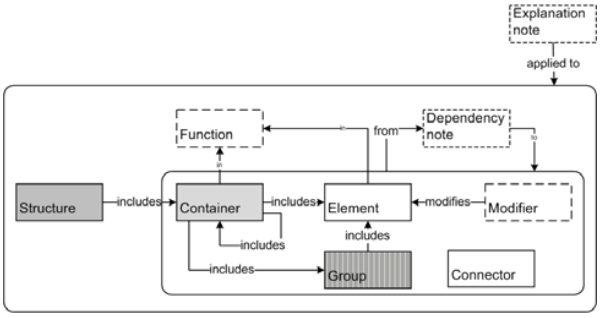

Fig. 3 Grammar construction metamodel

Starting out from phrase-structure grammars where a sentence is analyzed into a linearly concatenated sequence of constituents [28] we define a *structure* that refers to a grammar construction to be learnt. The structure includes *containers* that in turn include *groups* and *elements*. Containers can be separate but related sentences:

{English} *Sorry, I'm late. Have you been waiting long?* (Using Present perfect in the second sentence (second container) is conditioned by the situation introduced in the first one (first container).

Containers can be the parts of a complex sentence:

{English} *Unless you work harder, you aren't going to pass the exam.*

Elements may be subject of modifications (verb forms, genre conditioned terminations, etc.). There may be dependencies between structural elements (in Fig. 3 we used the visual formalism similar to higraph blobs to illustrate this

fact), like in the following examples in French:

{French} *Les belles perles noires, elle* ***les*** *lui ai montré****es****.*

Dependency note and *explanation note* are necessary elements of the concept: hints, annotations, explanations contribute significantly even to learn language vocabulary [29].

Connectors (punctuation or word connectors) are used to connect containers. Sometimes a container (as well as an element) fulfils a function (for example, an inversion signal for the following container). *Groups* are necessary to include phrasal components like noun phrases, verb phrases, adjective phrases, and so on. Based on the metamodel, a teacher is encouraged to construct language-related model by introducing language-specific entities similar to the general schema shown in Fig. 4.

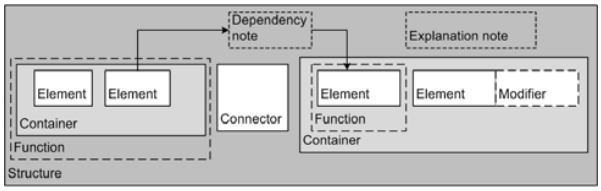

Fig. 4 Constructing grammar-driven examples

Fig. 5 shows an example of a model to manage German language grammar constructions for complex sentences with a subordinate clause.

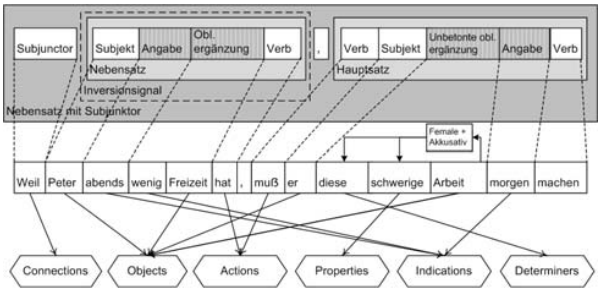

Fig. 5 German language construction example

Here we do not show all possible language-specific model entities, but only some selection, which is enough to illustrate the idea. Note that in this example the subordinate clause ("Nebensatz") fulfils a function of inversion signal for the main sentence ("Hauptsatz"). The sample sentence serves as an implementation of the models with using words from the dictionary.

Teacher's side view enables manipulation with language-specific model entitles, for example:

- **For English Language:** Containers: clause, conditional clause, relative clause …; Groups: noun phrase (NP), verb phrase (VP), determiner phrase (DP), propositional phrase (PP), adjective phrase (AdjP), adverbial phrase (AdvP), infinitive phrase (InfP); Elements: noun, verb, pronoun, adverb, preposition; Connectors: interjection, conjunction, question words…;
- **For German language:** Containers: Hauptsatz, Nebensatz, Relativesatz, Infinitivsatz,…; Groups: Angabe, Obligatorische ergãnzung, Unbetonte obl. ergãnzung, Präpositionalergãnzung; Elements: Subjekt, Verb, Pronomen, Reflexivepronomen, …; Connectors: Fragewort, Konjunktor, Subjunktor, …;
- **For Japanese language:** Elements: Noun, Verb, i-adjective, na-adjective, phrasal elements (see Fig. 6); Modifiers: verb te-form, verb na-form, verb nai-form, i-adjective negation, na-adjective negation, … .

At the initial stages of language learning it is difficult for learners to operate with all lexical categories. So we use simplified classification of dictionary words as they exposed to a learner. Such restricted classification makes easier applying the same approach to the languages with different grammar and lexical structure. This list currently includes objects, actions, determiners, properties, indications and connections, and will be extended in the future. However, these categories are sufficient for arranging experiments with relatively complex grammar structures. Fig. 6 illustrates this idea with an example of creating basic Japanese structures.

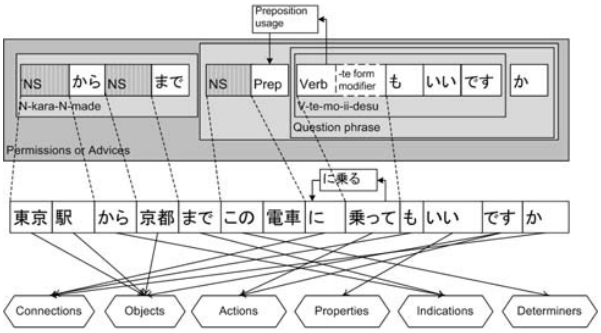

Fig. 6 Japanese language construction example

Let us propose the hypothesis that *a more detailed categorization is required only for advanced learners, who need different functionality from virtual language labs.*

B. Learner's Perspective: Model, Implementation and Experiments

Teachers create patterns. Learners use patterns explicitly or implicitly. As far as in 1965 McConlogue and Simmons reported the construction of a pattern-based English syntax parser that was able to show 77% accuracy after experience with 300 sentences [30]. Learners are expected to follow the similar process, i.e. to learn how to recognize grammar structures after being exposed to example phrases and patterns.

Basic learning scenarios can be supported by an explicitly exposed pattern or by a hidden pattern. In the first case, a learner should put required words into the correct positions by using correct modifiers to follow the exposed pattern. In the second case, a learner has no hints about the sentence structure and is encouraged to guess what the phrase could be.

Note that in many situations a user may construct a sentence that differs from the teacher's one but fits the grammar

structure (see Fig. 7). The bag of words may contain words defined as a part of an exercise pattern (that still leaves spaces for some creativity). However, it can be also an output of some special component that automatically selects words from the dictionary. If there is an available implementation of the language ontology (e.g. WordNet), this approach can be used in combination with other tools to improve words classification and to automate bag of words generation for certain exercises. A similar approach is used in the implementation of the WordNet-based web search assistant [31].

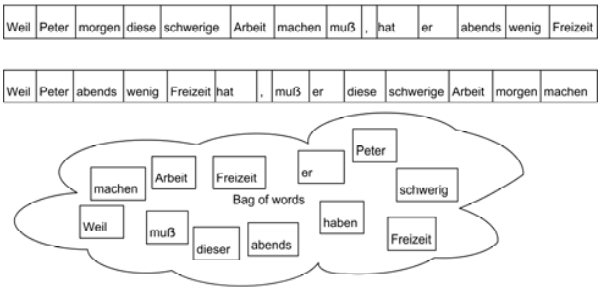

Fig. 7 Different sentences with the same structure

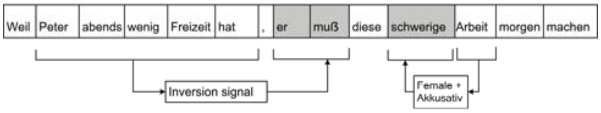

Fig. 8 Learner's mistakes

We introduced such elements as dependency notes which are used to explain how one part of the grammar construction depends on another part. They help to provide hints on possible mistakes unrelated to the sentence structure as Fig. 8 shows. While the outlined above experiments and exercises do not require traditional grammar checking technologies, they can be used in combination. Representing sentences with known grammar formalisms may be useful to analyze mistakes unrelated to teaching patterns, or to automate hints of using alternative words or phrases.

C. The Case of Programming Languages

While the proposed structure and organization are aimed mostly at natural language learning, similar techniques can be used to support programming languages learning. Indeed, programming language structures are more formal and restricted, but as mentioned in [32], modern programming languages are as rich and expressive as natural languages, and can be modeled using models similar to those used in natural language processing.

IV. Prototyping

We created a prototype application that implements some of the models introduced in the previous sections. Our prototype is focused on such aspect of learning process as teacher/learner interaction. Existing learning environments often ignore that teachers and learners act on different levels of language structures. Thus, we extend a virtual language lab concept in order to provide different usage modes for a language expert, a teacher, and a learner. The shapes of the grammar elements are created and configured by an expert. Metamodel constructions are created and managed by an expert. Patterns to learn are created by a teacher and then used by a learner. Fig. 9 illustrates this idea.

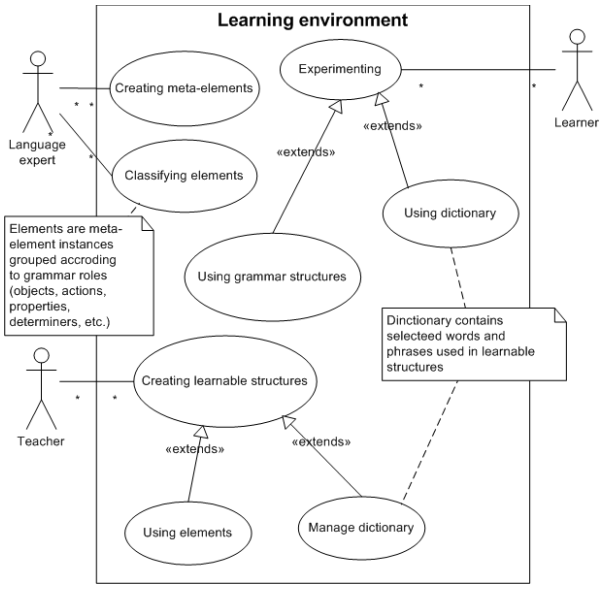

Fig. 9 Learning environment use cases

Prototype components are developed in correspondence with three levels of user interactions. **Meta-element editor** is developed to support creating new meta-elements (like objects, actions, determiners, etc.) to be used for grammar structure constructions. **Grammar rule editor** is used to construct rules and rule chains in the form of visual grammar structures connecting meta-elements and dictionary elements. **Learner panel** allows learners to construct sentences with grammar and dictionary elements, following grammar rules defined by a teacher. Learners deal with concrete implementations of meta-elements in form of words classified with respect to their grammar role.

Let us note that the current prototype puts the problems of representing and analysis of language grammars on the back burner to the benefits of supporting teacher – software – learner interaction based on visual editors of language constructions.

A. Modeling Meta-Element Shapes

In order to support constructing visual representations of the language grammar rules, basic elements have to be defined. Meta-elements are entities defined by an expert who decides which properties are used to differentiate them from each other. Currently, such properties include: entity name (object, action, subject, property, etc.), its geometry and color, list of corresponding word types, connector types, and user-defined properties. Meta-elements created at this stage are used by a

teacher for constructing visual grammar rules. Elements that implement meta-elements are intended to be used by a student while experimenting with the language constructions. Fig. 10 shows an example of configuring shapes, connectors and grammar properties of meta-elements.

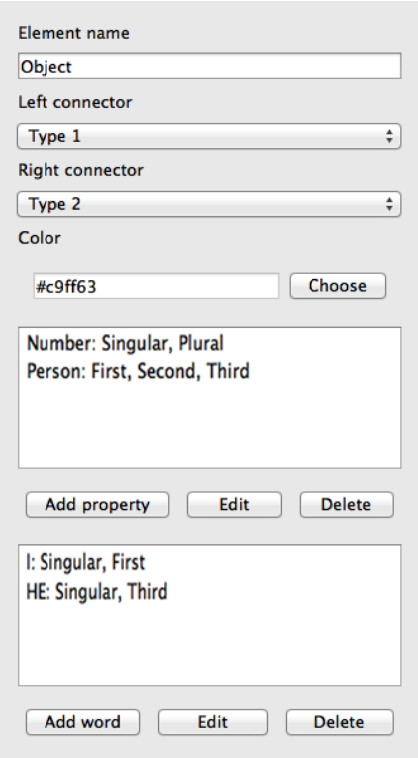

Fig. 10 Examples of meta-element views and properties

B. Constructing Grammar Rules

A visual representation of the grammar structure is constructed by establishing associative connections between the basic shapes as shown in Fig. 11.

Fig. 11 Constructing visual grammar rules by connecting meta-elements

Each visual grammar rule is considered to be a connected sequence of meta-elements (generally, a semantic network), where all connections have their own attributes, such as name, description, and linked meta-element properties (responsible for the genre, person, and tense concordance). Associations limit correct combinations of elements. Every association does not only define a possibility to connect elements, it also may contain information used during learning stage to explain the learners whether their manipulations are admissible or not. Connection properties allow managing such properties as genre, number or person, tense or case concordance.

C. Playing with the Language

Currently we support learner scenarios of two types: 1) a user follows exercises prepared by a teacher; 2) a user plays with the language freely. The freedom of experiments is limited by the shapes and grammar rules defined by a teacher. The idea is similar to WordBricks: the learners either should not be able to create illegal constructions, or they should be notified why a certain construction is invalid. Fig. 12 illustrates this idea with a simple example.

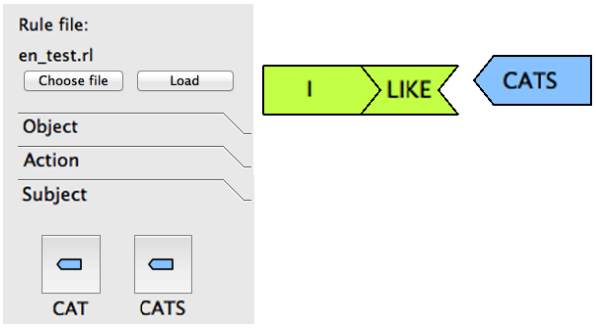

Fig. 12 Learner's experiments

Note that learners do not work explicitly with element associations. Associations are used at the internal level in order to check whether the words or phrases may be connected and interpreted correctly according to the existing grammar patterns.

V. Conclusion

Language learning techniques form a symbiotic relationship with the current computer technology. Today we see how the term "language lab" is applied not only to physical language classroom equipment, but also to a learning concept. Advancing the idea of a language virtual lab refines the current understanding of CALL: it should not be limited to a mere digitalization of learning process by transferring traditional techniques of storing and managing language-related data to the computer. It should create new use cases that are hard to implement without computer technologies.

A concept of virtual language lab can be considered as a learning model that advances the traditional architecture of an intelligent tutoring system, having four models: a domain model, a learner model, a tutor model, and a user interface model [24]. In our research we have highlighted the aspects of teacher/learner interactions and have drawn models supporting interactive incremental learning process. Naturally, the biggest challenge in the design of visual constructors similar to WordBricks is the trade-off between simplicity of the visual elements and the potential to create complex (ideally, any grammatically correct) constructions. Sentence structure is a well-researched topic in linguistics, and a number of theories of sentence representation were developed. However, traditional formalisms can only show the structure of existing sentences: there are no visual elements showing what kind of links can be potentially established between the given words.

Following Meyer's DIAMO classification of software engineering we are at the "describe" stage [33]. Our tools did not undergo an extensive classroom evaluation yet, so it is difficult to say whether full exposure to sentence structure and grammar attributes of the individual words actually improve learning. One may argue that the knowledge of parts of speech or word-word relationships in a sentence is not required to master foreign language, but this topic requires detailed discussion.

In 2002, Hubbard conducted a survey, revealing that even CALL experts are not convinced in the effectiveness of computer-supported language education [34]. Specialized educational software is still perceived by teachers as yielding only marginal improvements. Perhaps, before planning new contributions to CALL, we should answer the following question: what kind of systems would make the largest impact on learning comparing to the traditional classroom practices? We believe that computer systems should help to develop "the feel of language" in the learners' minds by exposing them to numerous examples and patterns, and by providing them with a possibility to create and test their own ideas of feasible language constructions. Such mix of handcrafted examples and open experimentation has a long tradition in science education, but in language learning similar experience is available only during private teacher-learner interaction. We hope that computer-supported virtual languages labs will help learners to perform at least simple language experiments at their own pace, even when the private teacher is not an option.

Acknowledgment

We thank Matvey Pyshkin for his successful experiments with the Scratch programming environment that he did when he was about 12 years old, thus proving that it is possible to manage relatively complex program flow structures without any primary knowledge about language syntactic rules.

References

[1] Efimov R., Mozgovoy M., Brine J. 2014. CALL for Open Experiments. In *Proceedings of International Conference on Computer Supported Education* (CSEDU 2014, Barcelona, Spain, April, 1–3, 2014), http://www.csedu.org/Abstracts/ 2014/CSEDU_2014_Abstracts.htm.

[2] Beatty, K. 2010. *Teaching and Researching Computer-assisted Language Learning*. 2nd ed. Pearson Education.

[3] Garrett, N. 2009. Computer-assisted language learning trends and issues revisited: integrating innovation. *The Modern Language Journal* 93 (Dec. 2009), 719–740. DOI= 10.1111/j.1540-4781.2009.00969.x.

[4] Mozgovoy, M., and Efimov, R. 2013. Wordbricks: a virtual language lab inspired by scratch environment and dependency grammars. *Human-centric Computing and Information Sciences* 3, 1 (2013), 1–9. DOI=10.1186/2192-1962-3-5.

[5] KET's Virtual Physics Labs, http://virtuallabs.ket.org/physics/overview/. Accessed: June, 30, 2014.

[6] UCLA ePhysics, http://ephysics.physics.ucla.edu/. Accessed: June, 30, 2014.

[7] Virtlab: A virtual laboratory: Teaching and learning chemistry can be fun! http://www.virtlab.com/main.aspx. Accessed: June, 29, 2014.

[8] ChemCollective: Online resources for teaching and learning chemistry, http://www.chemcollective.org. Accessed: June, 29, 2014.

[9] Chemist: Virtual chem. lab, http://thixlab.com/. Accessed: June, 29, 2014.

[10] Serious games for healthcare market, http://breakawayltd.com/serious-games/solutions/healthcare/. Accessed: June, 28, 2014.

[11] Indiana University Virtual Anatomy Lab, http://www.indiana.edu/ ~anat215/virtuallab/index.html. Accessed: June, 30, 2014.

[12] Simulink: simulation and model-based design, http://www.mathworks.com/products/simulink/. Accessed: June, 28, 2014.

[13] ContLab Automatic Control Laboratory, http://www.contlab.eu/en/. Accessed: June, 30, 2014.

[14] PIDlab PID control laboratory, http://www.pidlab.com/en/. Accessed: June, 30, 2014.

[15] Gallagher, S. A. 1997. Problem-based learning: where did it come from, what *does it do, and where is it going? Journal for the Education of the Gifted 20, 4 (Sum 1997), 332*–362.

[16] Cabezas, D., Vassiliev, A., and Pyshkin E. Assisted lung ventilation control system as a human centered application: The project and its educational impact on the course of embedded systems. In J.J. (Jong Hyuk) Park et al. (eds.), Ubiquitous Computing Application and Wireless Sensor, Lecture Notes in Electrical Engineering, 331, Springer Science+ Business Media Dordrecht, 2015.

[17] Nietzio, A., Scheer, B. and Bühler C. 2012. How long is a short sentence? – A linguistic approach to definition and validation of rules for easy-to-read material. In *K. Miesenberger et al. (Eds.): Computers Helping People with Special Needs, ICCHP 2012, Part II* (Linz, Austria, July 11–13, 2012), 369–376, LNCS 7383, Springer Berlin Heidelberg. DOI= 10.1007/978-3-642-31534-3_55.

[18] Robin, R. 2007. Commentary: learner-based listening and technological authencity. *Language Learning & Technology* 11, 1 (Feb. 2007), 109–115.

[19] Maloney, J., Resnick, M., Rusk, N., Silverman, B., and Eastmond, E. 2010. The scratch programming language and environment. *Trans. Comput. Educ.* 10, 4 (Nov. 2010), 16:1-16:15. DOI= 10.1145/1868358.1868363.

[20] Malan D.J. Scratch for budding computer scientists (Online) URL: http://cs.harvard.edu/malan/scratch/. Accessed: June, 27, 2014.

[21] Nagata N. 2009. Robo-Sensei's NLP-based error detection and feedback generation. *Calico Journal*, 26(3), 562-579.

[22] Levy, M. 1*997. Computer-assisted language learning: Context and conceptualization.* Oxford University Press, 1997.

[23] Murray J.H., Morgenstern D., Furstenberg G. 1989. The Athena Language Learning Project: design issues for the next generation of computer-based language learning tools. *Modern Technology in Foreign Language Education* (1989), 97-118.

[24] Joshi, A. and Sasikumar, M. 2009. A constructivist approach to teaching sentences in Indian language. In *International Workshop on Technology for Education* (Bangalore, Aug. 2009), 75–80.

[25] Fum, D., Pani, B. and Tasso, C. 1991. Teaching the English tense: integrating naive and formal grammars in an intelligent tutor for foreign language teaching. In *Proceedings of the Fifth Conference on European Chapter of the Association for Computational Linguistics*, (Stroudsburg, PA, USA, 1991), 149–154, EACL '91, Association for Computational Linguistics. DOI= 10.3115/977180.977206.

[26] Milne, I. and Rowe, G. 2002. Difficulties in learning and teaching programming views of students and tutors. *Education and Information Technologies* 7, 1 (Mar. 2002), 55–66, Kluwer Academic Publishers Hingham, MA, USA. DOI= 10.1023/A:1015362608943.

[27] Petrov, S., Dipanjan D. and McDonald, R. 2012. A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC '12).*

[28] Lees, R.B. 1957. Syntactic Structures. Languages, 3(3), part 1, 375-408.

[29] Chun, D. 2006. CALL technologies for L2 reading. *Calling on CALL: From Theory and Research to New Directions in Foreign Language Teaching*, Ed. by L. Ducate, N. Arnold, CALICO, 69–98 (2006).

[30] Mcconlogue, K. 1965. Analyzing English syntax with a pattern-learning parser. *Commun. ACM* 8, 11 (Nov. 1965), 687–698, ACM, NY, USA.

[31] Pyshkin, E. and Kuznetsov, A. 2010. Approaches for web search user interfaces. *Journal of Convergence* 1, 1 (2010).

[32] Hindle, A., Barr, E. T., Su, Z., Gabel, M. and Devanbu, P. 2012. On the naturalness of software. In *Proceedings of the 34th International Conference on Software Engineering*, (Piscataway, NJ, USA, 2012), 837–847, ICSE '12, IEEE Press.

[33] Meyer, B. 2009. *Touch of Class. Learning to Programm Well with Objects and Contracts.* Springer Verlag.

[34] Hubbard, P. 2002. Survey of unanswered questions in *Computer Assisted Language Learning*, Stanford University, http://www.stanford.edu/~efs/callsurvey/index.html.