On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises

Laura Zavala
Medgar Evers College | CUNY
Brooklyn, NY 11225
rzgutierrez@mec.cuny.edu

Benito Mendoza
New York City College of Technology | CUNY
Brooklyn, NY 11201
bmendoza@citytech.cuny.edu

ABSTRACT

In introductory programming courses, proficiency is typically achieved through substantial practice in the form of relatively small assignments and quizzes. Unfortunately, creating programming assignments and quizzes is both, time-consuming and error-prone. We use Automatic Item Generation (AIG) in order to address the problem of creating numerous programming exercises that can be used for assignments or quizzes in introductory programming courses. AIG is based on the use of test-item templates with embedded variables and formulas which are resolved by a computer program with actual values to generate test-items. Thus, hundreds or even thousands of test-items can be generated with a single test-item template. We present a semantic-based AIG that uses linked open data (LOD) and automatically generates contextual programming exercises. The approach was incorporated into an existing self-assessment and practice tool for students learning computer programming. The tool has been used in different introductory programming courses to generate a set of practice exercises different for each student, but with the same difficulty and quality.

ACM Reference format:
Laura Zavala and Benito Mendoza. 2018. On the Use of Semantic-Based AIG to Automatically Generate Programming Exercises. In *SIGCSE '18: 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, NY, NY, USA, 6 pages.
https://doi.org/10.1145/3159450.3159608

1 INTRODUCTION

The literature abounds with research on pedagogies and innovative approaches for introductory computer programming courses (CS1), such as collaborative learning, pair-programming, peer-lead instruction, flipped classrooms, and live coding [1-6]. Many works are motivated by the high failure rates in CS1 courses all over the world. Passing rates are estimated to be around 63%.

Regardless of the approach or the degree to which an approach is used in the classroom, the need for considerable practice in introductory programming courses is indisputable and widely acknowledged. Proficiency in these courses is usually reached through small but frequent assignments. The work in [7] validates the importance of performing multiple exercises with prompt feedback in order for students to gain proficiency on a concept. In [8], evidence is provided that both, practice and reflection, play critical roles in the development of programming proficiency.

Preparing assignments and assessments is a time-consuming task for instructors. Automatic Item Generation (AIG) was used to create a tool that automatically generates programming practice exercises thus relieving the instructor from having to generate them. AIG is an approach for developing test-items or questions for exams, automatically by a program [9]. Existing approaches to AIG are mainly template-based. Instead of creating a question, experts create a template with embedded variables and formulas. By replacing those variables and formulas with different values from a range of values specified by the expert, a high volume of test-items can be generated from a single item template. AIG is critical in applications such as Computer Adaptive Testing (CAT) where a very large bank of items is needed. CAT is a form of computer-based testing that adapts to the examinee's ability level by selecting questions based on what is known about the examinee from answers to previous questions [10]. CAT facilitates precise evaluation at the individual level, which could lead to shorter and faster tests (i.e., the test can stop as soon as an assessment of the student's knowledge has been made).

The type of practice given to students should not be overlooked. Several educational theories emphasize the need for introductory contexts that align with students' interests and goals [11, 12]. Examples in CS1 courses should make sense to students and promote engagement. Recent works [8, 12-21] have explored the use of engaging applications such as robotics, music, games, media, and physical computing in introductory programming courses. Results show that these approaches engage students positively, increase motivation, facilitate understanding, and improve outcomes and retention rates.

Instead of choosing one specific application for a CS1 course, we are concerned with creating numerous practice exercises that are meaningful to a certain degree. It is not easy to create contextual examples for minimal exercises and to manually create plenty of examples that will satisfy a broad variety of learners. We have extended the traditional template-based AIG

SIGCSE '18, February 21–24, 2018, Baltimore, MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159608

approach with a semantic-based approach that connects to existing Linked Open Data (LOD)[1] sources to generate different contexts for a practice exercise. To the best of our knowledge, there is no other known work that combines linked open data and automatic item generation to generate contextualized items. We are concerned with introductory programming courses and problem solving as application domain, but our approach is transferable to many other domains.

We have incorporated our semantic-based AIG tool as part of a web-based system that creates and delivers exams online [22][2]. We have used such system in several courses to deliver quizzes. We also used our tool to generate coding problems that we have administered on paper. This paper presents the semantic-based AIG approach used in our tool as well as an initial evaluation based on our experience thus far and the results of a pilot study. Advantages of the semantic-based AIG approach presented here include: a) having a large pool of practice exercises or test items; b) generating different questions for each student and thus making it harder for students to cheat; c) increasing motivation and reducing chances of misunderstanding the question; and d) providing students with plenty of exercises to practice until proficiency is achieved. Although our approach currently has mild contextualization, it can form the basis for a more advanced learning platform with more specialized contextual and CAT and/or intelligent tutoring features.

2 RELATED WORK

Some works have explored the use of AIG in the computer-programming domain. In [23] AIG is used to automatically generate questions in the mathematics, physics and computer programming domains. The late one however, the only variability is in the programming language asked to solve the problem (which assumes students can write in different programming languages). The authors point out that the main point of interest of these exercises is in its automatic grading (through test cases).

In [24-27] the authors present an ontology-based, multiple choice question generation approach. They use ontologies along with some natural language processing to generate factual questions about the domain of the ontology. For example, a geographic ontology is used in [25] to generate questions about geography. No AIG style templates are used. In contrast, we use Linked Open Data and its associated ontologies to insert context into the questions we generate using AIG templates.

The authors in [28] perform an assessment of the usability of Linked Data for the generation of item variables in AIG. Their focus is on the use of LOD as the domain knowledge from which questions can be generated. They raise the issue of data quality and inconsistencies in LOD which can be a problem when LOD is using as source of knowledge. In contrast, we use LOD to contextualize the test items, which do not belong to the same domain as the ontology (the domain is *computer programming*). For example, using a *movies* ontology, [28] would generate quizzes about movies while we instead generate computer programming questions in the context of movies (using movies as part of the problem formulation).

3 SEMANTIC-BASED AIG

3.1 Automated Item Generation

AIG is an approach for developing test-items or questions for exams, automatically by a program. The most common AIG approach is based on the use of test-item templates with embedded variables and formulas. The variables and formulas in the template are resolved by a computer program with actual values to generate test-items. Current approaches to AIG vary by the method used for giving values to the variables: a text [29], mathematical equations [9, 30], or a semantic model [24-28]. The obvious advantage of an AIG system is its ability to produce high volumes of test-items and therefore numerous different tests with the same difficulty and quality.

We use AIG in the computer-programming domain to generate questions of different types, such as open-ended, short answer, multiple choice, and true or false. Figure 1 shows an example of a short answer test-item template and two sample test-items generated from the template. The variables are marked with {{}}. A test-item template is composed of:

- Stem – The stem contains the actual question with embedded variables.
- Options (optional)– For multiple-choice test-items templates, the answer options with embedded variables. The answer options include one correct option and one or more incorrect options or distractors.
- Key – The key is an indicator to the correct answer.
- Script – The script is a computer program that generates values for the embedded variables in the previous elements of the template and generates the key.

3.2 Linked Open Data

Linked Data is a method of publishing data using recognized standards so that it can be interlinked and become more useful through semantic queries. It uses standards and technologies that allow sharing of information in a way that can be read automatically by computers. This enables data from different sources to be connected and queried [31]. Linked Open Data (LOD) is Linked Data that is released under an open license, which does not impede its reuse for free [32].

In LOD, relationships are represented as (*subject, predicate, object*) triples. Resources are represented with URIs (Uniform Resource Identifiers), which can be abbreviated as prefixed names. The *predicate* specifies how the *subject* and *object* are related. A comprehensive introduction to linked open data is out of the scope of this paper. The interested reader can check [31] and [32].

[1] https://www.w3.org/standards/semanticweb/data
[2] http://mz-unbound.com/newipractice/

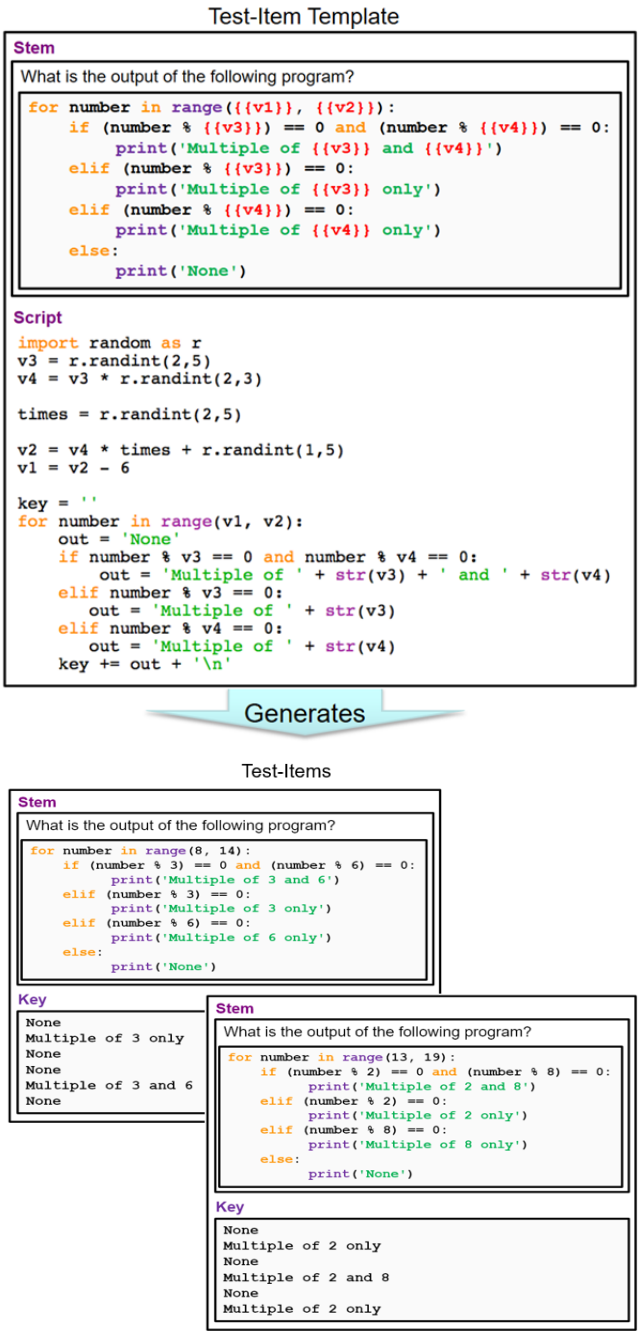

Figure 1: Example of AIG applied to the computer programming domain: a test-item template, the algorithm to instantiate it, and two of the hundreds of questions that can be generated with the template.

The DBpedia[3] linked open dataset consists of RDF triples extracted from the *infoboxes* commonly seen on the right-hand side of Wikipedia articles. The following is an excerpt from DBpedia about *The Hunger Games* movie:

PREFIX db: <http://dbpedia.org/resource/>

PREFIX dbo: <http://dbpedia.org/ontology/>

PREFIX dbp: <http://dbpedia.org/property/>

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

db:The_Hunger_Games_(film) rdf:type dbo:Film .

db:The_Hunger_Games_(film) dbo:writer db:Suzanne_Collins.

db:The_Hunger_Games_(film) dbo:starring db:Elizabeth_Banks.

db:The_Hunger_Games_(film) dbo:starring db:Jennifer_Lawrence.

db:The_Hunger_Games_(film) dbo:starring db:Liam_Hemsworth.

db:The_Hunger_Games_(film) dbo:editing db:Juliette_Welfling.

db:The_Hunger_Games_(film) dbo:director db:Gary_Ross.

db:The_Hunger_Games_(film) dbo:producer db:Nina_Jacobson.

db:The_Hunger_Games_(film) dbp:country "United States".

Optionally, resources in LOD can be associated with an ontology that specifies the concepts (classes) that a resource can belong to, as well as the types of relations among classes. In the example above, The *Hunger Games* film is specified to belong to the *Film* class. Further, the DBpedia ontology states that a resource of the class *Film* (e.g. *The Hunger Games*) has a *writer* relation to a resource of the class *Person* (e.g. Suzanne Collins). Resources in different LOD datasets can also be interlinked to complement the knowledge about the resources described in the data. For example, the country associated with the *Hunger Games* movie in the previous example is United States:

db:The_Hunger_Games_(film) dbp:country "United States"

Geonames[4], an LOD dataset with knowledge about millions of geographical locations worldwide, could be queried to retrieve further information about the resource *United States*.

LOD is used in our semantic-based AIG approach to populate the variables of the test-item templates in order to generate programming exercises that are associated with real-world concepts and examples. LOD datasets can be queried using the SPARQL query language to query local or remote repositories (e.g. http://dbpedia.org/sparql/). Figure 2 shows an SPARQL query that can be used to get a list of actors from the DBpedia dataset.

We specify ontological elements that can be used to populate the variables in the test-item templates. We use SPARQL to query linked open datasets in order to obtain instances of the ontological elements and use them as values for the variables of a test-item template.

[3] http://wiki.dbpedia.org/

[4] http://www.geonames.org/ontology

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?subject WHERE
{
   ?subject rdf:type <http://dbpedia.org/ontology/Actor> .
}
```

Figure 2: An SPARQL query to obtain a list of actors from DBpedia.

Figure 3 provides an example of a short answer, semantic-based test-item template and two sample questions generated from the template. For simplicity purposes, we provide only the algorithm that generates the values for the variables.

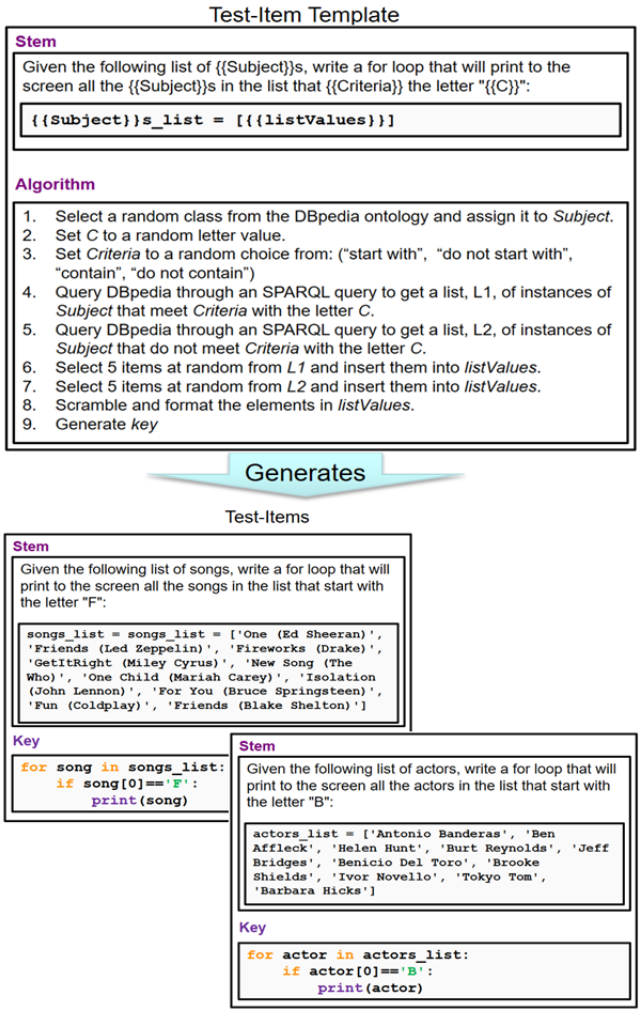

Figure 3: Example of semantic-based AIG applied to the computer programming domain: a test-item template, the algorithm to instantiate it, and two of the hundreds of questions that can be generated with the template.

A small number of instances in DBPedia are mapped to the ontology. For example, few actors are declared to be instances of the class *Actor*. However, they are a DBPedia resource and are related to other resources such as films. For example, the *starring* relation associates a film with an actor (even if the actor is not declared as such in the ontology; he might be declared only as a *Person*). Therefore, it can be inferred that the object of the *starring* relation is an *Actor*. Due to this limitation, our approach cannot rely on generic queries like the one in Figure 2. Currently, we have a fixed set of classes/concepts that can be used to instantiate the variables in a template and we have pre-built SPARQL queries for them. The SPARQL queries used in the actors' example of Figure 3 are:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?actor  WHERE
{
  ?movie <http://dbpedia.org/ontology/starring> ?actor.
  FILTER (strStarts(str(?actor), "http://dbpedia.org/resource/B")).
}
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?actor  WHERE
{
   ?movie <http://dbpedia.org/ontology/starring> ?actor.
   FILTER (regex(?actor, "^((?!B).)*$")).
}
```

Queries to LOD can be more complex and detailed than the queries in these examples, allowing for more specific contexts to be defined. For example, in the movies domain, one could query specifically French films, films in a specific genre, or films associated with a particular actor or director.

The content of the LOD cloud is diverse. It comprises data about geographic locations, people, companies, books, music, scientific publications, films, television and radio programs, genes, proteins, online communities, census results, and product reviews. As May 2009, the Web of Data consisted of 4.7 billion triples, which are interlinked by around 142 million RDF links [30]. Currently, we have focused our work on a few LOD datasets (i.e. DBPedia, foaf, and geonames), which are the biggest datasets in the LOD cloud. And as it was mentioned before, we only use a fixed set of concepts from them. Full use of a variety of LOD sources will allow to provide further contextualization.

4 EVALUATION

We have used our semantic-based AIG tool in several courses to generate quizzes with positive outcomes for instructors and students. In this paper, we focus on a pilot study designed to conduct an initial assessment of the impact of explicitly spending extra time on practice exercises generated with our sematic-based AIG tool.

Two sections of an introductory programming course were used for the study. The course is taught at a four-year urban college and uses Python programming language. One section of the course was used to test an intervention strategy implemented with the goal of honing students' programming skills. The other section was used as a control group. Students in both groups were given a test to evaluate their coding skills in the topics learned up to that point. The test was given past mid-

semester, when students had already learned some basic programming.

Our semantic-based AIG approach was used to generate a set of practice exercises (four sets of twelve problems) individualized to each student's needs in the intervention group. Students worked on their exercises for a period of two weeks. Some exercises were given to students as in-class lab assignments and others as homework assignments. Nothing else was covered during that period of time in the intervention group while the control group continued with the schedule as planned. A post-assessment was given to students in both groups near the end of the semester to evaluate them on the same skills than in the pre-test. We were interested in observing whether there was any significant difference in the progress made in learning-to-code between the students from the two groups.

Figures 4 to 7 show the results obtained in the pre and post assessments from the control group and the intervention group, respectively. The stacked line charts depict the cumulative students' scores by type of question: 1) code reading (tracing code and indicating the output of a program), 2) code manipulation (modify/extend provided starter code to meet requirements), and 3) code writing (write code from scratch to meet requirements). Students have been anonymized (s1, s2, etc.). The evaluation instrument (quiz) for each skill level had a max score of 100.

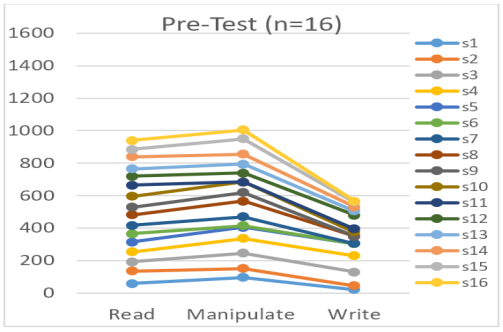

Figure 4: Pre-test results: Control group cumulative scores by level.

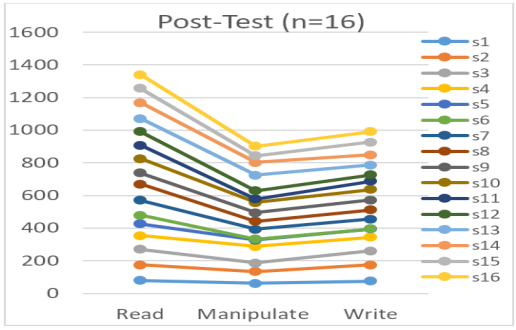

Figure 5: Post-test results: Control group cumulative scores by level.

Table 1 shows the average grade of the students on the exams pre- and post-intervention. Both groups exhibited an improvement towards the end of the course. However, students in the intervention group showed a greater improvement than students who did not. All the students in the intervention group finished at a proficient level in code reading questions, while some in the control group did not. A larger improvement was observed in the code writing skills of the students in the intervention group.

Table 1: Average grade obtained by both groups of students, pre- and post-intervention.

	Pre-Test			Post-Test		
Group	**Read**	**Manip.**	**Write**	**Read**	**Manip.**	**Write**
Control	**58.8**	**62.8**	**35.3**	**83.8**	**56.4**	**61.9**
Intervention	**76.3**	**49.4**	**43.8**	**95.0**	**92.3**	**81.5**

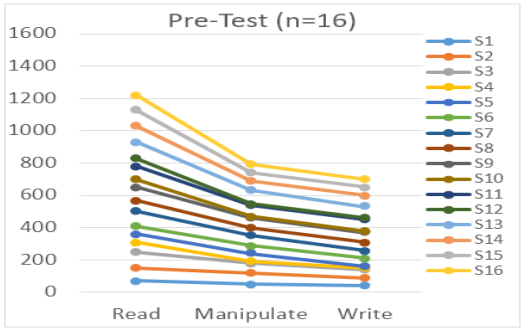

Figure 6: Pre-test results: Intervention group cumulative scores by level.

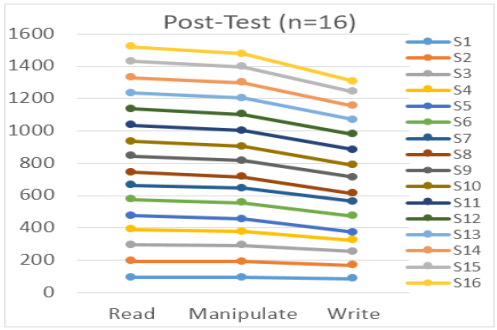

Figure 7: Post-test results: Intervention group cumulative scores by level.

6 CONCLUSIONS AND FUTURE WORK

We presented a semantic-based AIG approach to the automatic generation of contextualized programming exercises that can be used for quizzes and homework assignments in introductory programming courses. The approach extends the traditional template-based AIG by connecting to existing Linked Open Data sources to generate different contexts for a programming practice exercise template.

A pilot study assessed the impact of explicitly spending extra time on practice exercises generated with our sematic-based AIG

tool. Results obtained from both, an intervention group and a control group, show the benefits of dedicating extra time to practicing. Students that participated in the intervention strategy show a greater improvement in learning-to-code over the students in the control group.

In its current state, our semantic-based AIG approach generates examples with simple contextualization. We will continue our work to devise how LOD and ontologies can be exploited to generate richer and more detailed contexts. We are also interested in evaluating whether it really matters that the students be familiar with the context of the question. Once further contextualization is achieved and preferences of students can be inferred, research must be done to evaluate the impact of this approach on learners' cognition and motivation. In the long term, we are interested in building a suite of CAT and intelligent tutor tools that will offer students the opportunity for plenty of personalized practice that will potentially help them achieve proficiency.

6 ACKNOWLEDGMENTS

We gratefully acknowledge the support of the Research Foundation of CUNY under PSC-CUNY Award 68164-00 46.

REFERENCES

[1] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagen, Y. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. 2001. A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First- year CS Students. *SIGCSE Bulletin*, 33, no. 4, 125-140.

[2] L. Zavala. 2016. Read, Manipulate, and Write: A study of the role of these cumulative skills in learning computer programming. *Proceedings of the ASEE NE 2016 Conference.* April 28-30, University of Rhode Island, Kingston, RI.

[3] L. Zavala and B. Mendoza. 2017. Precursor skills to writing code. *Journal of Computing Science in Colleges*, 32, 3, 149-156.

[4] J. Sweller, and G. A. Cooper. 1985. The use of worked examples as a substitute for problem-solving in learning algebra. *Cognition and Instruction*, 2, no. 1(1985), 59–89.

[5] M. Guzdial and J. Robertson. 2010. Too much programming too soon? *Communications of the ACM*, 53, no. 3 10-11, ACM, March.

[6] M. Guzdial. 2015. What's the best way to teach computer science to beginners? *Communications of the ACM*, 58, no. 2,12-13, ACM, January.

[7] D. Deb, M. M. Fuad, and M. Kanan. 2017. Creating Engaging Exercises with Mobile Response System (MRS). *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education.* March 08-11, Seattle, Washington.

[8] M. J. Scott, S. Counsell, S. Lauria, S. Swift, A. Tucker, M. Shepperd and G. Ghinea. 2015. Enhancing Practice and Achievement in Introductory Programming with a Robot Olympics. *IEEE Transactions on Education*, vol. 58, no. 4, 249-254.

[9] M. J. Gierl and H. Lai. 2012. The Role of Item Models in Automatic Item Generation. *International Journal of Testing*, 12(3), 273-298.

[10] H. Wainer, N. J. Dorans, D. Eignor, R. Flaugher, B. F. Green, R. J. Mislevy, L. Steinberg, and D. Thissen. 2000. *Computerized adaptive testing: A primer, Second Edition.* Hillsdale, NJ: Lawrence Erlbaum Associates.

[11] J. R. Savery and T. M. Duffy. 1995. Problem based learning: An instructional model and its constructivist framework. *Educational technology*, 35, no. 5, 31-38.

[12] W. W. Cobern. 1993. Contextual constructivism: The impact of culture on the learning and teaching of science. InK. G.Tobin (Ed.),*The practice of constructivism in science education*(pp. 51–69). Hillsdale, NJ: Lawrence Erlbaum Associates.

[13] T. Im, S. Siva, J. Freeman, B. Magerko, G. Hendler, S. Engelman, M. Miller, B. Villa, and T. McKlin. 2017 Incorporating music into an introductory college level programming course for non-majors. *Proceedings of the IEEE Integrated STEM Education Conference (ISEC)*, Princeton, NJ, 2017, pp. 43-48.

[14] J. Stigall and S. Sharma. 2017. Virtual reality instructional modules for introductory programming courses. *Proceedings of the 2017 IEEE Integrated STEM Education Conference (ISEC)*, Princeton, NJ, pp. 34-42.

[15] M. A. Rubio, R. Romero-Zaliz, C. Mañoso and A. P. de Madrid. 2014. Enhancing an introductory programming course with physical computing modules. *Proceedings of the 2014 IEEE Frontiers in Education Conference (FIE)*, Madrid, 2014, pp. 1-8.

[16] F. I. Anfurrutia, A. Álvarez, M. Larrañaga and J. M. López-Gil. 2016. Incorporating educational robots and visual programming environments in introductory programming courses. *Proceedings of the 2016 International Symposium on Computers in Education (SIIE)*, Salamanca, pp. 1-4.

[17] E. Weilemann, P. Brune, and D. Meyer. 2016. Geek Toys for Non-techies? Using Robots in Introductory Programming Courses for Computer Science Non-majors. *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, Koloa, HI, pp. 31-40.

[18] W. S. Yue and W. L. Wan. 2015. The effectiveness of digital game for introductory programming concepts. *Proceedings of the 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, London, pp. 421-425.

[19] M. Paralič and E. Pietriková. 2014. Learning by game creation in introductory programming course: 5-Year-long study. *Proceedings of the 12th IEEE International Conference on Emerging eLearning Technologies and Applications (ICETA)*, Stary Smokovec, pp. 391-396.

[20] R. Rahul, A. Whitchurch and M. Rao. 2014. An open source graphical robot programming environment in introductory programming curriculum for undergraduates. *Proceedings of the International Conference on MOOC, Innovation and Technology in Education (MITE)*, Patiala, pp. 96-100.

[21] N. Adamo-Villani, T. Haley-Hermiz and R. Cutler. 2013. Using a Serious Game Approach to Teach 'Operator Precedence' to Introductory Programming Students. *Proceedings of the 17th International Conference on Information Visualization*, London, pp. 523-526.

[22] B. Mendoza, J. Reyes-Alamo, H. Wu, A. Carranza, and L. Zavala. 2016. iPractice: A Self-assessment Tool for Students Learning Computer Programming in an Urban Campus. *Journal of Computing Sciences in Colleges*, 31, 3, 93-100..

[23] F. Prados, I. Boada, J. Soler, and J. Poch. 2005. Automatic generation and correction of technical exercices. *Proceedings of the International Conference on Engineering and Computer Education (ICECE 2005)*, Madrid, Spain.

[24] Vinu E.V and P Sreenivasa Kumar. 2015. Improving Large-Scale Assessment Tests by Ontology Based Approach. *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference*. May 18-20, Hollywood, Florida, USA.

[25] A. Papasalouros, K. Kotis and K. Kanaris. 2008. Automatic generation of multiple-choice questions from domain ontologies. *Proceedings of the IADIS e-Learning 2008 conference*, Amsterdam, Netherlands.

[26] Al-Yahya M. 2014. Ontology-Based Multiple Choice Question Generation. *The Scientific World Journal.* Vol. 2014, Article ID 274949, 9 pages, 2014. doi:10.1155/2014/274949.

[27] M. Cubric and M. Tosic. 2010 Towards automatic generation of eAssessment using semantic web technologies. *In Proceedings of the 2010 International Computer Assisted Assessment Conference*, Jul 2010. University of Southampton.

[28] Foulonneau, M. 2012. Generating Educational Assessment Items from Linked Open Data: the Case of DBpedia. *In: Garcia-Castro, R.e.a. (ed.): Extended Semantic Web Conference Workshops (at the 8th International conference on The Semantic Web)*, Vol. LNCS 7117. Springer, Heraklion, Greece, pp 16-27.

[29] N. Karamanis, L. A. Ha, and R. Mitkov. 2006. Generating multiple-choice test items from medical text: A pilot study. *Paper presented at the Fourth International Conference Natural Language Generation.* Sydney, Australia.

[30] B. Liu, H. Chen, and W. He. 2008. A framework of deriving adaptive feedback from educational ontologies. *Proceedings of the 9th International Conference for Young Computer Scientists*, Zhang Jia Jie Hunan, China.

[31] T. Berners-Lee. 2006. Linked Data. *Design Issues.* W3C. Retrieved 2017-08-18.

[32] C. Bizer, T. Heath, T. Berners-Lee. 2009. Linked Data—The Story So Far. *International Journal on Semantic Web and Information Systems.* 5 (3): 1–22. ISSN 1552-6283.