Full Stack Web Development Teaching: Current Status and a New Proposal

Anna Petrikoglou and Theodore H. Kaskalis
Department of Applied Informatics, University of Macedonia, 156 Egnatia Str., 54636, Thessaloniki, Greece

Keywords: Code Playground, Full Stack Development, Computing Learning System, Web Technologies, Code Testing.

Abstract: The main purpose of this effort is to present a brand-new environment for practising some of the most broadly used – both client- and server- side – web technologies. It is about a web-based, access-free, educational platform, which provides a user-friendly interface, illustrative graphics and supporting material, as well. Full stack development platforms are rarely met online, as most of them are usually oriented towards either front- or back- end development and focus on specific programming languages without offering an overview of actual, integrated projects. This research also involves evaluating existing teaching methods, scanning and comparing some of the most popular, educational web platforms and, furthermore, discovering simple techniques and efficient approaches to reach valuable programming resources for both students and self-learners. The paper places particular emphasis on the recognition of the applications' key features and the variety of programming tools that promote learning and skill enhancement. Moreover, it discusses the roles of tutors and learners, while suggesting a learning path for novice developers. Given the fact that computer science courses often require exceptional practices, this study aims at encouraging active, self-motivated and self-paced learning.

1 INTRODUCTION

In the past several decades, multiple dilemmas and educational difficulties have emerged while endeavouring to familiarise, but also to teach computer science (Gomes and Mendes, 2007; Robins, Rountree and Rountree, 2003; Xinogalos and Kaskalis, 2012).

It is widely recognized that there is an expanding scope of web technologies, as human needs are countless and pioneering ideas are interminably cultivated (Kelly, 2016; Silberglitt, Antón, Howell and Wong, 2006). The availability of valuable programming tools may speed up production processes, improve their efficiency, and broaden the prospects of each venture (Andreessen, 2011; Baker, 2017). The fact that modern technology companies have to deal with a significant lack of skilled, well-trained and experienced staff poses a burning issue and creates favourable conditions for the preparation of the new generation's technologists and developers.

Given the above, engaging in programming involves acquiring useful skills and a wide range of career choices, but also ignites potential in the most promising fields of the modern economies (Maiorana, 2014). Furthermore, it reportedly upgrades cognition abilities (Eisenberg, His and Oh, 2017; Papert, 1980). At the same time, academics confront a constant necessity to accelerate all requisite, yet crucial, educational reforms, while tutoring prospective industry professionals (Bergersen, 2015; Bers, 2018; Wilson, 2002; Decker, 2007). But, how do they deal with the numerous challenges they face (Gomes and Mendes, 2007; Jenkins, 2001; Robins, Rountree and Rountree, 2003; Xinogalos and Kaskalis, 2012)?

Prior to our teaching system's implementation, we found it mandatory to take into consideration the most frequently encountered and much-talked-about difficulties in teaching programming. In order to launch an all-inclusive and functional web development platform, we also needed to explore the pre-existing tools and the already stated solutions, and identify the most desirable features. A literature review in combination with a research on the already offered possibilities has led to the ultimate development of a teaching system, that is expected to replace others, as it integrates heterogeneous technologies and functionalities.

Petrikoglou, A. and H. Kaskalis, T.
Full Stack Web Development Teaching: Current Status and a New Proposal.
DOI: 10.5220/0008066202180225
In *Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST 2019)*, pages 218-225
ISBN: 978-989-758-386-5

2 CHALLENGES AND EDUCATIONAL APPROACHES

Research has shown that some of the difficulties mentioned above are related to the learners' background, skills, methodology and way of thinking, as well (Wilson and Shrock, 2001). In addition, their perception, experience and discipline may vary. That makes tutors' work more difficult (Bennedsen, 2008; Gomes and Mendes, 2007). The development of a consolidated educational model becomes extremely challenging, and an apparent question arises as to the adjustment and the personalization of the teaching material.

In many cases, tutors are also responsible for several difficulties. Their approach, the teaching methods they use, their experience, the content of the curriculum, the technologies and the tools in use, as well as the way they deal with possible learners' misunderstandings or stagnancies, are also considered as crucial factors.

The bibliography, referring to CS courses, common difficulties encountered while teaching, tutors' dilemmas, proposals and most effective practices, is extensive (Chao, Parker and Davey, 2013; Ertmer, Gopalakrishnan and Ross, 2001; Gomes and Mendes, 2007; Harrop, 2018; Jackson and Miller 2009; Liu and Phelps, 2011; Péter and László, 2003; Xinogalos and Kaskalis, 2012). Reflections, speculations, as well as ways of solving problems that have emerged from time to time, have been presented in scientific articles, analysed in conferences and discussed in educational communities, aiming at upgrading educational processes and improving learners' performance.

Admittedly, these obstacles are often due to the combination of the aforementioned factors and, of course, to the general assumption that this is about a demanding subject both for learners and for tutors. Programming skills require basic knowledge of computer science, programming languages and tools, critical and methodical thinking and theoretical background, as well (Bennedsen, 2008; Robins, Rountree and Rountree, 2003). Teaching material should be regularly renewed, updated to the latest data and emerging technologies and adjusted to learners' needs (Atkin, Black, James, Olson, Raizen, Sáez and Simons, 1996).

Educators are challenged to reconstitute teaching techniques and incorporate innovative methods that will attract and retain learners' interest (Lathrop, 2010; Prensky, 2007). Traditional knowledge transmission (through lectures and notes) is not enough. Effective teaching implies close and targeted guidance, support, and inherent interest in learners' consistent progress and high performance (Jenkins, 2001). Concurrently, providing comprehensive knowledge from diverse and reputable resources, in order to meet educational needs is also ancillary and enriches the available teaching material.

The use of the internet for educational purposes marked a new era for the educational community. In particular, this type of education provides demonstrated flexibility, which makes it especially attractive to learners. Tutors and learners are able to work remotely, in joint projects, - simultaneously or at any preferred time - share educational material, examine the correctness of the code, experiment with different technologies, and communicate online (Means, Toyama, Murphy, Bakia, and Jones, 2010). E-learning also eliminates temporal, geographical and financial constraints. E-courses can be conducted asynchronously, without distance being an inhibiting factor. The corresponding services provided free of charge account for a large percentage of the total online offer, thus allowing the ones interested to enjoy a wide range of alternatives for their education. Although their contribution may be complementary, in some cases they eventually replace traditional courses.

Our research focuses on the online mediums that encourage learning and support addressing common difficulties confronted by tutors and learners (including both students and self-learners). Inspiring ideas and encouraging new ventures (Gottfried, 1997; Robins, Rountree and Rountree, 2003), motivating novice developers and urging active participation in code deployment and program implementation are at the core of this paper's purpose. Tutors' role is auxiliary, as they only contribute with further clarifications, constructive feedback and motivation, when necessary. In the latter case, a learner intends to preserve a research attitude towards the subject, manage the upcoming challenges and achieve results by dividing a perplexing problem into individual arguments.

The presented work promotes an active approach and advocates the benefits of searching for information. Active learning has been observed to be beneficial to novice developers' training, curriculum comprehension and adaptation to new technologies. Learners take greater responsibility for the type, quality and quantity of education they receive. Consequently, they determine their path and resources, as they research at will. By understanding principal rules, proper syntax and correlated concepts, they gradually familiarize with the task of breaking a problem into small parts. In this way, they

are methodically guided to solving it (Raj, Patel, and Halverson, 2018; Mironova, Amitan, Vilipõld and Saar, 2016).

3 PLATFORM COMPARISON AND REVIEW

In recent years, various training programs have been developed, focusing on specific programming languages, serving different purposes. In the following analysis, readers have the chance to browse through the most popular, web-based applications that focus on web technologies. They may also discover the wide variety of the opportunities offered online and select the most suitable environment for their education, based on their preferences and their needs. This report is the outcome of scanning more than one hundred web-based, educational environments and the findings are related to the most prevalent of them (32 in number), which fulfil certain conditions (such as those of active learning).

The majority of the online code playgrounds (henceforth CPs) have some primary similarities regarding structure and layout. Typically, they consist of one or more input fields (depending on the number of technologies in use) and an output area where the result of the code execution is displayed.

Along the way, the different functionalities and the comparative advantages of the most popular CPs are going to be analysed. The information provided has assisted us in recognizing their strengths, their weaknesses, the most in-demand features and hence more accurately go on to implement our full stack web development teaching system.

Most of them focus on client-side technologies, as the implementation and the maintenance of systems that support server-side technologies are more complex, enclose limitations and security issues and, furthermore, target a smaller group of users.

Online presence, ease of access and totally or partially free provision of services to the public have been the basic criteria for including existing applications into the comparative analysis below. Also, their popularity and number of references on technology websites, forums and blogs have been a significant metric.

First of all, it would be helpful to report the data under consideration. The examined environments in alphabetical order are the following:

- Codecademy (https://www.codecademy.com/)
- CodeMagic (http://codemagic.gr/)
- Codepan (https://codepan.net/)
- CodePen (https://codepen.io/)
- Codeply (https://www.codeply.com/)
- CSSDeck (http://cssdeck.com/)
- CSSDesk (http://www.cssdesk.com/)
- Dabbler (http://dabbler.org/home/)
- Dabblet (http://dabblet.com/)
- Dash GA (https://dash.generalassemb.ly/)
- ESNextbin (https://esnextb.in/)
- FreeCodeCamp (https://www.freecodecamp.org/)
- HowJS (http://howjs.com/)
- HyperDev – Glitch (https://glitch.com/)
- IDEOne (https://ideone.com/)
- JSBin (https://jsbin.com/)
- JSdo.it (http://jsdo.it/)
- JS.do (https://js.do/)
- JSFiddle (https://jsfiddle.net/)
- JSHint (https://jshint.com/)
- JSLint (https://www.jslint.com/)
- JS Nice (http://jsnice.org/)
- Khan Academy (https://khanacademy.org/)
- LiveGap Editor (http://editor.livegap.com/)
- Liveweave (https://liveweave.com/)
- Pastebin (https://pastebin.com/)
- Plunker (https://plnkr.co/)
- Repl.it (https://repl.it/)
- Runcode – Rextester (https://rextester.com/)
- SQLFiddle (http://sqlfiddle.com/)
- Udacity (https://www.udacity.com/)
- W3Schools (https://www.w3schools.com/)

Given users' free access to all platforms reviewed, either by creating an account or directly by entering the website, the criteria by which the benchmarking is performed are as follows: Web technologies, automation, sharing and storage tools, error control, samples and user interface (UI). The displayed findings (percentages in tables 1 and 2) derive from an exhaustive list of the CPs, which have been minutely evaluated as per each of the mentioned functionalities and features and the results have been condensed into representative proportions.

3.1 Web Technologies

As observed in Table 1, an extremely high percentage (96.9%) of teaching systems is centred on client-side technologies. Even in programs involving server-side technologies, the presence of client-side technologies is a fixed value. The percentage of programs that offer server-side technologies reaches 38%, while only 3% of the websites reviewed provide solely server-side technologies. Specifically, only one web-based application out of a total of thirty-two concentrates on database management (SQL Fiddle).

Table 1: Percentage of the examined teaching systems that provide client-side and server-side technologies.

	Client-side	Server-side	Both client-side and server-side	Only client-side	Only server-side
%	96.9	37.5	28.1	62.5	3.1

3.2 Automation

Autorun, autocomplete and syntax highlighting are useful functionalities that not only facilitate code execution, but also help users understand concepts and syntax. Almost all CPs (96.9%) provide such features. Nonetheless, automation is not a primary criterion while evaluating a CP. Typical examples include the extremely popular CodePen CP (lack of automatic input completion) and the JSFiddle CP (no automatic code execution).

3.3 Sharing and Storage Tools

As it is evident in Table 2, almost 91% (90.6%) of these platforms provide ways to store, share or export projects. Thus, users are able to store code snippets, share URL links or export files to other platforms. 9.4% of the examined applications do not provide any means of storing or sharing the generated code. However, those systems (debuggers), e.g. JSHint and JS Nice, are exclusively designed for error checking (syntax, indentation, etc.).

Table 2: Percentage of the examined teaching systems that provide sharing and storage tools.

	Store	Share	Export	At least one	None
%	81.3	65.6	40.6	90.6	9.4

3.4 Error Control

Identifying and highlighting bugs in code while implementing programs is perhaps one of the most useful features of a CP. It is of utmost importance, especially for novice developers. Not all applications provide error checking. In most cases, MOOCs offer systematic user guidance and assignments of specific commands to execute. Nevertheless, that is also a feature of some typical CPs, such as Liveweave, and JSFiddle. 43.8% of the examined platforms provide this functionality.

3.5 Samples

Samples or templates include existing code snippets that are available for practising on a CP. This feature provides users with the opportunity to experiment with a variety of examples and discover ideas while exploring samples or other users' projects. 53.1% of the reviewed CPs offer such templates.

3.6 User Interface (UI)

It concerns the design, structure and layout of a CP's interface. It includes content as well as page formatting, which can be simplistic or complex, with a modern or outdated design. Colours, fonts and images, may also influence visitors' impressions. In addition, the options available in the workspace, undoubtfully affect users' experience and are considered to be primary success criteria.

Some basic features that could be mentioned in this section are the output preview alternatives (e.g. full screen, new tab etc.) and the rearrangement of a page's input fields' position.

All in all, such criteria are somehow subjective. The best practices usually follow contemporary design standards that facilitate users' experience.

4 IMPLEMENTED FULL STACK TEACHING SYSTEM

4.1 Structure and Functionality

In late 2018, a new, web-based application (available at https://codetrip.gr/) that integrates basic functionalities of pre-existing CPs was implemented. It is aimed at comprehending modern web development, further facilitating users' experience and encouraging their engagement in programming through a simplified, educational environment. In addition, this application combines the use of both client- and server- side technologies, which expedites the training process.

Expository, active or interactive learning (Means, Toyama, Murphy, Bakia, and Jones, 2010) is achieved through the use of a single platform. The programming languages available for practice have been selected on the basis of their popularity (rank among the 10 most widely used programming languages according to Redmonk and Tiobe's latest reports), but also their usefulness in the development of integrated applications. We have taken into account the varied knowledge levels and the different

backgrounds of learners. We intended to cultivate multiple skills through heterogeneity. For these reasons, the web technologies we have selected and are available for use in our teaching system are: HTML, CSS, JS (client-side) and PHP and MySQL (server-side). Undoubtfully, plenty of other popular technologies (such as Python or Node.js) could also serve the learning process.

The development of an online application rather than a desktop one has been an easy decision due to numerous factors, such as efficiency, convenience and flexibility. Especially when it comes to such software, direct access and instant execution of users' commands play a leading role in acknowledging it as serviceable and preferable to others that may require installation (waiting time) and occupy precious space in computers' hard disks.

As the architectural design of the system is concerned, the technologies that have been used for the implementation of the CP are the same as the ones provided on the platform (HTML, CSS, JS, PHP and MySQL). Several libraries, such as Bootstrap and jQuery, have also been utilized.

The website provides a particularly friendly interface with engaging content, simple navigation, systematic user guidance and responsive design. Its features include sample tools, fragments of code for experimentation, error warnings and notifications, graphic representations of client and server communication, support material, articles with links to valuable resources, explanatory paradigms and extensive references to commonly used concepts.

When using a CP for educational purposes, saving projects is of paramount importance for making progress. In the said CP users are welcome to work on multiple projects and keep track of their work by storing files, which are available for further processing in future sessions. In this way, they maintain their entire work concentrated in one place and can look back at previous files, when needed. Tutors and learners are able to check code executions, experiment and practise using the aforementioned technologies, all-in-one. Hence, there is no need for switching programs, in order to serve different purposes.

As the workplace of the implemented CP is concerned, it should be clarified that server-side technologies are only available after users' submission. So, does the save option. Then, the logged in users' home page is modified and adjusted to their extended capabilities as subscribed members. Additional features such as code editors' syntax highlighting, autorun and autocomplete are also provided (Figure 1). Users can view, revise and enhance previous projects, which are saved in chronological order and categorized by technology (Figure 2).

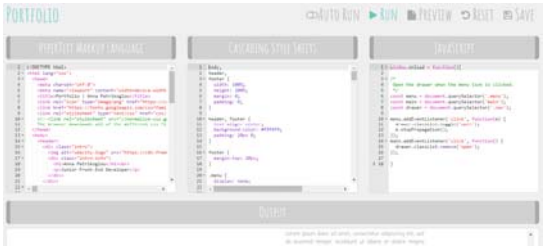

Figure 1: A previously saved web project that has been named as "Portfolio". Whenever needed, the user can alter the code inside all three editors and apply changes to the final file. Note: Useful features are also provided thanks to the embedded Ace Editor.

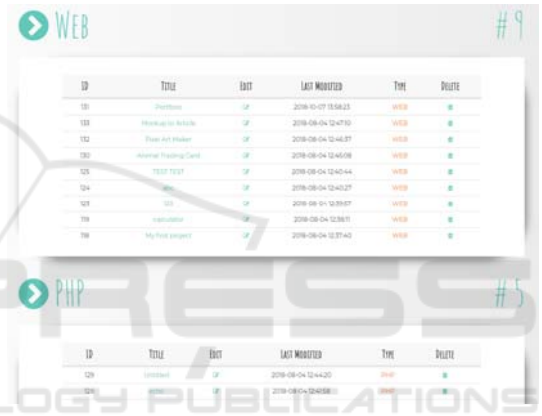

Figure 2: A list of all projects that have been created by the same user, grouped by technologies in use.

An effort has been made to provoke learners' interest and encourage their curiosity by employing appealing graphics and diversified sections. For instance, the SQL playground is constructed upon a database that contains real information about well-known technology "gurus" that have marked a new era of CS (Figure 3). In that way, users acquire general knowledge of contemporary issues and industrial needs.

Learners are encouraged to search for unknown concepts, functions, techniques and solutions, in every applicable way. For this reason, several tools and techniques have been employed. For instance, the application embeds an input field, where users can insert queries. They eventually get directed to several answers, as results of a search on a popular search engine. The goal is to memorize that process and get used to finding the keys to their problems without necessitating extensive guidance.

Figure 3: The SQL playground with the default tables (Users & Groups), which are editable and provide further knowledge of the recent history of technology and personages that have made a difference. Note: A new database is created every time a new user is registered on the system (for their personal use and practice). Users can modify or delete existing tables, create new ones or even drop the database. In that case, the button “Database” restores default data.

4.2 Proposed Learning Path

A beginner with zero programming skills usually looks for support, supervision and step-by-step directions. MOOCs are highly recommended for this purpose. These educational environments offer comprehensive courses per technology, with consistent guidance by knowledgeable tutors (or content creators) and deliberate focus on fundamental concepts, functions and syntax. Websites, like Udacity, Codecademy and FreeCodeCamp, offer high-quality classes, divided into modules that cover specific technologies. Once a module is completed, the next one follows, which is usually based on newly acquired knowledge and enhances users' understanding and performance. Through exercises and examples, learners get introduced to programming languages and gradually master how to use them. They also have the opportunity to participate in forums that are formed within these environments. As a result, they can communicate with each other, exchange ideas, views and concerns, get informed and receive advice and suggestions from experienced developers. The education process takes place asynchronously, in preferable hours and days. Occasionally, there is a time limitation on the completion of each course. That helps to alert learners and keep their interest vivid.

Several courses that cover a wide range of basic terms and programming methods, are available for free. Even platforms that provide billable courses, offer part of their services for free and attract thousands of users.

Following an elementary exposure on the before-mentioned platforms, it is proposed to utilize typical CPs for project implementations. This practice helps to promote learners' research aptitude, improve their programming skills and discover new ways to deal with problems they may be facing. The pattern is simple and brings impressive results. The key to improvement lies in repeating the actions of searching, reading and applying code. The active learning approach has proven to be particularly effective, at least in terms of CS courses, where learners’ active participation plays an important role in perceiving complex concepts, applying latest techniques and adding new technologies to their toolkit.

Nowadays, information accessible online is countless and a wide variety of educational destinations awaits to be explored.

5 CONCLUSIONS AND FUTURE WORK

As mentioned above, the environments compared in the analysis compose a representative sample of the most popular online educational platforms. The research has been conducted under particular limitations, which include online presence, free access and availability of the examined web technologies (HTML, CSS, JS, PHP, MySQL). It should be pointed out that, as time goes by, the services offered by different providers are frequently altered, enhanced or even suspended, as that dynamic, web “ecosystem” evolves. As observed, several of the scanned CPs have halted their operation, while others emerged.

Determining the most serviceable CP among the most prevalent ones depends chiefly on somebody’s preferences, experience level and educational or professional needs. Although several distinguished platforms can be found online, there is no particular environment that incorporates all existing and widely used web technologies. In addition, each one of them employs different functionalities, that makes the selection process purely subjective. In some cases, the popularity of a platform is attributed to its attractive environment, whereas in others it is the powerful presence of a community. The crowds of active users, the sufficient, auxiliary material or the combination of them may also bring a CP to the top of the list. Likewise, a crucial factor could be a specific feature, which at a particular moment serves a user's needs. It is evident that an ideal environment, especially for beginners, should include distinct commands and simplified functions that assist the learning process and prevent unnecessary browsing. At the same time,

it should also provide learners with a modern design and a pleasant - not obsolete - UI, as the learning process is burdensome and can be even tedious sometimes.

Oftentimes, especially when it comes to ever-changing fields, such as CS, the rich profusion of information and the issues that emerge throughout research may lead to temporary deadlocks. For this purpose, diverse learning method testing, and concurrent use of different tools are proposed. It is important to clarify that information on the web is diffused rather than concentrated on a single teaching platform. No educational environment holds absolute knowledge of a study object. Frequently, it is necessary to utilise multiple resources and medians, in order to achieve the desired outcome. For this purpose, websites with diversified content such as guides, articles, freeware, etc. are highly recommended.

In summary, this paper aims at expediting the teaching process by proposing a recently introduced approach, where learners retain an active, research attitude. In a rapidly growing industry, the prospects of development and optimization of emerging applications seem infinite. There will always be more advanced, helpful and complicated features to enrich educational programs with. Questioning and testing technologies in use are key points for a successful evolution.

Future research should further develop and confirm initial findings by providing deeper, more detailed and extended data derived from modern educational platforms combined with modern teaching methods and techniques. Moreover, the inclusion of cutting-edge technologies, such as AI and machine learning, in the learning process is definitely an indispensable next step.

Future in class testing and evaluation of the newly implemented full stack web development system is anticipated to bring positive outcomes and corroborate our estimations.

Considering every aspect of an effective teaching system is undoubtfully an arduous task to tackle. To the best of our knowledge, there has been no perfect environment for practising programming languages. That is the reason why further analyses will produce more accurate insights regarding the principles, the practices and the settings that will advance the learning process.

REFERENCES

Andreessen, M. (2011). Why software is eating the world, *The Wall Street Journal*, August 20.

Atkin, J.M., Black, P., James, R., D., E., Olson, J., Raizen, D., P., S., Sáez, M., J. and Simons, H. (1996). *Changing the subject – Innovations in science, mathematics and technology education*, London and New York in association with OECD, Routledge.

Baker, M. (2017). Code alert, *Springer Nature*, Macmillan Publishers Limited, 541, 563 – 565.

Bennedsen, J. (2008). *Teaching and learning introductory programming – A model-based approach*, Department of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo, Norway.

Bergersen, G., R. (2015). *Measuring programming skill – Construction and validation of an instrument for evaluating java developers*, Department of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo, Norway.

Bers, M., U. (2018). *Coding as a playground – Programming and computational thinking in the early childhood classroom*, Routledge.

Chao, J., T., Parker K., R. and Davey, B. (2013). Navigating the framework jungle for teaching web application development, In E. Cohen & E. Boyd (Eds.), *Proceedings of the Informing Science and Information Technology Education Conference*, Informing Science Institute.

Decker, A. (2007). *How students measure up: An assessment instrument for introductory computer science*, Department of Computer Science and Engineering, Faculty of the Graduate School of The State University of New York at Buffalo.

Eisenberg, M., His S. and Oh, H. (2017). Machines and minds: The new cognitive science, and the potential evolution of children's intuitions about thinking, *Elsevier B.V., International Journal of Child-Computer Interaction* 14, 1 – 4.

Ertmer, P., A., Gopalakrishnan, S. and Ross, E., M. (2001). *Technology-using teachers: Comparing perceptions of exemplary technology use to best practice*, Purdue University, Journal of Research on Technology in Education, ISTE.

Raj, A., G., S., Patel, J., M. and Halverson, R. (2018). *Is more active always better for teaching introductory programming?*, University of Wisconsin-Madison, Wisconsin, USA.

Gomes, A. and Mendes, A. J. (2007). Learning to program - difficulties and solutions, *International Conference on Engineering Education.*

Gottfried B., S. (1997). *Teaching computer programming effectively using active learning*, University of Pittsburgh.

Harrop, W. (2018). *Coding for children and young adults in libraries: A practical guide for librarians 45*, Rowman & Littlefield, USA.

Jackson, D. and Miller, R. (2009). *A new approach to teaching programming*.

Jenkins, T. (2001). Teaching programming – A journey from teacher to motivator, School of Computing, University of Leeds, Leeds, LTSN Centre for Information and Computer Sciences, *2nd Annual LTSN-ICS Conference*, London, UK.

Kelly, K. (2016). *The inevitable – Understanding the 12 technological forces that will shape our future*, Penguin Books.

Lathrop, S., D. (2010). *Teaching techniques for advanced computer programming*, Center for Teaching Excellence, United States Military Academy, West Point, New York.

Liu Y. and Phelps, G. (2011). Challenges and professional tools used when teaching web programming, J. Comput. Sci. Coll. 26, 5, 116 – 121.

Maiorana, F. (2014). Teaching Web Programming - An Approach Rooted in Database Principles, *6th International Conference on Computer Supported Education*, 49 – 56.

Means, B., Toyama, Y., Murphy, R., Bakia, M. and Jones, K. (2010). *Evaluation of evidence-based practices in online learning: A meta-analysis and review of online learning studies*, Center for Technology in Learning, U.S. Department of Education, ED-04-CO-0040, 0006, SRI International.

Mironova, O., Amitan, I., Vilipõld, J. and Saar, M. (2016). Active learning methods in programming for non-IT students, Tallinn University of Technology, Estonia, *International Conference e-Learning*, 239 – 242.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*, Library of Congress Cataloging in Publication Data, Basic Books.

Péter S. and László, Z. (2003). *Methods of teaching programming*, Department of Informatics Methodology, Eötvös Loránd University of Budapest.

Prensky, M. (2007). How to teach with technology: keeping both teachers and students comfortable in an era of exponential change, Emerging Technologies For Learning – Volume 2, British Educational Communications and Technology Agency (Becta), 40 – 46.

Robins, A., Rountree, J. and Rountree, N. (2003). *Learning and teaching programming: A review and discussion*, Computer Science, University of Otago, Dunedin, New Zealand, 137 – 172.

Silberglitt, R., Antón, P., S., Howell D., R. and Wong, A. (2006). The global technology revolution 2020, Executive summary: Bio/Nano/Materials/Information trends, drivers, barriers and social implications, National Intelligence Council (NIC), RAND National Security Research Division, Library of Congress Cataloging-in-Publication Data.

Wilson, B., C. (2002). *A Study of Factors Promoting Success in Computer Science Including Gender Differences*, Department of Computer Science & Information Systems, Murray State University, Murray, KY, USA, 12, 1 – 2, 141 – 164.

Wilson, B., C. and Shrock, S. (2001). Contributing to success in an introductory computer science course: A study of twelve factors, *ACM, SIGCSE 2101 Charlotte*, NC, USA, 184 – 188.

Xinogalos, S. and Kaskalis, T., H. (2012). The challenges of teaching web programming - Literature Review and Proposed Guidelines, *8th International Conference on Web Information Systems and Technologies*, 207-212.