The 14th International Scientific Conference
eLearning and Software for Education
Bucharest, April 19-20, 2018
10.12753/2066-026X-18-000

PROGRAMMING GUIDE FOR THE SILVER CODE COMMUNITY

Ciprian-Bogdan Chirila
Department of Computers and Information Technology, Politehnica University Timisoara, V. Parvan 2, Timisoara, Romania
chirila@cs.upt.ro

***Abstract**: The digital society evolved in the last decade such that home devices like: telephones, TV sets, refrigerators, light switches, electricity sockets etc. became smart and connected to the internet communicating and taking decisions to assist us in our everyday life. Subway trains became autonomous, automobiles tend to go in the same direction all this progress is based on algorithms for control, image recognition, data mining, machine learning etc. having different levels of complexity. Young people that were born in the digital ecosystem are considered digital natives. They can use easily smart phones, tablets, computers for different everyday life tasks such as: selecting and reading the news, online paying utilities like water, electricity, gas; using ATMs for financial operations etc. On the other hand, elder people which do not have such digital skills do not adapt well to the changes of the society. In order to increase the quality of life for elder people they must be integrated in the continuously growing digital world of the present and of the future through digital education. For the education to have a deep impact we consider that it should go to the bases of the digital fundaments which is programming and specifically coding. Programming concepts for elders are delivered in five modules. The first module presents basic computer programming elements. The second module presents the most important digital thinking elements. The third module presents technical details about a few web technologies like HTML, CSS and JavaScript. The fourth module teaches how to put hands on JavaScript and to build two single page applications. Module five is a glossary of digital terms.*

***Keywords:** Programming, coding, algorithms, elders over 55 years, JavaScript*

I. INTRODUCTION

The digital society that we live in today is contaminated irreversibly with electronical devices. Mobile phones became smarter and smarter each generation embedding a plethora of sensors. Cameras allow taking photos and processing information afterwards. Accelerometers detect the acceleration of the holder. Then GPS (Global Positioning System) detects the coordinates of the holder in order to help him providing directions for orientation. Magnetometers are based on the Hall Effect and measure the intensity of the magnetic field in order to emulate a compass for orientation. Gyroscopes based on axis rotation are used to determine the orientation of the device when in motion. Proximity sensors based on Infrared radiant energy detect the presence of human ears. Light sensors detect the intensity of the device surrounding light. Barometers measure the pressure of the atmosphere. Thermometers measure the device surrounding temperature. All these sensors together with pedometers, hearth rate monitors, harmful radiation detectors etc., enable the creation of applications to be used in our everyday life.

TV sets became smarter, they can connect to the Internet and display YouTube clips and allow browsing. Nowadays refrigerators are also connected to the Internet taking decisions about placing online orders for fresh food. Light switches and electricity sockets are controllable online from the Internet webpages. Subway trains became autonomous, having no driver to drive the train and to open and close the doors to passengers.

All these devices became smart because of the sophisticated programs they run inside. All the actions and behaviours of these gadgets are controlled by computer programs. In order for elders to use such devices with confidence and thus to integrate in the digital world they need to understand how programming works and to train them to write small pieces of code to create such applications.

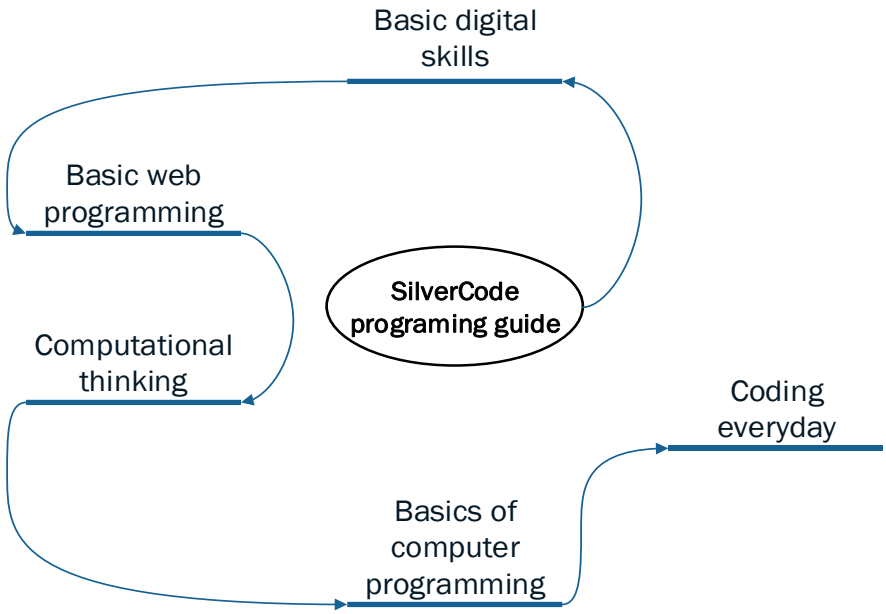

Figure 1. Silver Code project programming guide structure

In this paper we present the structure of a computer programming guide [7] dedicated to elders over 55 years old for learning to program computers, formed out of 5 sections: i) basic digital skills; ii) basic web programming; iii) computational thinking; iv) basics of computer programming; v) coding every day (see Figure 1). The guide is implemented in the context of an Erasmus+ European project named Silver Code and identified by code 2016-1-BG01-KA2014-023736. The participating countries are: Bulgaria, Portugal, Slovenia, Italy, Austria, Romania and Poland. The goal of the project is to form a community of elders over 55 years old around the concepts of programming. We will make an inventory of the presented concepts and then at the end we will check how the guide's final projects use these concepts.

The paper is structures as follows. In chapter II we present related works in the area of computer science teaching. Chapter III presents the structure of the Silver Code project programming guide. Chapter IV presents in detail the guide final projects and makes an inventory of the taught concepts. Chapter V concludes and sets the future work.

II. RELATED WORKS

In [15] is presented the model of a competence platform for primary and middle school students. The competence model has 5 levels of composition with several dimensions inspired from the Bloom taxonomy and auto-adaptation mechanisms based on fuzzy rules. Silver Code project involves a blended learning approach grouped in 5 modules having both face to face meetings and online tasks. Our curriculum is more restricted than the ones in regular schools.

In [4, 5] are presented examples of auto-generated learning objects in the field of computer science. The focus is set on the Data Structures disciplines taught to university students. The approach described in the work for creating learning materials would be suitable in the context of the Silver Code project, but it lacks the financial and time resources. Generative learning objects are a good for exercising and repeating uses of computational concepts having virtually unlimited number of instances and automatic evaluation.

In [1, 2, 3] are set guidelines for the development of educational environments in the automotive industry. The focus was set on teaching the AUTOSAR standard in the context of

monitoring the management, development and testing teams. The goals of the automotive learning project and Silver Code project is similar since both need to establish how suitable the proposed solution is for the target group.

In [8] are presented elements of Blockchain based applications design for the open education. Silver Code project is a form of open education and could benefit from the blockchain technology in issuing certificates and open badges for the elders when accomplishing different modules, according to the project proposal.

[10] presents the use of Java programing language and technology in university curricula across European countries. In this sense JavaScript [6] is a lightweight programming language and a subset of Java being a good choice for teaching elders.

[13,14] describes the integration of virtual learning communities into knowledge management models from academia. Such concepts are useful for the virtual learning community of elders in computer programming.

III. THE SILVERCODE GUIDE CURRICULUM

In this chapter we will present the Silver Code project programming guide curriculum with an emphasis on the concepts and competencies to be used further in writing software applications.

The curriculum is organized in 5 modules plus one module for the glossary of technical terms. Each module contains multiple units. The units were written by different project partners, usually a sequence of related units was designed by a single partner, individually.

3.1 Basic digital skills

Module 1 is about basic digital skills and is composed out of 7 units. Unit 1.1 presents the basic structure of a computer: CPU, motherboard, memory, HDD, keyboard, mouse and the basic actions of the operating system like: startup, shutdown, files and folders operations, shortcuts management, cut-copy-paste operations. Basic actions in MS Word, Excel and PowerPoint are presented. The elders will learn how to write a letter in Word, how to create a budget in Excel and how to design a presentation in PowerPoint. Unit 1.2 teaches the elders how to use a set of Internet applications like: web browsers, e-mail clients, chat applications, namely Skype and social networks, namely Facebook. These skills are useful since elders will understand the philosophy behind these tools and in the near future they will be able to use different more evolved applications doing the same tasks on different devices: mobile phones, tablets and on different platforms: Linux, Android, IOs, MacOs etc. Unit 1.3 presents logical schemas for representing the basic blocks of coding: input, output, decision, assignment, arrows and connecters. These representations will explain the von Neumann paradigm of computing that is used in most of the computer programs. Unit 1.4 presents a set of basic array or vector algorithms like, reading from input, writing to output, computing minimum and maximum values, sorting in ascendant or descendent orders. Unit 1.5 is extending the vector presentation with one more dimension resulting in matrixes. Matrixes are read from input, written to output. Vectors and matrixes are considered mathematical tools with a wide range of applications. Computer graphics is mostly based on vectors and matrixes for displaying images, performing animations in games. Three dimensional vectors may represent coordinates in space or directions and may form graphical primitives like lines, triangles, 3D mashes etc. Two dimensional images are represented in computers as matrixes, so matrix operators become image operators and so on. Unit 1.6 presents subroutines which are stereotypes for functions in structural programming and methods in object-oriented programming. They are a set of instructions grouped together under a name in order to be reused by calling from different contexts. Often routines have arguments and results and the mechanism of transmission for these program entities is important in understanding how they work. Unit 1.7 presents strings and their basic operations: creation, catenation, reverse, character accesses etc. Strings are usually represented as a vector of characters and have specific operations.

3.2 Basic web programming

Module 2 presents the basics of web programming in 3 units. Unit 2.1 presents the HTML [12] markup language and some simple web pages are created. The rationale behind choosing HTML

for learning is the fact that almost all applications tend to go online having user interfaces based on markup languages. In the majority of the partner countries the house utilities bills are payable online through a web application exhibiting HTML user interface. Unit 2.2 presents the CSS (Cascading Style Sheet) [11] formatting tags language in order to be able to layout the HTML pages. Most modern web applications make use of this language which allows creating styles that can be applied to individual or group HTML elements. By style we mean a combination of fonts, foreground colors, background colors etc. that can be applied to paragraphs, headings, tables etc. Unit 2.3 introduces basic JavaScript enabled interactions for web pages in a very brief manner. It is presented the way JavaScript code is inserted into an HTML page, then variable examples with their types, some basic operators, the if instruction, functions and events. In the applicative section of the unit the elders will write a simple application running in the browser displaying a bulb image which responds to mouse click by changing the image with the bulb on into an image with the bulb off and vice versa.

3.3 Computational thinking

Module 3 sets up some concepts necessary for the computational thinking. Unit 3.1 is an introductory module to the concepts of computational thinking: decomposition, pattern recognition, abstraction. Unit 3.2 explains the concept of decomposition and how it must be applied in practice in order to solve a concrete problem with the help of the computer. Unit 3.3 presents the concepts of pattern recognition and abstraction. Unit 3.4 presents the concept of algorithm design which moves the problem from the design phase into the operational stage.

3.4 Basics of computer programming

Module 4 presents basics of computer programming. Unit 4.1 presents an introduction to a set of programming languages like: C, C++, Java, Objective C, Swift, C#, PHP, Ruby, Python, SQL, Visual Basic, LaTeX. Unit 4.2 presents the basics of object-oriented programming in JavaScript: classes, objects, attributes, methods, constructors. Unit 4.3 presents the steps to create a JavaScript application running in a web browser. Unit 4.4 presents four JavaScript applications for the elders to edit and run.

3.5 Coding everyday

Module 5 proposes the creation of two web applications from scratch. Unit 5.1 is a step by step tutorial to build a flower watering web application. Unit 5.2 is a step by step tutorial to build a pill reminder web application. Unit 5.3 presents demonstrative examples of how coding can be used in our everyday life. Unit 5.4 presents demonstrative examples of augmented reality use cases. Unit 5.5 presents demonstrative examples of virtual reality use cases. Module 6 is a glossary of terms and keywords. Unit 6.1 contains a set of keywords from the digital mindset. Unit 6.2 contains a set of online terms that were used in the training materials.

IV. THE FLOWER WATERING APPLICATION PROGRAMMING CONCEPTS

In this chapter we exemplify the use of programing concepts from the guide in the context of a web application the elders will implement in Module 5.

The Flower watering application is a single page web application having three use cases: i) to list the managed flowers on the web page; ii) to add a new flower in the list; ii) to delete a flower from the list. Elders work with the concept of lists and their specific operations. On the same time the flower listing use case is keeping the watering periods up to date, so a user may water a flower and this action can be recorded in the application setting up the next term for watering.

Figure 2. Flower watering application

Figure 2 depicts the main user interface. The implementation is made in modules which are represented by separate files so elders use the concept of modularization in software.

The “index.html” module is the presentation web page where the content, namely the HTML elements are manipulated by JavaScript methods. The tags manipulation is mage through the jQuery library. The main web page contains a hidden flower editing form for filling details like: name, specie, graphical image, watering period expressed in days and last watering date. The main flower panel is visible at page load but is empty. After the loading of the web page some JavaScript code will fill in the list the flowers stored in the local memory, namely in a cookie. If the cookie is empty the application will initialize it with a predefined set of flowers. Afterwards the users may add or delete freely any flowers they want. The “index.css” module contains the formatting elements for the index.html previously presented module. The flower editing form elements (divisions, images, buttons) are formatted by elders with specific font sizes and colors.

The “index.js” module is the starting point of the JavaScript program in the web application. For the “onload” window event a function is written where the flower list object is created and the show method is called to display the flowers in the HTML web page. The “flowers.js” module is the main module of the application. The module models the flowers collection as a class being composed out of methods. Next, we will present briefly the class methods enabling elders to pass messages between objects.

The Flowers() function is used as a constructor for the instantiation of flower collection objects. The constructor gives the name of the class, namely Flowers. This method uses the procedure concept and object creation concept. Elders will exercise the creation of objects in the object-oriented technology.

The Flowers.init() method is used for the flowers collection initialization. If the cookie content for the flower is empty then the next method is called. The Flowers.iniData() method is used for the creation of 7 flowers with some random data: rose, rose spray, purple rose, tulip, daffodil, lily, cactus. Thus, the user will have a starting flower collection to operate with. The Flowers.findById() method is used to search a given flower by its identifier in order to perform on it several operations.

The Flowers.load() method loads all the flowers from the cookie of the browser. Cookies are used as a micro database in order to avoid using real databases which are complex. The Flowers.store() method is dedicated for the storing of the flowers collection in to the browser cookie.

The Flowers.create() method is used for setting up the HTML layout and for the moment when the user wants to create a new flower by showing up its attributes editing menu. The Flowers.doCreate() is the method linked to the button of the new flower effective creation. The Flowers.doCancel() sets up the HTML layout in the flowers listing mode after quitting the flower editing mode.

The Flowers.show() method is used to display the flowers using HTML elements. Each flower object will call its method to get its HTML representation. The Flowers.delete() method is used for the deletion of a flower from the list, it is a method which is linked to the deletion buttons from the flower list. Each button will call the same method but with different identifiers. At the end of the method the storing method is called. The Flowers.water() method is linked to the watering button and updates the last watering date with the current date. At the end of the method the storing method is called.

Module "flower.js" models the flowers individually. The class contains a 6 arguments constructor for the objects creation and also two HTML rendering methods. The two methods are necessary to simplify the composition on two levels in order to simplify their management.

All the materials of the programming guide are published online using a custom Ilias LMS [9] platform deployed at address https://www.silvercodeproject.eu.

V. CONCLUSIONS AND FUTURE WORK

In conclusion we may state that we built a guide for teaching programing dedicated to elders from several European countries. The programming guide has some overlapping content which may be ameliorated through the peer review process that will take place in January 2018 according to the project schedule.

As future work on short term the guide will be used in the context of a pilot testing in all seven countries. In each country 30 elder students will participate in a face to face pilot testing and afterwards recommendations will be compiled and applied on the online materials.

[1] Bogdan, R., 2016. *Guidelines for developing educational environments in the automotive industry*, Interaction Design and Architectures, Issue: 31, pp. 59-73.

[2] Bogdan, R., Ancuşa, V., 2016. *Developing e-learning solutions in the automotive industry*. World Journal on Educational Technology: Current Issues. 8(2), 139-146.

[3] Bogdan, R., 2017. *Sentiment Analysis on Embedded Systems Blended Courses*, Brain-Broad Research In Artificial Intelligence and Neuroscience, Volume 8, Issue 1, pp. 35-41.

[4] Chirila, C.B., 2015. *Auto-Generative Learning Objects for IT Disciplines*, In Proceedings of the International Conference on Virtual Learning 2015, ISSN 1844-8933, Timisoara, Romania, October.

[5] Chirila, C.B., 2017. Auto-generative learning objects in online assessment of data structures disciplines, BRAIN - Broad Research in Artificial Intelligence and Neuroscience, vol. 8, no. 1, Bacau, Romania, April.

[6] ECMA International, 2017. *Standard ECMA-262 ECMAScript 2016 Language Specification*, http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[7] Georgieva, V., Antunes, M.E., Chiesa, E., Sirk, K., Koc, E., Kobylarek, A., Chirila, C.B., 2018. *Silver Code Training Materials*, www.silvercodeproject.eu.

[8] Holotescu C., 2018. *OpenEduChain: Design and Applications of a Blockchain for Open Education*, In Proceedings of the 14-th International Scientific Conference eLearning and Software for Education (ELSE), Bucharest, Romania, April, accepted.

[9] Ilias Team, 2018. *Ilias E-Learning*, www.ilias.de.

[10] Ivanovic, M., Budimac, Z., Mishev, A., Bothe, K., Jurca, I., 2013. *Java Across Different Curricula, Courses and Countries Using a Common Pool of Teaching Material*, Informatics in Education, 12(2), 153-179.

[11] Mozilla Corporation, 2018. *Cascading Style Sheet (CSS)*, https://developer.mozilla.org/en-US/docs/Web/CSS.

[12] Mozilla Corporation, 2018. *Web technology for developers*, *HTML*, https://developer.mozilla.org/en-US/docs/Web/HTML.

[13] Strunga, A., 2015. *The Integration of Virtual Learning Communities into Universities' Knowledge Management Models*. In A. Alevriadou (Ed.), 7th World Conference on Educational Sciences. vol. 197, pp. 2430-2434.

[14] Strunga, A., 2015. *Using virtual learning communities in shaping the professional identity of primary and preschool pedagogy specialization students: a knowledge management approach*. In E. Soare & C. Langa (Eds.), 6th International Conference Edu World 2014: Education Facing Contemporary World Issues, vol. 180, pp. 460-467.

[15] Vlasin, I., Chirila, C.B., 2014. *The model of a competence development platform for primary and middle school students*, In Proceedings of Social Media in Academia Research and Teaching 2014 (SMART 2014), ISBN 978-88-7587-712-5, Timisoara, Romania, September.