CBSD – A Suitable Solution for Building a Centralized Educational Library of Software

Irina-Miruna RADU
The Bucharest University of Economic Studies, Romania
miruna.radu@outlook.com

The educational system is one of the few systems that have not found until this date a worldwide accepted solution when it comes to building educational apps. And knowing the importance of educational systems in developing future generation, further research is important in this field. This article analyses one possible solution in building a centralized library of educational pieces that can be reutilized through different applications. The methodology name is component-based software development and through this paper, it is investigated the way this can impact the way educational systems are developed. To achieve this goal different studies are analyzed to underline the most common advantages and disadvantages of CBSD, which then are analyzed through the aspects and problems that the current educational system is facing.
Keywords: *CBSD, Component-Based Software Development, E-Education*
DOI: 10.24818/issn14531305/24.1.2020.07

1 Introduction

In recent years, technology has evolved exponentially and influenced the way of living. This impact was seen even in domains that in the past it did not seem possible to introduce it for a clear benefit. No matter if we talk about the healthcare environment, juridical field and even educational system they all felt obliged to keep up with the trend to still be relevant for the current society. The paper will focus on the transition from traditional education to e-education and how the methodology of component software development can help in smoothing this process.

More and more countries are turning their attention in implementing a digital environment for education because having an educated society that is a "digital native" one can be the key to a great civilization, one that can help in growing a community globally. Having a worldwide process of digitalization can bring lots of benefits like having the knowledge, experts and other resources public accessible for anyone at any time. But to be able to conquer the full potential of educational software developers and also the educators need to give importance to ways in which technology can help without hurting the process of learning. In a recent publication [1] that analyzes 126 evaluations of ways in which technology is used in education, it is shown four important key results. First one is regarding the fact that even though access to computer and internet alone do not improve grades, it can increase computer proficiency. The second result shows that educational software that is well designed to help students acquire skills in their own passing has shown that it can be useful and has great potential. The third finding presents the fact that small piece of one-time action like reminders can be really useful and impact on various education-related tasks, sometimes at a lower cost. The last discovery is the most expected by the educator community and shows that online and in-person instructions are combined they can work as traditional in-person only classes. On the other hand, the study also showed that students in online-only courses tend to perform worse than students in in-person-only courses.

Besides all of this, when constructing an e-educational tool it is important to have in mind that for a good blending education the developer needs to take into consideration different teaching styles, different curricula and different needs that pupils can have because all of these are particular from region to region, institution to institution. So, getting to a point where the educational tool meets the expectations that the public eye has a lot of effort should be concentrated into it. On these grounds, where these cannot be a one man job,

but instead a community effort it is clear why it is expected to analyze methodologies that can transform the dream into reality. The one that makes the subject of this study is about introducing the mindset of software reutilization in the development of e-education.

So, this paper aims to explore how the community can benefit from creating a worldwide library of pieces that can be used and reused and refined through different needs and expectations. These components can benefit from expert knowledge and help in growing from everyone interested in helping the system. For the first part of this paper will try to analyze synthesis and present what are the most mentioned advantages and disadvantages of component-based software development in articles, studies and conference papers, throughout the years. In the second part, all the findings will be seen through the eyes of the building and educational tool. So, the advantages will be analyzed through the impact they have in developing a collaborative environment for building educational components. The problems of component-based software development that the analyzed studies outlined will be reflected throughout the way they are impacting on e-education.

The study will be an eye-opener towards how CBSD opportunities and challenges are seen in the eyes of the specialists as their opinions emerge from scientific articles and to come to a conclusion if developing a library of components can be an option for helping in increasing popularity of e-education.

2 How CBSD emerged

More than fifty years ago, McIlory [2] proposed to the world of software developers a new concept that was inspired by the industrialization of engineering fields, a concept that could change the mentality of the community and could help the process of bringing software to the market in a faster rhythm. This idea is based on commercial component production, building a product from little pieces that can be manufactured anywhere and be used by different companies into different projects. This way the methodology of Component-Based Software Engineering, also known as component-based software development (CBSD) arisen.

In the modern days, the CBSD methodology can be easily explained by thinking of it as a Lego game. There is one big software system that needs a lot of effort to be built, but this one can be broken into smaller pieces, just like Lego parts, that can have a function on their own and can be used in other systems as well.

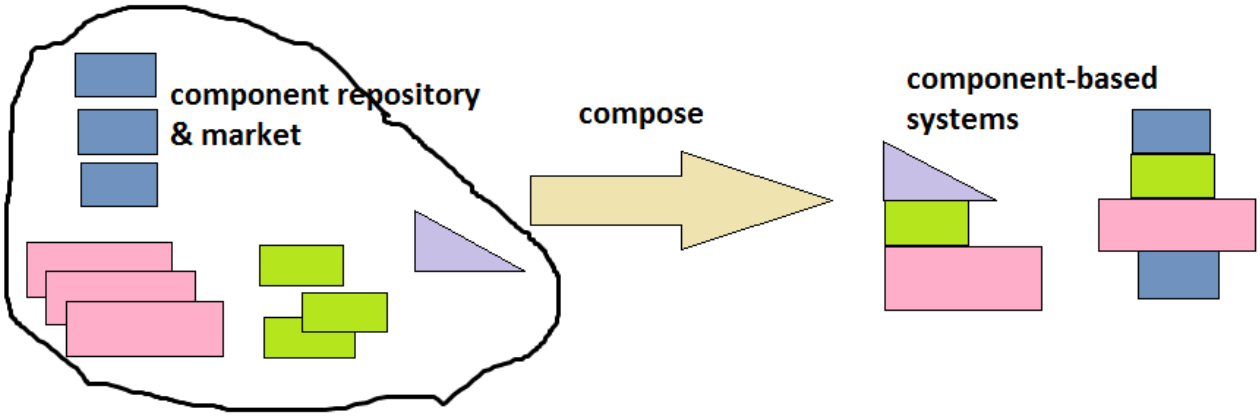

Fig. 1. Component-based software development

As shown in Figure 1 usually there are different components, little pieces on the market, that can be chosen to be put together in order to resolve a more complex need. It at some point a piece becomes obsolete it should be easily swapped out with a new one. So all components that were built keeping in mind the purpose of reutilization can be later put in a library and can be used to build different types of systems, systems that can continually

grow and transform through replacing or updating different components. [3]
Therefore, in the component thinking spirit, the development team starts from the systems' requirements and tries to cover most of them with components that already exist on the market. For the more specific needs of the product, the developers create the necessary parts as standalone pieces that respond to certain purposes and that can be used and reused over time to build something new. Every little piece is working on its own and can be reused with little or no modification. For the success of this architecture, the parts must be well documented, with less to no effort to understand, accessible for deployment, simple to update to recent versions and painless to replace by other pieces.

3 More about Components

The specialty literature has a few definitions when it comes to software components; these pieces that can be put together to build more complex applications. One definition presents software components as being an encapsulated container that is platform free and written into a neutral language and that can be accessible from the outside through an interface [4]. Microsoft Corporation emphasizes that a component is a unit ready for packaging, distribution or delivery that brings services within encapsulation boundary or data integrity. In another definition [5] a software component is seen as a unit that has an interface following some clear specifications and designed for a certain context, having some specified dependencies. Moreover, a software component can be deployed as a standalone part, independently. To summarize a software component is a piece with a well-defined interface that is platform free, reusable and encapsulated to be ready for distribution and delivery independently from other components. In addition, it is important to remember that a component is built to solve a need, a certain scenario of utilization.
The components are built in the idea of reutilization, idea that has a word saying in the way the architecture and also the development should be structured and planned. To be able to bring benefits to this kind of development, the components should fulfill some requirements as mentioned in [6] :

- Standardization – CBSD components have the data, interface, composition defined to meet a standard component model
- Independence – a component should not be linked with other components be particular needs, instead, they should be able to be deployed and composed with little effort. In the case they have particular requests, they should be specified in the interface
- Documentation – to enable the full potential of components they require to be very well documented so the developers can quickly understand if that component can be useful for their project
- The ability to be easily deployable – a component has to be self-contained and must have the behavior of a stand-alone entity. Usually, a component is developed in a binary way, with no need for compilation before deployment. If a component is a service, the deployment is on the service provider side.
- Composability – a component must be able to interact with other pieces and for this, it must have a public specified interface, with information about its methods and attributes.

4 Types of Reutilization

As mentioned before, components can be seen as little boxes formed from code and documentation. Depending on the access to this information, components can be developed having different types of reutilization in mind [7] :

White Box Reutilization

White box reutilization means that the developer gains access to all the parts of the component no matter if it is the interface, the documentation or the code. In case that the component requires to be improved with additional functionalities, it can be easily modified through inheritance or delegation to solve the newly appeared needs.

Black Box Reutilization

In the case of black box, the developer that wants to use the component gets access only to the available interface, not having any kind of information regarding the way the component was implemented. In this situation, the component must have a clear documentation of the methods and restrictions necessary.

Glass Box Reutilization

Glass box reutilization is a type of reutilization where the user can see both the exterior and the interior but cannot "touch" the interior. Even though the developer cannot change the implementation, he can still have a better understanding of the solution, making the interaction with the component clearer.

5 Research Method

For the purpose of this article, the research method includes different stages: searching for studies related to CBSD, evaluating their information, determining which the most encountered characteristics are and relating the findings within the topic of education.

The quantitative approach was chosen to complete the purpose of this paper. Multiple research papers, 10 papers for the advantages and 7 for the challenges, where studied to present a closer to reality overview on the impact CBSD has on software development. The conclusion on the suitability of using this methodology for creating a centralized library of pieces that can be used for helping the educational system in building applications that can be relevant in the global community is drawn after analyzing the discoveries from the point of view of the needs of education.

Paper selection

In the first stage, databases like Google Scholar, Science Direct and others were searched for significant studies. For this selection were chosen publications from 2010 up to now, that contained in their title keywords like:,, component-based software development", "component-based software engineering", "advantages" , "challenges", "disadvantages".

In the next stage, all the abstracts were dissected and the studies that did not respond to expectations were rejected from the research. In the last step, every article was deciphered and only for the ones that were considered relevant, the advantages and disadvantages were outlined in order to draw attention to the most important ones.

Process

After the selection of the articles that were suitable, the advantages and disadvantages discussed were extracted. The ones that were most present through the studies are the ones that will be present in the paper. They will be analyzed after investigating some of the problems of the educational system to discover how they can bring value to it.

Results

In Table 1 is presented an overview of the findings. Every advantage and challenge is shown in relation to the articles in which it was mentioned. This way it is easier to understand which the most relevant features on CBSD are as they emerge through the specialized literature. For example, it is easier to see that reusability is a key characteristic that was mentioned in all the studies that were relevant for the advantages.

Table 1. Results

Advantages/ Challenges	**Category**	**Description**	**Citation**
Advantages	Reusability	Reusability refers to being able to use one component for different types of applications where it can solve different needs.	[8]; [9]; [10]; [11]; [12]; [13]; [14]; [15]; [16]; [17]
	Reduce development time	Giving the fact that sometimes the developer has to choose the component he wants to use instead of building everything from	[8]; [9]; [10]; [11]; [12]; [13];

		scratch, lots of studies consider reduce development an important advantage.	[14]; [15]; [16]; [17]
	Reduce costs	Building software from scratch can be really expensive, but building software from components that already exist on the market can sometimes be a cost-efficient solution.	[8]; [10]; [11]; [12]; [13]; [14]; [15]; [16]; [17]
	Improve quality	Components that are used in a variety of applications can improve the quality of the software that uses them.	[8]; [9]; [10]; [11]; [12]; [13]; [14]; [16];
	Increase productivity	Using components that have evolved through time and different types of applications and testing can really help transmitting expert knowledge throughout different systems.	[8]; [9]; [10]; [11]; [12]; [14]; [15];
	Reduce complexity	Having to build an application from smaller part with simple functionality can transform the nightmare of building a big application into something simpler.	[8]; [14]; [15]; [16]; [17]
	Maintainability	Having to change some functionality in the application can be easier when you only update or replace certain parts of the application and try to figure out how to include the new needs in a big monolith.	[8]; [13]; [14]; [16]
	Easier software construction	In CBSD, software construction can be strip down to choosing the right components for you needs and combining them, like Lego pieces, in building the solution.	[16]; [17]
Challenges	Finding suitable components	Finding the right components for the needs of certain application can be really a pain. On the market there are lots of components not well documented or having any endorsement from other users. Sometimes the developers have to choose a component that does not completely cover the functionality they desire. In this case they either complete the functionality on their own; either they rethink the functionalities of the software.	[8]; [9]; [11]; [13]; [14]; [15]; [18]
	Maintenance	Maintaining a component to the higher standards can really be a challenge. Trying to keep the component up with the newest technologies and discoveries from the market while keeping it standardize and general as possible can require a lot of effort from the development team.	[8]; [9]; [13]; [14]; [15]; [18]
	Updates	There are few companies that keep updating components after their selling. In most cases when a developer wants to update a component by its own it has to either alter	[8]; [9]; [13]; [14]; [15]; [18]

		the code, in case it has access to it, either to replace the component with one that is more suitable.	
	Ambiguous Requirements	When building a component, the developer has to have in mind different restrictions like: the component must be generic, with few to none restrictions regarding architecture or environment and has to solve a certain need that it can repeat itself in numerous software.	[8]; [9]; [11]; [14]; [18]
	Interoperability	In cases where the component is developed in a black box matter, where doesn't have a good documentation is hard to understand everything will look in the great scheme of things.	[8]; [14]; [15]; [18]
	Testing	Even though the components claim to be tested as individual parts, testing them after combined can be a really hard thing to do.	[8]; [11]; [13]; [18]

6 E-education Nowadays

To be able to correlate the gathering regarding CBSD methodology with the educational system, firstly it is needed to find more about it.

In recent years, an excitement around the way technology can have a significant saying in education system can be seen. Various countries are trying to concentrate their power in this direction. Even throughout all of these efforts there are still pupils that do not have access to an internet connection at home and for whom having a computer for solving homework is a faraway dream.

What is education lacking today?

In today's society it is important to form individuals that are capable to use technology as a helper in their day-to-day life. As the generations are evolving, more and more jobs require these skills. Modern education requires developing the next civilization, a digital native one. To be able to deliver solutions for it, solutions that can emerge rapidly to be relevant to current society and gain support worldwide it is relevant knowing the obstacles and needs of the system now. Even though from region to region it may seem that there are differences, the problems different education systems face can be generalized to similar issues throughout the globe.

As mentioned in [19] , [20] and [21] the difficulties can be summarized as:

- lack of funds and other resources: intuitions worldwide are faced with restricted budgets that limit their power in changing the poor infrastructure and equipment and with less human power than needed that obstructs the time of innovation to doing necessary things
- lack of executive power for the teachers: important decisions regarding the educational systems changes are not in the hands of the teachers. Furthermore, when having innovative ideas teachers need to go up against a barrier of paperwork and administrative and management office to explain the importance of that change
- lack of access to development: sometimes for using certain application the teachers need trainings on how to use them, or forming other skills to operate them. Usually the access to this kind of resources is hard get and the teachers are discourage in using the technology in their lectures
- lack of curricula updates: the curricula is outdated, not focusing on the creative thinking skills critical for the digital native generation, but on the ability to memorize and stock information that it is not such a critical feature for today's society
- lack of personalized learning paths: each individual in our society is different and

these difference makes him special. Therefore, the way he learns, discovers and transforms the information he receives from outside is also unique. For developing a great civilization of individual human beings that are at their best, the educational system must be able to offer personalized learning paths

The impact of technology in education

A recent study [1] has shown the ed-tech can really go beyond the hype and the marketing and can really improve the way pupils are learning when applied correctly. Some of the findings can be kept in mind when thinking about building a library of components for developing more the educational systems.

Firstly, an important key result is the fact that granting pupils access to computers and internet alone do not improve their grade so this imply the need to build and give access to applications specially developed for them. Another fact of interest for this research is the idea that the students respond well to educational software developed for particular skills at their own pace. And the most important finding is that blended learning can work as well as traditional classes. Moreover, students in in-person-only courses tend to perform better than the students in online classes only.

The ideal educational software

From these findings it can be draw the image of an ideal education technological environment. A software that should be cost-efficient, easy to use, modular per pupils needs, intertwined with the in-person assistance, quickly to modified, little to none effort in updating it and making it relevant, and that can share expert knowledge worldwide with ease.

7 CBSD suitable for Building Educational Software

This paper proposes to analyze the advantages and challenges that building a library of components for educational system can bring on the way. All of this will be reflected through the findings discovered previously.

Benefits of CBSD

CBSD can bring lots of benefits when implemented right. As shown in fig. 2 the most reiterated advantages through the selected articles are: reusability, reduction of time to market, cost reduction, quality improvement.

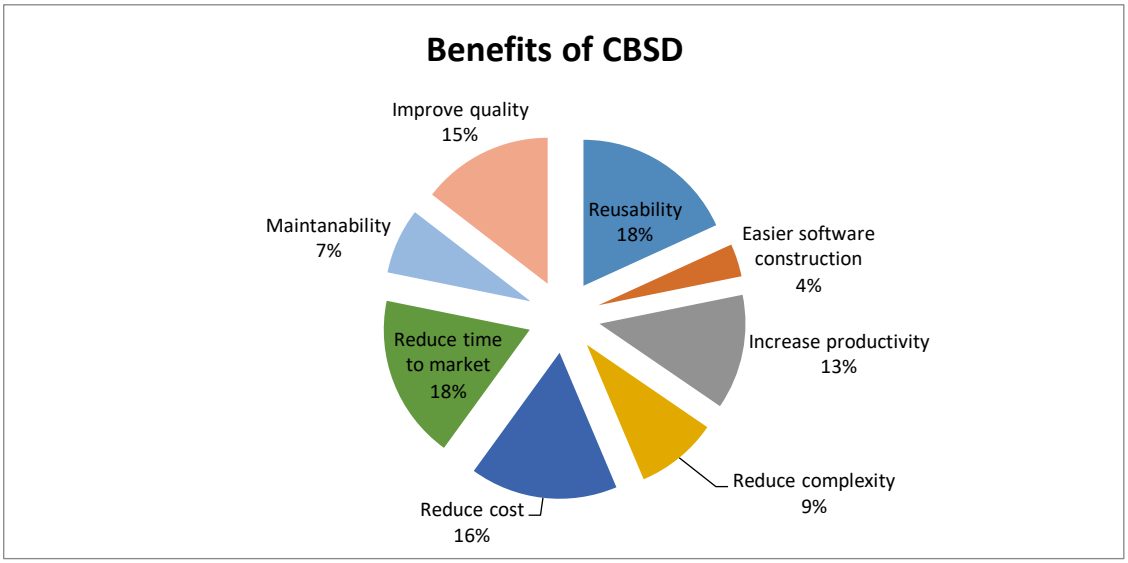

Fig. 2. Benefits of CBSD

Increase in reusability

The concept of reusability in CBSD it the most important characteristic of this methodology has. Different component can be assembled and reassembled together to serve different purposes. [9] Components are developed to respond to particular needs, needs that can repeat themselves throughout different systems. For example, a login component can be used by a variety of software platforms. If in some cases new requirements arise then the component can be updated.

In the perspective of building educational software this concept can really be helpful. Developing a library of components specialize on different topics related to education like different types of exercises, different types of learning methods, covering different needs of education can create the basis of access to resources in building the best e-educational system for our children, a system that is able to deliver personalized learning paths based on pupil needs. These components can be available to use worldwide, a fact that can lead to a unified knowledge of education, with access to different experts in this field. Every teacher can access the resources that he thinks fits best his pupils and can help them to assimilate skills for life faster.

Reduction of development time

When building an application from assembling components, where most of them already exist on the market it is clear that the amount of time necessary during the development phase it is less compering with building a totally new application from scratch with all the designing, coding and testing involved into it. In addition, if the components already exist in other systems and they had already been validated as efficient, user-friendly, bug-free it is easier to introduce the application to public and it can be more positively received and embraced.

In a continuously changing world, where people need different skills from generation to generation in order to improve quality of life, it is important that ed-tech keeps up being relevant. Being able to choose from different components the ones that are relevant for the current times and being able to be combined for the wanted application it can really reduce the time needed for bringing actualize education content to the market.

Cost Reduction

Lots of the studies highlighted the fact that CBSD has an impact on the budget needed for development. Instead of building software from zero, the development is based on integrating pre-fabricated software components [8]. This way less man-power and resources are needed in developing software, things that can lead to savings.

Bearing in mind that the main problem of education is the reduced budget, the advantage that CBSD proposes can be a game changer in making educational software accessible to more institutions and pupils worldwide. Having a library of components that can be developed in an open source manner can open the door to building different types of apps with reduced cost, applications that can diversify the market for educational software and maybe can resolve some certain scenarios of utilization at a lower cost.

Improvement of quality

The quality of software in the case of CBSD depends on the quality of its individual components [22]. Considering that most of the components are developed in different teams and are used by different applications, the quality of them it is assured and validated by being on the market for a while and being tested in different configurations.

An application that has a good quality, it is an application that can easily gain reputation on the market and can easily become popular to the public. In the perspective of an "ed-app" having a good quality means being renowned and becoming more and more the first choice of teachers and pupils. This way the relevant resources can be easily visible, used worldwide assuring the software received by the market is a good one that can be reliable.

Challenges of CBSD

When talking about a solution it is always important to know from the start what challenges it can bring along the way. As shown in fig. 3 the most reemerged challenges from the analyzed studied are regarding finding the right components, updating and maintaining the components, ambiguous requirements and fixing problems.

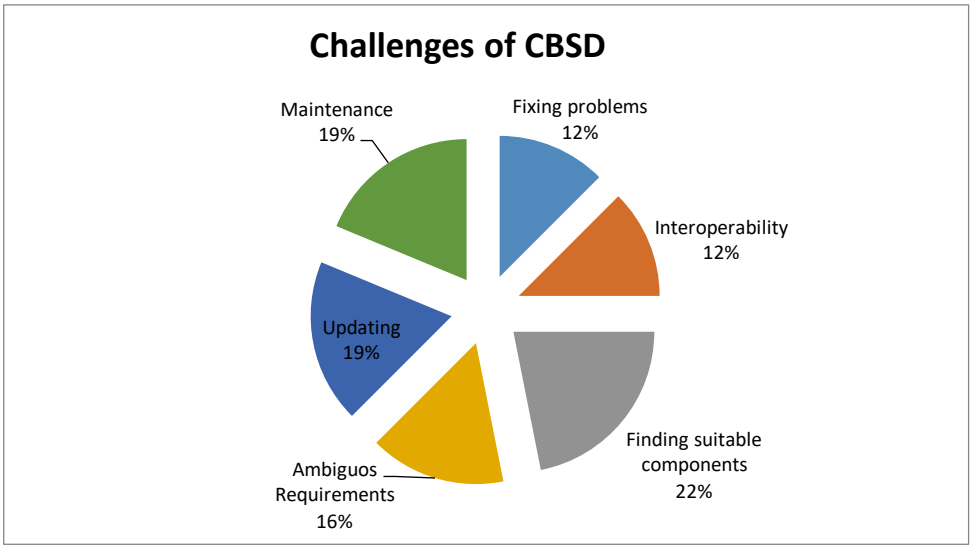

Fig. 3. Challenges of CBSD

Choice of components

When it comes to CBSD, the developer has the difficult task to identify and choose the proper components that can cover all the requirements of the product. As the market is limited, it is hard to find something that will cover the needs one hundred percent. As [18] mentioned before, this problem can be overpowered by picking the right components from the available pool of choices and trying to cover most of the requirements that were negotiated with the stakeholders.

It is clear that when it comes to component reusability, the generality, scalability and adaptability are really important and therefore the investment in it is more demanding.

In developing educational software, this challenge is even more overwhelming. The developer should have clearly in mind what he wants to obtain, what he is ready to bargain and what he can cover by his own. He also should have a metric to evaluate the components and choose the ones that can really bring a benefit to ease the process of learning.

Updating & Maintenance

When thinking of CBSD one must have in mind that while the overall maintenance costs can be less than a totally new software, the individual component maintenance price can be higher since the component must be ready to face different needs, different environments and even different types of customers with different expectations towards quality and support [14].

Moreover, when it comes to modifying the application, the developer either has to update the components to meet the new requirements; either has to find replacements for the existing parts, replacements that can fit the needed modifications. Sometimes, the components are developed by externals and the in-house knowledge is little which can lead to lots of problems. Also if the components are developed into a black-box method nothing can be done to adapt them to the necessities of the application.

Looking at this challenge from the perspective of educational software it is clear that this should be an important issue to watch out for because the educational system has little resources to deal with it and on the other side it is important to keep up with changes to still be relevant. This challenge can be overcome by choosing the right components from the start.

Ambiguous requirements

Managing, defining and refining requirements are really important actions in every development process and even more important in CBSD methodology. The CBSD components are developed with the purpose of reutilization, which means that they have to be packaged and reused in different types of applications. Therefore, after the requirements are collected and analyzed, the best path to a generalized the component has to be chosen [14]. So to be able to build components for helping

the educational software industry a good amount of resources should be invested in refining the needs and defining proper components, components that are based on findings like blended-learning is better than computer only learning and how different methods and paces can really help the learning process.

Fixing problems

When testing an application, the final aim is to know that the customer needs are covered without any problems and at an accepted level of quality. The lack of detail regarding the source code that appears in the case of external components makes tracking errors an actual problem. Furthermore, if some bugs are found in components with vendors unwilling to deliver fixes, the developers have a difficult time trying different workarounds to find fixes and sometimes the only solution is to change the entire component.

In the vision of developing an educational system based on components having bugs that cannot be fixes can really impact the public eye in a bad way. This challenge can be also overcome by choosing the right components, components from trustworthy vendors, components that are already used successfully and components where the developer has access to the code and can intervene in the case of an issue.

8 Conclusion

Nowadays the systems that are the base of this society like healthcare, legal, education, have to improve and become more relevant in this world that is in a continuous change, where technology is evolving at an exponential rate and where society is facing everyday new problems.

This paper is trying to concentrate its attention to the educational system and what problems he faces today in introducing technology as a primary aiding tool. The main focus is to create a clear view on how the CBSD methodology can be used and useful in helping the system step up to the public expectations, while also keeping an eye on the current problems such as lack of funds and teacher empowerment, lack of access to resources, curricula updates and personalized learning paths. Also, the newer information on how technology can be used in improving the students learning process is kept in mind in this review. To find out if the idea of having a library of components specialized on the needs and requirements of the educational systems could be a feasible solution different characteristic of CBSD where analyzed. From reducing development time to reducing costs, increasing quality and even increasing reusability, the advantages of CBSD cannot be overlooked when it comes to building the future of e-education. However, in this study challenges such as choice of components, difficulty in updating, maintenance, testing and ambiguous requirements are not omitted as they paint a clear vision of the real life and the obstacles that someone has to be considered when choosing this solution.

This research has managed to lay out the most important advantages that can be achieved by overcoming the challenges that derive from CBSD, both of which have been mentioned in the specialized literature. Based on this study one can conclude that CBSD is the right choice to make in order to improve the development of educational software considering all the problems this system faces and building a library of educational components with a strong community behind can be a suitable solution.

Acknowledgements

Parts of this paper have been presented at the 18th International Conference on Informatics in Economy (IE 2019), hosted by The Department of Economic Informatics and Cybernetics, Faculty of Cybernetics, Statistics and Economic Informatics from the Bucharest University of Economic Studies with the Romanian Association for Informatics in Economy Training Promotion – INFOREC and the Romanian Chapter of the Association for Information Systems, Bucharest, 30-31 May, 2019.

This paper was co-financed from the Human Capital Operational Program 2014-2020, project number POCU / 380/6/13/125245 no. 36482 / 23.05.2019 "Excellence in interdisciplinary PhD and post-PhD research, career alternatives through entrepreneurial initiative

(EXCIA)", coordinator The Bucharest University of Economic Studies".

References

[1] J.-P. N. America, "Education Technology Evidence Review," [Online]. Available: https://www.povertyactionlab.org/sites/default/files/documents/education-technology-evidence-review.pdf. [Accessed 29 August 2019].

[2] M. D. McIlroy, "Mass-produced software components," in Software Engineering Concepts and Techniques (1968 NATO Conference on Software Engineering), New York , Van Nostrand Reinhold, pp 138-155, 1976.

[3] C. Szyperski, Component Software: Beyond Object-Oriented Programming (2nd), Boston, MA, USA: Addison-Wesley Longman Publishing Co., 2002.

[4] M. Goulão, "CBSE: a Quantitative Approach," in PhD Workshop at ECOOP, Darmstadt, Germany, July, 2003.

[5] H. Washizaki, H. Yamamoto and Y. Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components," in Proceedings of the 9th International Symposium on Software Metrics, Sydney, Australia, September 2003 , pp 211-223.

[6] I. Sommerville, Software Engineering, 10th ed, Scotland: Pearson, 2016.

[7] N. S. Gill, "Reusability Issues in Component-based Development," ACM SIGSOFT SEN, vol. 28, no. 6, p. 30.

[8] T. Vale, I. Crnkovic, E. S. d. N. Almeida, P. A. d. M. Silveira, Y. C. Cavalcanti and S. R. d. L. Meira, "Twenty-eight years of component-based software engineering," Journal of Systems and Software, vol. 111, no. January 2016, pp. 128-148, 2016.

[9] A. I. Khan, N.-u.-. Qayyum and U. A. Khan, "An Improved Model for Component Based Software Development," Journal of Software Engineering, vol. 2, no. 4, pp. 138-146, 2012.

[10] T. Wijayasiriwardhane, R. Lai and K. Kang, " Effort estimation of component-based software development–a survey," IET software, vol. 5, no. 2, p. 216–228, 2011.

[11] N. J. Basha and D. S. A. Moiz, "Component Based Software Development: A State of Art," in IEEE-International Conference On Advances In Engineering, Science And Management, 2012.

[12] D. N. A. Jawawi, S. Sabil, R. Mamat, M. Z. M. Zaki, M. A. S. Talab, R. Mohamad, N. M. Hamdan and K. Kamal, Mobile Robots - Control Architectures, Bio-Interfacing, Navigation, Multi Robot Motion Planning and Operator Training - Chapter A Robotic Wheelchair Component-Based, InTechOpen, 2011.

[13] F. L. F. Almeida and C. M. Calistru, " Assessing Quality Issues in Component Based Software Development," International Journal of Advanced Research in Computer Science, vol. 2, no. 2, pp. 212-218, 2011.

[14] S. Singh, A. Singh, Samson and S. Singh, "Component Based Software Engineering," International Research Journal of Engineering and Technology, vol. 3, no. 5, pp. 448-454, 2016.

[15] T.-T. Pham, X. Défago and Q.-T. Huynh, "Reliability prediction for component-based software systems: Dealing with concurrent and propagating errors," Science of Computer Programming , vol. 97, p. 426–457, January 2015.

[16] P. RANA and R. SINGH, "COMPARATIVE ANALYSIS OF COHESION METRICS FOR COMPONENT BASED SOFTWARE SYSTEM," Journal of Theoretical and Applied Information Technology, vol. 96, no. 14, pp. 4369-4378, 2018.

[17] H. Yin and H. Hansson, "Fighting CPS Complexity by Component-Based Development of Multi-Mode Systems," Designs , vol. 2, no. 39, pp. 1-22, 2018.

[18] A. I. Khan, M. M. Alam, Noor-ul-Qayyum and U. A. Khan, "Empirical Study of an Improved Component Based Software Development Model using Expert Opinion Technique," International Journal of Information Technology and Computer Science, vol. 08, no. July 2013, pp. 1-14, 2013.

[19] N. Onofrei, "The verdict of education

experts on the Romanian education system: nepotism, underfinancing, inadequacy," 21 June 2016. [Online]. Available: https://adevarul.ro/educatie/scoala/verdictul-expertilor-educatie-sistemul-romanesc-invatamant-nepotism-subfinantare-inadecvare-1_576955105ab6550cb8299d92/index.html. [Accessed 4 March 2019].

[20] P. Gupta, "Innovation in Teacher Education," International Journal of Recent Research Aspects, Special Issue Conscientious Computing Technologies, pp. 964-966, April 2018.

[21] K. T. Boyes, "Why our education system needs to change," 4 September 2018. [Online]. Available: http://www.karentuiboyes.com/2018/09/why-our-education-system-needs-to-change/. [Accessed 4 March 2019].

[22] P. Banerjee and A. Sarkar, "Quality Evaluation of Component-Based Software: An Empirical Approach," I.J. Intelligent Systems and Applications,, vol. 12, no. December 2018, pp. 80-91, 2018.

Irina-Miruna RADU has graduated the Faculty of Cybernetics, Statistics and Economics Informatics in 2018. She holds a bachelor and a master degree in Economics Informatics and she is enrolled in PhD program. Currently she is a Java Developer and an Innovation Team member at Metro Systems Romania. She is interested in JAVA Programming and Business Development.