Students' Understanding of Computational Thinking with a Focus on Decomposition in Building Network Simulations

Steve Mvalo and Chris Bates
Department of Computing, Sheffield Hallam University, Faculty of Arts, Computing, Engineering and Sciences, City Campus, Sheffield, U.K.

Keywords: Computational Thinking, Simulation Software, Problem-solving Experiments.

Abstract: This paper reports a study into students' understanding of decomposition when building network simulations. Students were asked to complete three problem-solving tasks involving designing and troubleshooting computer networks using simulation software. Through online surveys, interviews and focus groups the students' understanding of computational thinking was interrogated. The results show that students were not conscious that they were applying computational thinking concepts when designing and troubleshooting networks on simulation software. It appears their interest were to simply get problems solved but not necessarily with the understanding of the application of the concepts of computational thinking.

1 INTRODUCTION

In this study we investigate students' understanding and application of one concept of computational thinking: decomposition. We examine how students apply the concept when building network simulations and how, in turn, those simulations facilitate students' ability to decompose a networking problem into a set of smaller tasks.

Decomposition is one of the core concepts of computational thinking. When using decomposition, problems are systematically broken into levels of abstraction that can be understood and solved more readily than can the original, complex problem. Computational thinking brings together a number of ideas about problem solving and algorithmic thinking in ways that can be readily applied to a wide variety of problems across diverse domains.

Section two introduces the idea of computational thinking, section three looks at how simulation tools can facilitate teaching computer networks, section four looks at experimental design, section five introduces the tasks that were set for the students and section six presents the results. We conclude by providing some emerging ideas and recommendations for further studies.

2 COMPUTATIONAL THINKING

Computational thinking is an approach to problem-solving which uses abstraction, decomposition, generalization and the creation of algorithms to identify solutions. The approach closely mirrors that which is used in software development and creates solutions that can be implemented relatively easily by people or by machines. Originally coined by Papert (1980), the term was popularized in (Wing, 2006) where the approach was applied to general problem-solving rather than being restricted to the domain of computer science. Wing (2006, 2008) describes computational thinking as involving problem-solving encompassing a set of mental tools, the design of systems and an understanding of human behaviour and that it represents a universally applicable attitude and skill set everyone, not just computer scientists, [could] learn and use. In 2011, Wing revised her definition of computational thinking as the "thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information processing agent" (p. 60).

Computational thinking helps students think algorithmically, define abstractions, decompose and reify them in their solutions, (Wing, 2011). The core

Mvalo, S. and Bates, C.
Students' Understanding of Computational Thinking with a Focus on Decomposition in Building Network Simulations.
DOI: 10.5220/0006693302450252
In *Proceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018)*, pages 245-252
ISBN: 978-989-758-291-2
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

concepts of computational thinking are abstraction; algorithmic thinking; problem solving; pattern recognition (generalization); design-based thinking; conceptualising; decomposition; automation; analysis; testing and debugging; mathematical reasoning; implementing solutions; modelling, (Grover & Pea, 2013; Kalelioglu et al., 2016).

This study uses a working definition of computational thinking as:

Those thought processes that apply fundamental ideas and approaches from computer science including, but not limited to, algorithms, abstraction, decomposition and generalization to solving technological problems such as the design of computer networks.

Within the broad discipline of computer science computational thinking can easily become entwined with the use of specific tools such as those used for software design because when using such tools the ideas and practices of abstraction, decomposition, generalization and so on become explicit to students. Wing (2008) is clear that tools should not get in the way of understanding and applying the concepts behind computational thinking but, rather, should reinforce and facilitate them. It is not sufficient that a learner be adept in the tool, they must become adept in using the tool to produce abstractions and concrete implementations from those abstractions.

3 DECOMPOSITION

The intellectual skill of decomposition is the ability to breakdown complex problems to a level such that it can be understood, solved, developed or evaluated (Csizmadia et al., 2015). Decomposition can involve looking at similarities within, and patterns of, the constituent parts of the problem. In so doing they become easier to understand and work with. The ability to identify these similarities and patterns depends on one’s previous knowledge, experiences and skills (Bocconi, et al., 2016). Decomposing a problem is one thing but solving the problem is another matter, albeit one that also requires prior knowledge, experiences and skills.

Wing (2008) defines decomposition within Computational Thinking as the process of unwrapping an abstraction of a complex problem into a concrete solution. Once students have solved a complex problem, they should be able to describe both how they identified the problem and the strategies they used.

4 SIMULATION TOOLS IN TEACHING COMPUTER NETWORK DESIGN

Simulation software provides a platform on which students can design, build and test networks that vary in complexity from trivial to complex simulations of the infrastructure of multi-national companies. Using such software students are able to work with systems that are far too complex for them to be able to build in real-life and to include technologies that they would otherwise not meet at University.

Teaching students to build even relatively simple networks that include a couple of routers and a few VLANs can require racks of dedicated hardware. Typical university class sizes mean that significant quantities of specialized equipment must be available to the students both within formal teaching sessions and when they undertake their own project and assessed work. This hardware is not intended by its manufacturers for a classroom setting. It has the robustness necessary to run almost indefinitely in the controlled environment of a network server room but is less able to withstand the rigour of constant re-cabling or power cycling or, indeed, of operating in warm, dusty classrooms.

Simulation software provides an excellent alternative to physical infrastructure when teaching computer networking (Janitor et al., 2010). Network simulations provide feature-rich, flexible platforms with a range of devices and software that is far greater than would be possible when using physical devices (Ruiz-Martinez et al., 2013). When using network simulations students have the flexibility to work on their network designs away from the classroom, (Zhang et al., 2012).

Many studies including those of (Galan, Fernandez, Fuertes, Gomez, & de Vergara, 2009; Hwang et al., 2014; Ruiz-Martinez et al., 2013) have shown that simulation software provides a highly realistic way of teaching computer networks, conducting research and experiment in designing complex network systems. And because simulations are inherently flexible, extensible and highly configurable, students are able to use them to design network topologies that range from simple to highly complex. The inherent plasticity of a good simulation tool means that it provides a platform upon which students can build almost any structure and, in so doing, extend their inventiveness and innovation (Ruiz-Martinez et al., 2013).

5 DATA COLLECTION METHODS

This study used mixed methods with dominant qualitative approaches. Initially two separate surveys comprising 69 students and 14 lecturers respectively were conducted to discover participants' understanding of computational thinking and of the use of simulation software in network design. Seven undergraduate students studying for computer networks were sampled randomly for a focus group interviews. In this focus group, students were queried about their understanding of computational thinking and their experiences of using simulation software to design networks in their day-to-day lab activities. A small group of postgraduate students undertook three consecutive problem-solving tasks and followed-up with one-to-one and focus group interviews. The postgraduate students were taught by one of the authors and the three tasks formed part of their assessment.

The students were given three different problems and six weeks to complete each. In total the students were monitored for almost six months as they designed and built simulations and undertook troubleshooting of their simulations. On completing each problem, the students recorded videos in which they demonstrated their problem-solving approach, showed their solutions and reflected on their learning through the task. In the focus group the postgraduate students were asked to reflect upon their understanding of the problems and how they had solved during the practical tasks.

All participants have been anonymized to preserve their confidentiality. Undergraduate and postgraduate students in this study have been referred to as UGx and PGx respectively, where x represents a random number. Lecturers have been referred to as LecX where X represents a random number too.

In the focus groups the students were prompted to explain their understanding of the differences between computational thinking and critical thinking. Students were asked about the strategies that they used when designing complex network simulations to try and reveal whether, and how, they might apply the core concepts of computational thinking. Students were further asked to explain their previous experiences in using simulation software against physical hardware and finally were asked to explain their general recommendations on the use of simulation software in developing their understanding of computational thinking.

The focus groups investigated the students' perceptions and reflections on the use of simulations in network problem solving. These students were also asked their understanding of computational thinking and their perceptions on the use of simulation software in developing their computational thinking. The intention was to investigate their ability to apply any of the core concepts of computational thinking and to further drill down into their use of decomposition in building network simulation.

To triangulate the findings of the online survey and focus groups, three lecturers participated on one-to-one interviews in which they talked about their own understanding of computational thinking with a particular interest in decomposing network abstraction when building network simulation. Lecturers further talked about how they apply the core concepts of computational thinking in their own teaching practice.

6 THE PROBLEM-SOLVING TASKS

The students undertook series of increasing complexity tasks across six months during laboratory sessions for three of their modules. They recorded themselves using the Screencastomatic software desktop capture program, https://screencast-o-matic.com/. Because the students were asked to record all of their activities in each lab session we were able to see exactly how they used the simulation software including mistakes, dead-ends and failed approaches. Desktop capture differs from other approaches because it is unobtrusive - these students tended to forget that it was recording them - and so gives the researcher a raw and unfiltered view of the activity.

When capturing their sessions, the students were asked to outline the task as they understood it, show themselves solving the task, demonstrate and discuss the strategies they used, and demonstrate their working solutions. On completing each lab task, the students were questioned about their thinking as they solved the problem.

6.1 Task One

Students had to reverse engineer an enterprise network from a list of routing tables. Routers advertise those networks to which they connect directly and share those networks advertised by their neighbours. The set of routes gives the topology of the enterprise network. In reverse engineering the networks which are advertised are traced back so that the topology of the entire network can be re-built.

In this first task the students had to reverse engineer the enterprise network infrastructure, troubleshoot design problems that were embedded in the routing tables, and implement appropriate solutions. To verify their designs, students had to show their routing tables matched those given in the task description and corrected its embedded errors.

6.2 Task Two

In the second task the students had to design from scratch an enterprise network infrastructure as shown in Figure 1 with sites dispersed across five cities. Specific requirements covering: the provision of bandwidth; throughput; response time; access by users to appropriate resources; confidentiality; and system integrity. The students were told to use IP addressing schemes that involved IPv4 and IPv6 and to choose suitable routing protocols to facilitate communication across the network.

This was a challenging task for these students because it built on their priori knowledge and skills in LAN design and implementation to produce a larger, more functional network infrastructure incorporating WANs. At the time that they undertook the task, the students were still becoming familiar with many aspects of networking technology.

6.3 Task Three

The third task was a set of activities that combined the design of LANs and WANs to implement a secure enterprise infrastructure. The key learning points for the students were the incorporation of security into otherwise familiar network infrastructure.

6.4 Task Four

The final task having attempted all three of the networking tasks, the students were asked to write an individual reflective report covering all three tasks. They had to discuss their thought processes and the strategies that they followed in creating their solutions and recommendations. This task was an important part of the assessment that the students were undertaking. For the researchers these reports had the benefit that the students' recollections and memories could be compared with the video evidence to show whether they had done the things as they thought they had.

7 RESULTS

Simulations are not just about designing and making complex systems, they also allow students to work with complex ideas. The flexibility and usability of simulations mean that students can be encouraged to do more testing and thus be able to critically evaluate their own work. This was alluded to by a

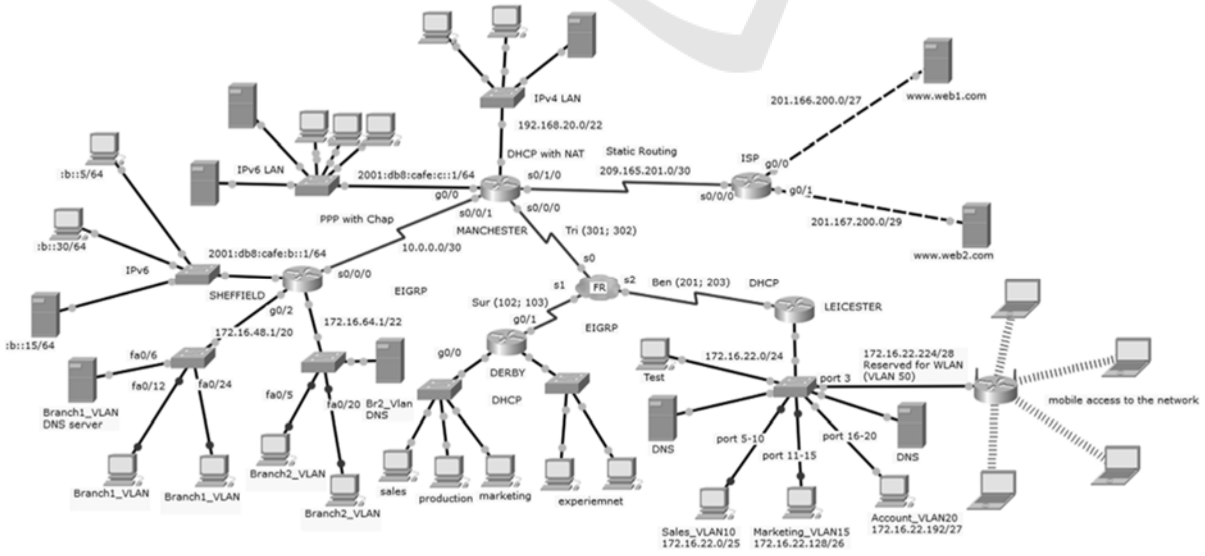

Figure 1: Sample problem-solving task.

number of subjects in the focus groups:

> *I think I will however, test more on simulation software than on real kit. You would therefore apply problem solving on simulation software with more critical because you know you are not doing it on the real thing (UG6)*

> *It's more efficient and you don't waste time when configuring on simulation software; as you have more and more ideas you can apply and implement them as you wish hence developing your computational thinking much better (UG4)*

As their learning progressed the students learned to decompose the complexity of their network topologies and the security requirements built into the problems into small, solvable tasks. This is what students had to say in their reflective reports:

> *The topology was designed in such a way that each branch is separate and can be easily evaluated. The idea is to break down the network structure to be less complex when sorting out issues, through computational thinking, it makes it easier to isolate the problem and solve it in bits (PG1)*

> *Breaking the task into smaller tasks and concentrating on the main task helped me a lot in solving network problems such as when designing a WAN, the whole design can be broken down to smaller task that is into LANs and the LANs can be breakdown into smaller branches like creating small networks and combining them as a LAN (PG4)*

By building the simulations the students were able to view the entire enterprise network and identify areas of vulnerability, loopholes, bottlenecks and threats. Once they were able to visualize and experiment with problems within the network, the students could begin to develop their ideas about overcoming them and so secure the system. In their reflections the students noted that decomposing systems within the simulation meant that they could better appreciate the abstraction and operation of routing tables. This is something that has been shown to be difficult to achieve when using physical hardware (Janitor et al., 2010). These are some of their comments:

> *Depending on the level of complexity working on simulation was much more appreciated […] it was less stressful to work on complex design than real set [hardware] (PG2)*

> *when you are analysing a network it is pretty much easier to analyse it through Packet tracer. It is easy to see things which need to be seen. You can easily break down problems. Packet tracer is user-friendly as a software and so it is easy to apply critical thinking (PG1)*

Through the simulation's visual representation of a network the students were able to work at differing levels of abstraction. They could think about hardware, applications or routing tables as necessary, focusing on important details as they produced the final concrete design. The topology in Figure 2 shows a partial output from one of the students after working out a reverse engineering problem-solving task.

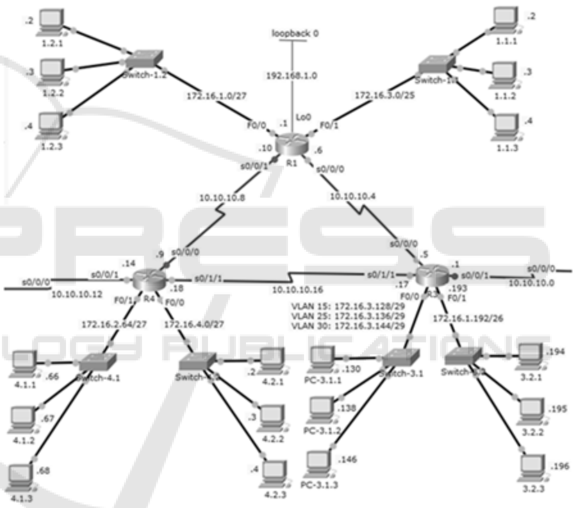

Figure 2: Student partial topology.

After creating a visual representation of the enterprise network, students were able to solve problems that were inherent in the routing table that they were given. The students managed to breakdown problems for each router and its switches to create a correct routing table.

Having learned to break problems down to their constituent parts, the students were asked to demonstrate whether they could generalize solutions from specific instances. In computational thinking, the concept of generalization is extended from the concept of decomposition (Bocconi et al., 2016). Once students have broken the problem down and begin solving them they must apply their prior knowledge, experience and skills to identify

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

patterns, similarities and commonalities to come up with their optimum solutions.

Students' reflective reports showed that they were often able to identify similarities and commonalities from their previous knowledge and experience but they were not aware that they were applying computational thinking skills to the problems. They were not able to demonstrate how they identified patterns in solving tasks that they could go on to apply to other tasks. This is what one of the students commented:

I think the main problem is your knowledge in solving the problem, because in as much as you may be able to break down the chunk of a bigger problem into small manageable problems but if you don't know how to solve all those small problems, it still remains a problem. So your knowledge to the problem you are solving is significant. […] Background knowledge helps in understanding the similarities and differences which will help in making appropriate decision in solving that problem (UG6)

Some of the lecturers who were surveyed as part of the work said that their students were interested in making sure that the problems were solved but not in how they were doing so. It became clear through the study that students were solving problems through a routine of troubleshooting, configuring and fine-tuning.

These were some of the comments students made in their reflective report which were not clearly demonstrated:

Sometimes viewing the case via a general point of view can be useful to find out possible solutions as it helped me to recognise the general similarities and differences in the whole scenario so that I could apply the same solution for the similar parts of the case. For example, in WAN assignment I found out that some LANs followed the similar patterns so I applied the same configuration for each of them based on my previous knowledge in configuring LAN Student (PG6)

Analyzing similar patterns (network requirements, when defined the role of each branch, specifically we had a plan of setting up similar configuration on different branches, such as where it was asking to provide NAT on LEICESTER and DERBY we had a same requirement, ACLs on VLANs) (PG2)

The students were not taught an explicit approach to problem solving and the strategies that they developed did not necessarily map onto a computational thinking approach. The students' design approach was not based on their understanding and application of computational thinking but was one of simply making sure through trial and error that their designs were operational.

This was a lot of trial and error for me. I found it most difficult to find how to use the redistribute command correctly. I had to use online resources to figure out a solution. I am still studying up on this so I may not have used it in the exactly correct way, but it did produce an output that appears to match [routing output] (PG3)

This concurred with what one of the lecturers commented:

I expect students will largely use trial and error in the beginning until they understand the problem. If students knew how to do computational thinking (or indeed any structured approach to thinking) they would be more organised. I guess we have to teach them that (Lec7)

This point was well encapsulated by one of undergraduate students during focus group interviews who alluded to the nature of their course as being one that gave practical skills rather than teaching a way of thinking about problems and systems. This suits these students who are focused on getting into employment on graduation. They are more interested in gaining good practical skills that will immediately help them to get employment in their field of study than in developing those higher-level analytical skills that may be used to build a career. This student thinks computational thinking is just "an academic buzz word":

I don't feel our course teaches us any computational thinking, I feel our course is design to incooperate workforce processes. The course is designed to introduce workplace processes, best practices from hardware, best practices from enterprises. It is designed ready to get you in the workplace; it gets you understand how the mind of everyone who works in the industry works; so I can jump into my job and design my network based on CISCO-based practice or Juniper-based practice, so you don't necessarily approach it from a computational thinking point of view. Computational thinking is kind of like an academic buzz word and not a *real while in deploying enterprise architects (UG4)*

The lecturers indicated that they do not focus on computational thinking. Their focus is to test all of the concepts that they have taught in class by making sure that students are able to demonstrate and apply them in practical tasks. One lecturer said that he does not influence his students' choice of the methods that they adopt when designing solutions because he believes that every student has his or her own best way of solving problems. These are some of the comments which other lecturers made:

I simply give them an assessment that test all the points of knowledge they should have and not necessarily from the computational thinking point of view. I look at can they implement it, can they look at why am I doing this, […] But I have never thought on how do I create an assessment from a computational point of view […] may be its some of the things we should be thinking about (Lec2)

I want to see that students can demonstrate that they can apply what is it they have learnt to produce a viable solution, for example, and be able to critically evaluate that solution that they come up with – so am not thinking down the levels of how would they break down the problem and how would they solve each element or how do they choose a protocol so which in effective is the algorithm path […], Or abstracting by saying this protocol functions like this and that – so that's not how am thinking about it when I am designing or assessing students (Lec3)

8 ANALYSIS

From the online survey, one-to-one interviews and focus groups it became clear that neither the students nor lecturers in this study were aware what computational thinking is. Several of them indicated that they had to use internet searches to understand what the term computational thinking means. It was, therefore, not surprising that lecturers said that they are neither conscious of, nor focus on, computational thinking when teaching and assessing students.

The results have shown that the use of simulation software in designing computer networks helps students breakdown complex problems into smaller, manageable tasks. This is decomposition. Simulation software allows visual representation (Janitor et al., 2010) of the enterprise network infrastructure. The students found it easier to understand the abstract concepts of network design once they had this representation. Working back from their abstractions students were able to produce a new functional network infrastructure. The ability of simulation software to provide visual representation of their entire design let students focus more closely on the problem (Zhang et al., 2012) so that new ideas emerged when solving problems. These results are consistent with Galan et al., (2009); Hwang et al., (2014); and Ruiz-Martinez et al., (2013).

Students reported that they were able to identify the security vulnerabilities and inherent errors in the design they were given, and hence, work to solve those problems. However, students found that they were unable to apply some solutions because of limitations with the software. Expósito, Trujillo and Gamess, (2010) reported that simulation software, particularly Packet tracer, has limitations compared to physical devices in that some commands cannot be applied.

It became clear that although students were able to explain in their reflective report how they identified patterns, similarities and commonalities in problems, they were not aware that in doing so they were applying the concept of computational thinking.

The results show that participants in this study have little understanding and application of computational thinking. The results show that students consistently applied one aspect of computational thinking: decomposition. At Sheffield Hallam University teaching and learning in the area of networking is skills-based which may explain why the students are able to decompose problems.

Focus-group responses show that stuednts think that computational thinking helps them understand abstract concepts and produce concrete solutions. During their demonstrations of their problem-solving tasks they were not clear how they applied computational thinking. Their interest was to solve problems in any way that worked, including through trial and error. This observation is in line with the comments from some of the lecturers when asked about the strategies they use when teaching and assessing students when designing networks.

9 CONCLUSION

This study has demonstrated that students are able to apply the ideas that together form computational thinking even when they have not formally been taught those ideas. The students who participated in this study were able to break complex problems into smaller sub-problems, build solutions to those sub-problems and compose them into simulations that solved the whole problem. Teaching staff who said that they were more engaged with the use of technology than with approaches to problem-solving were actually giving their students the types of advanced thinking skill that is usually included in a definition of computational thinking. The study shows that simulation software is a very important tool in the teaching of network design. It provides visual representations of computer networks that can be manipulated at different levels. By manipulating the levels of abstraction in their simulations, students are able to decompose problems. This helps them to develop their understanding of both problems and solutions. The use of simulations within networking courses is recommended because not only are students able to solve the immediate problems that they face, their use of the software improves their ability to apply some of the principles of computational thinking. It is also recommendable that lecturers familarise themselves with the concepts of computational thnking so that they are able to consciously teach and assess students when designing simulation networks. Further work is needed to investigate whether, and how, other aspects of computational thinking may be developed implicitly through this and other aspects of education in computer networking.

REFERENCES

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). Developing computational thinking in compulsory education – Implications for policy and practice; EUR 28295 EN; doi: 10.2791/792158

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking, a guide for teachers. Computing at school, *Digital Schoolhouse*. Available from http://computingatschool.org.uk/computational thinking [access on 02/03/2016]

Expósito, J., Trujillo, V., & Gamess, E. (2010). Using visual educational tools for the teaching and learning of EIGRP. In *Proceedings of the World Congress on Engineering and Computer Science (Vol. 1).*

Galán, F., Fernández, D., Fuertes, W., Gómez, M., & de Vergara, J. E. L. (2009). Scenario-based virtual network infrastructure management in research and educational testbeds with VNUML. *Annals of telecommunications-annales des télécommunications, 64(5-6), 305-323.*

Grover, S., & Pea, R. (2013). Computational thinking in K–12. A review of the state of the field. *Educational Researcher, 42(1), 38-43.*

Hwang, W.Y., Kongcharoen, C., & Ghinea, G. (2014). To enhance collaborative learning and practice network knowledge with a virtualization laboratory and online synchronous discussion. *The International Review of Research in Open and Distributed Learning, 15(4), 113-137.*

Janitor, J., Jakab, F., & Kniewald, K. (2010). Visual learning tools for teaching/learning computer networks: Cisco networking academy and packet tracer. In *Networking and Services (ICNS), Sixth International Conference on IEEE, 351-355.*

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing, 4(3), 583–596.*

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. *New York, NY: BasicBooks.*

Ruiz-Martinez, A., Pereniguez-Garcia, F., Marin-Lopez, R., Ruiz-Martínez, P.M., & Skarmeta-Gomez, A.F. (2013). Teaching advanced concepts in computer networks: Vnuml-um virtualization tool. Learning Technologies, *IEEE Transactions, 6 (1), 85-96.*

Wing, J. M. (2011). Computational thinking. In VL/HCC (p.3). Available from: https://csta.acm.org/Curriculum/sub/CurrFiles/WingCTPrez.pdf [access on 12/02/2016]

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366(1881), 3717-3725.*

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49(3), 33-35.*

Zhang, Y., Liang, R., & Ma, H. (2012). Teaching innovation in computer network course for undergraduate students with packet tracer. *IERI Procedia, 2, 504-510.*