University of Reading

Developing Computer Networks Students' Computational Thinking: The Case for the use of simulation software

Thesis submitted for the Degree of Doctor of Education

Institute of Education

Steve Mvalo

September 2018

Declaration of original authorship

I confirm that this is my own and use of all material from other sources has been properly and fully acknowledged

Abstract

The current study set out to investigate students' and lecturers' understanding of computational thinking (CT) and their perceptions of how students' CT skills could be facilitated through the use of simulation software. This is an important research area as studies have shown that developing Computer Sciences students' CT skills could enable them to become producers of technology in the 21st century. Mixed methodologies with qualitative methods were espoused in addressing the research questions. Data were collected from a UK university via surveys (69 students and 14 lecturers), interviews (four students and three lecturers), two focus groups with seven undergraduate students and six postgraduate students, problem-solving tasks using simulation software, and reflective reports from six postgraduate students. Using thematic qualitative data analysis, findings of the first research question indicated that students' and lecturers' understanding of CT went beyond the predetermined themes of the common conceptualisation of CT i.e. *abstraction, decomposition* and *generalisation*, and that it encompasses other dimensions of CT e.g. *algorithmic thinking* and *problem-solving approaches*. However, students' and lecturers' overall understanding of CT was predominantly based on the concept of decomposition and lecturers were not conscious of the concepts of CT when teaching and assessing students, indicating lack of their understanding in teaching and assessing students' CT skills. In the second research question, students' and lecturers' perceptions indicated that simulation software provides a conducive platform to facilitate students' CT skills. Finally, in the third research question, students were able to demonstrate the concepts of abstraction, decomposition and generalisation using problem solving tasks via simulation software though subconsciously. From this study it is recommendable that simulation software be used to facilitate the application and development of students' CT skills.

Acknowledgements

First and foremost, I give praise and honour to my God for taking me through this exciting but challenging task. I can do nothing without you, my Lord. I salute you, Lord.

I would like to extend my sincere gratitude and appreciation to some of the significant people who have given me a hand in the success of this project. First and foremost, I would like to thank my supervisors, Dr Natthapoj Vincent Trakulphadetkrai and Dr Yota Dimitriadi. You have been instrumental to my project. I really appreciate your timely, comprehensive, constructive and thought-provoking feedback which have massively shaped my way of thinking throughout this thesis. I cannot complain having you as my supervisors. I have no proper word to express my gratitude, but I can only say, thank you! and God Bless you.

Furthermore, I would like to thank the leadership and management of Sheffield Hallam University, particularly Prof. Tony Clark for endorsing the sponsorship of this study for the last three years. Additionally, I would like to appreciate all the students and lecturers who participated in this study - without you, the project would not have been possible. Thank you for your time. In a special way, I would like to thank my colleague, Dr Chris Bates who has always been available when I needed someone to debate and critique my thoughts. A true colleague and a true friend indeed. Thank you so much, Chris. Thank you, Mark Townley, for the support you gave me too. I really appreciate.

I would also like to thank Foster and Esnat for taking care of me all the time I visited the University of Reading. Thank you, Peace, Glory and Grace, for constantly praying for me. I own you a lot.

Special thanks to my wife, Jean, and our beautiful princesses, Uchizi and Kyrah to whom this thesis is dedicated. You always assured me that ***I can do it*** and indeed I have come to the end of the tunnel where I can prove you right. Jean, you took much of your time looking after Kyrah while I was busy doing the writing up. Thank you for your patience and encouragement. Uchizi, what can I say? You have been my push to the success of this thesis. You have spoken words of encouragement that I have wondered whether it was you or the angel. Kyrah, there have been so many times that you could just climb at my back seeking my attention while I was busy writing up. You have endured a lot without my support. I own you a million. I therefore dedicate this thesis to you three.

Table of Contents

CHAPTER 1: INTRODUCTION

1.1 Technology in the 21st century

Technology has become inevitable in almost every essential daily need, business and education. Almost everything is either technologically influenced or operates on a technological environment. This requires educating Computer Sciences (CS) students to critically understand the operation of technological systems (Angeli et al., 2016), but most importantly to enable them to design systems and solve problems when such systems malfunction (Czerkawski, 2015; Wing, 2006, 2008). Therefore, many scholars (e.g., Angeli et al., 2016; Fluck et al., 2016; Goode, Chapman, & Margolis, 2012) argue that the integration of computer sciences in education facilitates CS students' higher order thinking skills such that they can become the producers of technology in the 21st century.

In education, it is not only the use of technology in teaching and learning that matters, but how that technology enhances students' learning skills (Wing, 2008). For example, Ertmer et al. (2012), quoting McCain (2005), argues that it is not the use of technology in classroom that is an issue, but the ability to develop thinking skills in learners through the use of technology to solve problems. Technology in teaching and learning should drive CS students to become more imaginative, innovative and creative (Wing, 2006, 2008) in designing and engineering systems and constructing network infrastructures that are relevant and adaptable to the technologically-driven society (Angeli et al., 2016; Harris, Jones, & Baba S., 2013; Wing, 2008).

CS lecturers require a great deal of nurturing learners who can develop higher order thinking skills (HOTS) which according to Wing (2008) argues that HOTS share with mathematical thinking (i.e. *in problem solving*); engineering thinking (i.e. *approaches to designing and evaluating complex systems*); and scientific thinking (i.e. *approaches to computability,*

intelligence and human behaviour). Henceforth, CS scholars (e.g., Angeli et al., 2016; Barr & Stephenson, 2011; Grover & Pea, 2013; Wing, 2014) are advocating the concepts of *computational thinking* (CT) via problem-solving skills, approaches to designing and evaluating complex systems and understanding the intelligence of computation and human behaviour in teaching CS students.

1.2 **Computational thinking**

Currently, there are several definitions of computational thinking (CT) though without a consensus one. Some scholars (e.g. Wing, 2011) argue that CT is "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), others (e.g. Aho, 2012) argue that CT is "the thought processes involved in formulating problems, so their solutions can be represented as computational steps and algorithms" (p. 832). Many other scholars (e.g., Barr & Stephenson, 2011; Hambrusch, Hottmannn, Korb, Haugan, & Hosking, 2009; Kalelioglu, Gulbahar, & Kukul, 2016) argue that CT is a problem-solving concept applied in computing field that includes, but not limited to, formulating problems, representing data through abstraction, logical analysis and organisation of data, automation through algorithmic thinking. Despite the differences in the definition of CT, there are some common core concepts associated with CT (e.g. abstraction, decomposition and generalisation) which are fully discussed in the Literature Review chapter.

1.3 **Advocacy and background of computational thinking**

For the past 12 years, Wing (2006) has been advocating the use of CT across all disciplines. She cites disciplines such as, Sports, Biology, Engineering, Computer Science, Education, Law and Business that they all need CT in their day-to-day operation (Wing, 2011). Although Wing seems to be the current advocator of computational thinking, CT can be traced further back to

1950s when it was referred to as algorithmic thinking (Tedre & Denning, 2016). Tedre and Denning (2016) argue that Papert appears to be the first person in 1980 to use the phrase *computational thinking* in knowledge construction using computers and LOGO language.

The LOGO is the computer language in which children play with objects (e.g. Turtle) to create their own *microworld* (i.e. their own imaginary world) (Papert, 1993). Children use computer commands to move the object(s) to and from their working platform. Children can also create objects (e.g. squares, triangles, etc.) to expand their imagination, by drawing a house using computers. Therefore, children create their own microworld based on the interactions of objects, computers and commands issued on the computers. Papert's argument was that when children interact and work with computers in that manner, they enable them to develop their critical thinking skills through programming (Papert, 1980, 1991).

However, there are some commonalities in Papert's concept drawn from LOGO and those defined by Wing (2006). For instance, in their definition of CT, both Papert and Wing emphasize on reasoning, thought processes, analysing, exploring and formulating problems (Mannila et al., 2014). However, the difference is that while Papert's focus was on the use of programming in developing children's thinking skills, Wing's focus is on general concepts of CT drawn from computer science and applied across multidiscipline field (Mannila et al., 2014; Voogt, Fisser, Good, Mishra, & Yadav, 2015; Wing, 2006). Wing's (2006) advocacy for the intersection of computer science to other discipline triggered off more debate (Czerkawski, 2015) and formulated more research questions such as how CT might be taught and assessed (Selby & Woollard, 2014); at what schooling level CT should be focused (Wing, 2008), and how CT can be integrated across multiple discipline (Angeli et al., 2016; Mannila et al., 2014). In an attempt to answering some of these questions, research (e.g., Barr & Stephenson, 2011; Grover & Pea, 2013; Lu & Fletcher, 2009; Wing, 2008) has largely focused at primary and

secondary school, while at university level, research (e.g. Berland & Lee, 2011; Selby, 2015; Kazimoglu, Kiernan, Bacon & Mackinnon, 2010; Repenning, Basawapatna & Escherle, 2016) has focused on students studying for Programming and Games courses more than Computer Networks courses.

1.4 Context of the research and my role in the case-study institution

I joined the institution in 2015 as a Lecturer and then progressed to the role of a Course Leader and Senior Lecturer in 2016. Therefore, apart from being a Senior Lecturer in Computer Networks course I am responsible for leading and managing the Computer Networks course. Following on my leadership and management responsibilities over Computer Networks course, I observed some limitations of how students' skills are underdeveloped on the course. For instance, the current curriculum focuses on teaching students to repair, maintain, support existing technologies without further examining into how students can develop their own network design systems. Currently students have limited skills to be the producers of technology but are rather the consumers and/or engineers of the existing technologies. Besides, students do not have a structured-thinking approach to problem solving; they dominantly use trial-and-error approaches to problem solving. However, the application of CT skills may help them to have a structured-thinking approach to problem solving which would not only help them in designing and troubleshooting network design systems but enable them to be the producers rather than the consumers of technology. I acknowledge the biases and challenges which may be associated with myself as an inside researcher (Floyd & Arthur, 2012), such as misleading assumptions and using the knowledge and experiences I have within the institution. However, my colleagues were more open to give me some examples which they thought I was familiar with. For instance, they would use expression such as "*as you know*". Furthermore, as an inside researcher and course leader, students as well as my colleagues were very positive in

support of the project with the hope that I would positively influence change to the curriculum. Additionally, the department of computing was going through the phase of developing and strengthening its research area in teaching and innovation to become an outstanding applied university. This has been fully discussed in Section 3.5 of the Methodology chapter.

1.5 **Why computer networks course?**

Computer Networks course was chosen for the investigation of CT in this study because it is an area where I am familiar with. Also, for my own personal professional development and building on concrete and evident good practice, I opted to pursue this study on Computer Networks course. Additionally, computer networking in computing discipline is inevitable (Janitor, Jakab, & Kniewald, 2010; Zhang, Liang, & Ma, 2012). For instance, programmers depend on computer networking to populate, share and run their programs; forensic computing requires networking to cover a wider geographical scope for data investigation; network security requires network infrastructure to provide a wider protective mechanism for an organisation; businesses, require computer networking to advertise their services and products across the world. Therefore, computer networking forms a fundamental base for computer science learners; ultimately this means the knowledge gained from this study can be easily transferable to other areas of computing. Furthermore, literature has shown very little evidence at university level on how CT skills have been taught and applied to students pursing Computer Networks in the discipline of Computer Sciences. However, there is substantial evidence of CT skills taught and applied based on the concepts of programming and games at primary and secondary school levels.

1.6 **Use of simulation software to facilitate computational thinking**

Simulation sotware, according to many studies (e.g., Dobrilovic & Odadzic, 2006; Hwang, Kongcharoen, & Ghinea, 2014; Ruiz-Martinez, Pereniguez-Garcia, Marin-Lopez, Ruiz-

Martínez, & Skarmeta-Gomez, 2013) offers a test-bed platform that is conducive for students to design, build, modify, test and redesign simulated networks which can vary from simple to complex network infrastructure. Using such software, students can design complex network systems which they are not able to design on real physical devices due to limited resources, inflexibility and rigidity of hardware, and costing factors (Galan, Fernandez, Ruiz, Walid, & De Miguel, 2004; Ruiz-Martinez et al., 2013). For instance, teaching students to build a simple network that requires a couple of routers, switches and firewall devices can require racks of dedicated hardware. However, with simulation software, students only need basic training on how to drag and drop devices on a simulated working platform (Janitor et al., 2010; Zhang et al., 2012). Furthermore, via simulation software, students can design unlimited network system scenarios within their reach and capabilities, enabling their creativity, imagination and innovation (Dobrilovic & Odadzic, 2006; Wing, 2006; Zhang et al., 2012). Drawing on many studies (e.g., Angeli et al., 2016; Barr & Stephenson, 2011; Kalelioglu et al., 2016; Selby, 2015; Wing, 2014) which show that CT involves problem-solving skills, in this study, simulation software was used as a platform for problem-solving tasks. Using problem-solving tasks students demonstrated the concepts of abstraction, decomposition and generalisation. There is a broader discussion on this that has been fully discussed in the Literature Review chapter.

1.7 **Focus of the study**

Drawing on from literature, there is no straightforward evidence on how CT is taught in universities particularly to students studying for Computer Networks course. This study is thus focusing on investigating Computer Networks students' and lecturers' understanding of CT; their perceptions of using simulation software to facilitate students' CT skills, and how the use of simulation software might facilitate students' CT skills. The three core concepts of CT (abstraction, decomposition and generalisation) were chosen in this study following the fact

that when students design and troubleshoot computer network design problems, they need to understand the abstracts of computer networks and turn those abstraction into concrete (Yalcin, Altun & Kose, 2015; Wing, 2006); through the concepts of decomposition i.e. breaking down complex network designs into small manageable solutions (Lee et al., 2011; Selby, 2015), students can work on problems at their suitable level of understanding and solve problems (Angeli et al., 2016). Finally, when designing and troubleshooting network designs, through the concepts of generalisation they can identify differences and similarities of network devices, protocols, and can use their prior knowledge and experiences in problem-solving (Csizmadia et al., 2015). This section has been fully discussed in the Literature Review chapter.

1.8 **Significance of the study**

Besides my own personal development in teaching practice as a Lecturer in Computer Networks, the current study also provides solid and evidence-based foundation for sharing good practice in how CT can be taught and applied to students pursuing a career in computer networks. Currently, there is no straightforward evidence in the area of computer networks how CT is taught and applied though there are some evidences in the areas of games and programming (Berland & Lee, 2011; Kazimoglu et al., 2010). The study will bring more opportunities to share some good practices via publications, workshops and conferences to fellow lecturers on how to develop CT to Computer Networks students when designing computer networks on simulation software. Not only will this help lecturers and students develop CT skills, but hopefully the study will also guide software developers and other researchers to further research and develop programs or models which can facilitate the application of CT across disciplines as Wing (2011) argues. Again, hopefully the results of this study will also trigger more debate and interest to other researchers to investigate how other core concepts of CT not investigated in this study may be taught and applied in Computer

Networks courses. Therefore, the findings of this study will help the author and others in the field of Computer Sciences to foster CT in CS learners as they handle challenging, demanding but dynamic technological needs in the 21st century.

1.9 Structure of the thesis

Following this chapter, **Chapter 2 (Literature Review)** offers a critical discussion of the concept of CT, and how it can be facilitated through the use of simulation software. **Chapter 3 (Methodology)** explains the methodologies and methods adopted in this study. More specifically, the rationale leading to the philosophical stance of the author is discussed, and the adopted methodologies are then explained. This is followed by **Chapter 4 (Results and Discussion)** where findings are presented and discussed in relation to each of the three research questions. Finally, **Chapter 5 (Conclusion)** summarises the study's key findings and provides recommendations for further research and discusses the study's limitations.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The rapid growth of technology and its influence in our everyday life on business, private and public organisations, has undoubtedly raised some challenges in education (Wing, 2008). For instance, CS students should be able to develop computing systems, tools and applications which can be used in businesses, organisations as well as in education (Angeli et al., 2016; Wing, 2006, 2011). Ertmer, Ottenbreit-Lftwich, Sadik, Sendurur and Sendurur (2012), quoting McCain (2005), argues that it is not the use of computing in classroom that is an issue, but the ability to develop learners' thinking skills using computing to solve problems. Literature (e.g., Kules, 2016; Papert, 1993; Tedre & Denning, 2016; Wing, 2014) has shown that computers develop students' mind in solving problems thereby stimulating their creativity, imaginations and innovation (Wing, 2006, 2008) leading them to becoming producers of technology (Angeli et al., 2016). Now the question is, how do educators develop such kind of skills in Computer Sciences students? If students have these skills already, to what extent are they able to use them in developing or designing systems? In answers to some of these questions, educators have developed different courses focusing on computing principles rather than on computer programming skills (Wing, 2008); also, have engaged in the development of CT skills which consider a broader picture of disciplines (e.g., Tedre & Denning, 2016, Wing, 2011). However, there are more questions which need to be addressed further, leading to this study – the concepts and application of CT to Computer Networks students.

The structure of this chapter starts with the focus on the definitions of CT and then the discussion on how CT has been applied and developed particular at university level. Furthermore, literature shows different studies on how simulation software has been used to

develop students' problem-solving skills. The chapter concludes by discussing the research conceptual framework leading to the three research questions upon which this study is based.

2.2 **Computational thinking and its definitions**

Computational thinking (CT) is a conceptual view that has been strongly revived and advocated by Jeannette Wing since 2006 highlighting that "Computational thinking represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" (p. 1). This definition sparkled a great deal of debate (Tedre & Denning, 2016). Literature (e.g., Angeli et al., 2016; Barr & Stephenson, 2011; Grover & Pea, 2013) has shown that there is no clear outstanding agreed upon definition of CT. Drawing from relevant literature as shown in Table 2.1, CT seems to be associated with three key recurring concepts: *abstraction, decomposition, generalisation.*

Table 2.1
Core concepts of computational thinking

Authors	Concepts of Computational Thinking (CT)	Widely agreed core concepts of CT
Voogt et al. (2015)	Abstractions (mental tools of computing to solve problems); layers (problem solved on different levels); relationship between layers and abstractions; algorithmic approach to problems; thought process to solving problems	• abstraction • algorithms
Barr & Stephenson (2011)	Data collection; data analysis; data representation; problem decomposition; abstraction; algorithms and procedures; automation; parallelization; simulation	• problem decomposition • abstraction • algorithm and procedures
Grover & Pea (2013)	Abstractions and pattern generalisations (including models & simulations); systematic processing of information; symbol systems and representations; algorithmic notions of flow of control; structured problem decomposition (modularising); iterative, recursive and parallel thinking; conditional logic; debugging and systematic error detection	• abstraction; generalisation • algorithmic notions • decomposition • debugging
Angeli et al. (2016)	Abstraction; generalisation; decomposition; algorithmic thinking; debugging (detection & correction of errors)	• abstraction; generalisation • decomposition • algorithmic thinking
Repenning et al. (2016)	Problem formulation (abstraction); solution expression (automation); execution & evaluation (analysis)	• abstraction
Yadav et al. (2011)	Problem identification & decomposition; abstraction; logical thinking; algorithms; debugging	• problem identification & decomposition • abstraction • algorithms
Lu & Fletcher (2009)	solving problems & designing systems based on fundamental of computer science; levels of abstraction to fully understand the problem (decomposition); algorithmic approach to solve problems	• abstraction • algorithmic approach • decomposition
Seiter & Foreman (2013)	Procedures & algorithms; problem decomposition; parallelization & synchronisation; abstraction; data representation	• algorithms • decomposition • abstraction
Kalelioglu et al. (2016)	Abstraction; algorithmic thinking; problem solving; pattern recognition (generalisation); design-based thinking; conceptualising; decomposition; automation; analysis; test & debugging; generalisation; mathematical reasoning; implementing solutions; modelling.	• abstraction; generalisation • algorithmic thinking • decomposition

2.2.1 Abstraction

Abstraction is the ability to strip off a complex problem to its bare essentials (Lee et al., 2011). In other words, "making problems or systems easier to think about" (Csizmadia et al., 2015, p.7) by highlighting details which are important and hiding other unnecessary details from the user (Csizmadia et al., 2015; Wing, 2008). The process of abstraction may involve solving problems at a different layer of a system (Wing, 2008). For example, using teaching and assessment activities, Curzon, Dorling, Ng, Selby and Woollard (2014) show how primary school students may understand layers of abstraction by using computer architecture. In Curzon et al's (2014) study, students identified and studied the hardware of computer architecture, then the operating systems which operate on the hardware and then the applications which are driven by the operating system. Students were taught the level of system abstraction from hardware to operating systems and then to the applications. Each of these layers (i.e. hardware, operating system and applications) hides the unnecessary detail of a layer below it. Wing (2008) argues that working at different layer of abstraction enables computer scientist to build complex systems. For example, the ability to interleave two different algorithms, requires the right abstraction to get the desired output (Wing, 2008). From the teaching and learning point of view, the concept of abstraction should enable CS students to identify details of systems which are important and relevant in designing and troubleshooting systems while at the same time hide unnecessary details without losing anything that is significant (Csizmadia et al., 2015). Therefore, the process of abstraction enables students to work on problems at the level of their understanding. Abstraction can be demonstrated by visual representation of an abstract; for example, using symbols, diagrams, concepts which are difficult to understand in their raw context (Repenning et al., 2016; Willis & Miertscchin, 2005). Therefore, the process of abstraction enables students to work on problems at the level of their understanding.

2.2.2 Decomposition

Decomposition is the process of abstraction (Wing, 2008) where a complex system or task is broken into the level that it can be understood, solved, developed and evaluated separately (e.g., Csizmadia et al., 2015; Lee et al., 2011; Selby, 2015). For example, when building a complex network infrastructure different people can independently work on switching, routing and security respectively. The different constituents' parts (i.e. switching, routing and security) can eventually be integrated to form a single functional complex network infrastructure. This process is true when troubleshooting a complex network infrastructure. Switching, routing and security constituents' parts may be diagnosed independently to identify faults leading to a solution for the entire complex network infrastructure. The process of working on small constituent parts of a complex network system enable students to easily understand, solve, develop and evaluate systems separately (Curzon et al., 2014).

2.2.3 Generalisation

Generalisation is the ability of reducing the complexity of a problem by replacing it with multiple smaller but similar functionalities (Angeli et al., 2016). With the concepts of generalisation, students identify patterns of similarities, commonalities and/or connections of a given problem as a process of developing CT skills (Bocconi, Chioccariello, Dettori, Ferrari & Engelhardt, 2016). Students' prior knowledge and experiences play significant role to demonstrate the development of CT (Csizmadia et al., 2015). Typical example is, while students are solving a problem they can ask questions such as "is this similar to a problem I've already solved?" and "How is it different?" are important here, as is the process of recognising patterns both in the data being used and the processes/strategies being used" (Csizmadia et al., 2015, p.8). Students should be able to apply the skills that they learnt and used previously in solving different problems. The ability to map similarities, differences, patterns of previous

solutions in problem solving encapsulate the concept of generalisation (Angeli et al., 2016). Angeli et al. (2016) argue that abstraction and generalisation are often used together to provide greater utility.

2.3 Different views about the application of computational thinking

Literature shows different views on how and where CT can be applied. For instance, Hambrusch et al. (2009) and Repenning et al. (2016) argue that CT is applicable in only Computing discipline because the conceptual ideas of CT are developed and practiced from programming. However, Voogt et al. (2015) argue that although programming, computer science and computational thinking are interconnected, they do not study the same concepts. Other scholars (e.g., Aho, 2012; Wing, 2011) argue that CT is not only a thought process for problem solving constrained in the Computing discipline but across other disciplines as well. For instance, Wing (2011) argues that CT can be applied in the field of Computing, Biology, Humanities, Arts, History, Sports and many other disciplines. However, other scholars (e.g., Hemmendiger, 2010) argue against the notion that CT is applicable across many disciplines. For instance, Hemmendiger (2010) argues that computer scientists are trying to compel non-computer scientist to think like computer scientist. Hemendiger's argument is that this is the same as teaching an economist, artist, or physicist to apply CT concepts on how to use computation to solve problems in their disciplines (e.g. economic, art and physics) and formulate new research questions which can be fruitful for further exploration and application. The debate from the cited literature remain consistent with other scholars (e.g., Kalelioglu et al., 2016; Tedre & Denning, 2016; Selby & Woollard, 2013) who argue that the literature for CT is at an early stage of maturity to come up with a clear definition of CT, how to teach and assess the skills.

Following the reviewed literature, the working definition of CT in this study is:

> *the human thought process, using fundamentals of computer sciences in designing computer systems using, but not limited, to abstraction, decomposition, generalisation to solving technological network design problems*.

2.4 **Importance of computational thinking**

We are living in the world which is mostly driven by computation (e.g., Barr & Stephenson, 2011; Ertmer et al., 2012; Harris, Jones & Baba, 2013). Barr and Stephenson (2011) argue that students live in the world heavily influenced by computing therefore applying the concepts of CT in problem solving will adept students in the digital world (e.g., Bocconi et al., 2016; Harris et al., 2013; Wing, 2008). As far as 25 years old now, literature (e.g., McGuinness, 1993; Papert, 1993) has shown that computing attempts to increase the human mind in solving complex technological problems. Many studies (e.g., Bocconi et al., 2016; Pulimood, Pearson & Bates, 2016; Selby, 2012; Wing, 2011) argue that CT is the means of developing learners problem-solving abilities; developing students' innovation and creativity (Wing, 2006, 2011); and preparing the current generation to fit in the 21st century society (e.g., Angeli et al., 2016; Bocconi et al., 2016; Selby, 2012).

Wing (2008) cites some examples demonstrating how computational methods help the human mind to design complex systems. She gives examples of how Aerospace rely on computational methods to simulate an entire aircraft or space mission; how humanities depend on computational methods to discover patterns, trends and links in understanding and appreciating humankind; computational methods help humans to model and design complex systems from the huge amount of data generated by computation which would not be possible for a human.

To that effect, Wing (2006, 2008) advocates that when computers are involved in problem-solving, they need to "think" like humans and not humans think like computers.

Bocconi et al. (2016) conducted an extensive and comprehensive study in understanding and developing CT in compulsory education. The study was mostly qualitative approach involving desk research with over 570 reviewed literature, a survey of 18 ministries of Education in Europe and beyond, semi-structured interviews from expert practitioners and policy makers. In their findings, they concluded that CT gives primary and secondary school students the skills they need to thrive in the digital age and economy by fostering their logical thinking skills; problem solving skills; attracting more students into Computer Sciences and fostering employability in the ICT sector. Currently, over 13 countries in Europe and beyond have made CT as a mandatory subject to be integrated in the primary and secondary school curriculum (e.g., Bocconi et al., 2016; Grover & Pea, 2013; Wing, 2014).

In summary, CT uses scientific methodologies in developing students' inventive, innovative and imaginative thinking skills (e.g., Papert, 1993; Ulger, 2016; Voskoglou & Buckley, 2012). CT is the computer science approach to problem solving (e.g., Bocconi et al., 2016; Kalelioglu et al., 2016; Wing, 2011) using CT concepts such as, abstraction, decomposition and generalisation. CT gives students the computing thinking skills to thrive in the digital age, fosters their logical thinking skills, attract more students into computer science and more chances in ICT employment (e.g., Bocconi et al., 2016; Yadav, Zhou, Mayfield, Hambrusch & Korb, 2011). The overarching significance of CT is to prepare the 21st generation to fit and excel in the digital age with the hope of further improving the world's economy by producing more Computer Sciences students who can produce technologies (e.g., Angeli et al., 2016; Bocconi et al., 2016).

2.5 **Computational thinking at the university level**

Extensive research (e.g., Barr & Stephenson, 2011; Grover & Pea, 2013; Lu & Fletcher, 2009; Wing, 2008) has advocated the use of CT in primary and secondary schools. The government agencies and private interests including the National Science Foundation, British Royal Society, Microsoft and Google have supported that advocacy and effort (Wing, 2010). For instance, countries such as Russia, South Africa, New Zealand, Australia, UK, USA, to mention a few, have already implemented CT in their schools' curriculum (e.g., Bocconi et al., 2016; Czerkawski & Lyman, 2015; Grover & Pea, 2013; Wing, 2014). The tenet behind introducing CT at primary and secondary school level is to develop children's thinking skills before handling complex problems at university (Lu & Fletcher, 2009). Wing (2006) argues that "to reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (p. 33).

However, there is little evidence particularly in the area of Computer Networks courses on how CT has been advocated and implemented at university level (Czerkawski & Lyman, 2015). Wing (2008, p. 3720) makes assumptions that CT is applied at university level:

> Let us assume that the trend of using computational thinking in research in all fields is already happening, thereby already influencing the training of graduate students. Let us further assume that universities have already begun to incorporate computational thinking in their undergraduate curricula, thereby recognizing how the next generation will have to be able to think in order to succeed in modern society.

This assumption shows her uncertainty of how much CT has not been applied at university level. In fact, her persuasion to start applying CT at the elementary level (Wing, 2006) is with the hope that when these students progress to universities will have already acquired their CT skills to handle complex design problems (e.g., Barr & Stephenson, 2011; Lu & Fletcher, 2009;

Yadav et al., 2011). All this shows that there is little research showing how CT is fully developed and practiced at university level.

On another hand, in courses such as Computer Sciences at university level, where CT has been sparingly applied, the focus has been on the use of programming (i.e. in Software Development and Games courses) (e.g., Berland & Lee, 2011; Selby, 2015; Kazimoglu, Kiernan, Bacon & Mackinnon, 2010; Repenning et al., 2016). For instance, Kazimoglu et al. (2010) show teaching programming to undergraduate students using puzzle-solving computing game. The game comes as part of the learning integral showing how programming concepts work. Selby (2015) uses programming tool to demonstrate the relationship between CT and Bloom's Taxonomy. Berland and Lee (2011) use non-computational media called *Pandemic* (board games) as a way of engaging undergraduate students in complex computational thinking. Repenning et al. (2016) show different types of programming tools that are used to facilitate the application of CT. However, the study of Yadav et al. (2011) discussed in Section 2.10.1.2 shows how CT has been applied at university level without the use of programming. It was a small study for only a week and therefore their findings are to be treated with caution. Furthermore, focusing on primary and secondary schools, the studies of many scholars (e.g., Bocconi et al., 2016; Brennan & Resnick, 2012; Seiter & Foreman, 2013) show how CT has been applied using programming.

From the reviewed literature there is little evidence showing the wider use and application of CT at university level. It is mainly in programming, noticeable at primary and secondary school levels, where there are some evidences on how CT has been applied. However, Wing (2011) argues that CT can be applied in all disciplines including Engineering, Biology, Journalism, Sports and Humanities. It is in view of this gap that this study is conducted at university level and to students whose study focus is not on Programming but rather Computer Networks.

2.6 Existing strategies and tools to develop computational thinking

Research from many scholars (e.g., Grover & Pea, 2013; Repenning et al., 2016; Willis & Miertschin, 2005) has shown that there are many different tools which have been used in the discipline of programming to foster CT skills. For instance, Grover and Pea (2013) cite the examples of graphical programming tools, such as Scratch, Game maker, Kodu and web-based simulation tools, such as AgentSheet and Agent cubes. Grover and Pea argue that graphical programming tools, such as Scratch are easy to use and that they enable students to design and create programs by "snapping together graphical blocks that control the action of different dynamic actions on the screen" (p. 40). The graphical representation of a problem, such as the use of Scratch programming tool broadens students' thinking skills (Willis & Miertschin, 2005); and unfolds their different perceptions (Repenning et al., 2016) thereby develop students' broader, wider and deeper thinking. For instance, Willis and Miertschin, quoting the work of The Institute for the Advancement of Research in Education (IARE) argue that:

> research has shown that visual learning techniques help students: (1) make abstract ideas visible and concrete; (2) connect prior knowledge and new concepts; (3) provide structure for thinking, writing, discussing, planning, and reporting; and (4) focus thoughts and ideas that lead to understanding and interpretation (p. 250)

Repenning et al. (2016) cite more different examples of tools that foster CT such as simulations, Mindmap, Boxer, ToonTalk which are based on the notion of microworlds. Repenning and his colleagues further argue that these tools should be able to support three stages of CT process, namely: problem formulation, solution expression and solution execution. However, the focus of these tools discussed here are broadly applied at primary and secondary school levels, particularly on programming related disciplines. Literature has not shown clear tools that can be used to foster CT when designing computer networks. However, simulation tools such as Cisco Packet Tracer, GNS3, Opnet provide conducive

platform for students to design and create their own simulated network systems (e.g., Ruiz-Martinez, Pereniguez-Garcia, Marin-Lopez, Ruiz-Martínez & Skarmeta-Gomez, 2013; Zhang, Liang & Ma, 2012) which according to Willis and Miertscchin's (2005) argument help student to broaden their problem-solving and logical thinking skills.

2.7 **Simulation software**

Simulation software provides an environment where simulated devices such as routers, switches, cables, PCs and protocols mimic the real physical devices and software operation (Ruiz-Martinez et al. 2013; Yalcin, Altun & Kose, 2015). Additionally, simulation software provides flexible and unlimited devices which can be deployed and used in designing virtually limitless simulated network infrastructures. With the ability to drag and drop devices (Exposito, Trujillo & Gamess, 2010), students can add, edit and remove as many simulated devices as they can possibly wish, thereby extending their imaginations (Ruiz-Martinez et al., 2013).

However, emulation software, is slightly different in that emulation use real image programs. For example, GNS (Graphical Network Simulator) uses real image programs of Cisco routers and switches (Exposito et al., 2010). All devices used in emulation software are simulated. Because of the use of real image programs, emulation software can be more power-demanding – thus emulation software requires enough memory, hard disc space, and considerable powerful graphical interface. Students need to have appropriate licenses to install image programs to run emulation program legally. This problem can limit the number of devices which can used in designing complex networks (Exposito et al., 2010).

Virtualisation software is not simulation neither emulation software because using virtualisation software, student can deploy different operating systems and run live systems on physical devices in real-time (Ruiz-Martinez et al., 2013). The problem with virtualisation software is that it may take away the element of flexibility that is offered by simulation and

emulation software because students need to run with some physical infrastructure. Students have to have physical devices to facilitate the virtualisation. Having explained the differences between simulation, emulation and virtualization software, these terms are sometimes used interchangeably to refer to virtualised software (Ruiz-Martinez et al., 2013).

2.7.1 Types of simulation software

There are many simulation and emulation software which can be used to design computer networks (Dobrilovic & Odadzic, 2006; Ruiz-Martinez et al., 2013). For instance, Cisco Packet Tracer, GNS3, NS3, Opnet, VIRL, among others. Each of these software tools have their own strengths and weaknesses. GNS is an open source emulator which can be used to design different types of computer networks and run them as if on real physical devices. However, it has some limitations needing license image programs to be installed first for simulated devices and software to run legally. GNS is also power-demanding software which requires a lot of memory thereby limited to only smaller topologies of networks which can be designed (Exposito et al., 2010). NS3 is a discrete-event simulator that is mainly used by students who are studying for pure Computer Sciences where the focus is software programming. Opnet (which is now known as Riverbed Modeler) is another simulator that can also be used to design computer networks however does not provide the exact image programs that runs on normal physical devices. It is usually designed to be used at an advanced level where the network performance needs to be monitored, such us utilization, delay, jitter, response time, QoS etc. Packet Tracer is a Cisco proprietary simulation software which covers a larger population of computer networking specialist across the world (Janitor et al., 2010; Ristov, Spasov & Gusev, 2015; Zhang et al, 2012), particularly those trained and taught in all centres of Cisco networking academies. It is flexible and easy to use. Most of the universities in the UK use Cisco software and hardware products (i.e. Routers, switches, firewalls) for teaching students on computer networks design (Ristov, et al., 2015; Sun, Wu, Zhang & Yin, 2013). Packet

Tracer requires basic training for students to know how to use it. This simulation software is ideal for any student from novice to an advanced student in networking (Zhang & Yin, 2013). Packet Tracer provides a virtualised environment that helps students to consolidate their theoretical lessons through practice in a very flexible and realistic manner (Ruiz-Martinez et al. 2013).

2.7.2 Rationale for using simulation software to apply computational thinking

Simulation software provides a platform on which students can design, build and test networks that vary in complexity from basic to complex simulations of the infrastructure. Many studies (e.g., Galan, Fernandez, Fuertes, Gomez, & de Vergara, 2009; Hwang, Kongcharoen & Ghinea, 2014; Su et al., 2013) have shown that, in most cases, simulation software provides a highly realistic way of teaching computer networks, conducting research and experimenting in designing complex network systems, which may be limited on real physical devices. Using such software, students are able to work with systems that are far too complex for them to be able to build on physical harware devices. Additionally, such software enables students to experiement technologies that they would otherwise not meet at University.

Teaching students to build a relatively simple physical network that includes a couple of routers and a few switches can require racks of dedicated hardware. Simulation software is an ideal technology on teaching computer networks because it provides rich platform for unlimited devices and software (e.g., Galan, Fernandez, Ruiz, Walid & Miguel, 2004; Ruiz-Martinez et al., 2013) which would not ordinarily be possible on physical devices (Su et al., 2013). 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.

With the use of physical devices students do not have the flexibility to work on their network designs from home or anywhere they wish because they are fixed devices (Zhang et al., 2012). While simulation software provides the flexibility to work on their network design away from classroom environment to anywhere they wish (Zhang et al., 2012). Besides, simulation software provides feature-rich, flexible platforms with a range of devices and software that is far greater than would be possible when using physical devices (Galan et al., 2004; Ruiz-Martinez et al., 2013). Simulation software provides an evironment that stimulates students inventiveness, innovation and problem-solving skills (Ruiz-Martinez et al., 2013), which are keys elements to developing students' computaional thinking (Wing, 2011).

2.7.3 Simulation software and theoretical underpinning constructionism

Simulation software provides a graphical visual representation platform which according to research (e.g., Janitor et al., 2010; Zhang et al., 2012) enhances students' learning particularly on concepts which are theoretically difficult to understand (e.g., Musheer, Sotnikov & Heydari, 2012; Willis & Miertschin, 2005). Students can construct basic to advanced simulated networks which enhances their own understanding and application of abstracts learnt in class, leading us to the theories of constructionism and constructivism.

Constructionism as a learning theory was coined by a constructionist, Seymour Papert who focused on tools, such as computers and media to aid in child's cognitive development (Papert, 1993). Both constructivism and constructionism are within the epistemology of how learners learn but these approaches are slightly different. For instance, while constructivism focuses on the internalization of a child's cognitive development by means of social interaction (Fosnot, 2013; Vygotsky, 1978), constructionism focuses on the externalisation of a child's development by the use of external tools such as computers, media, simulation software, etc. Learners externalise (or project out) their understanding to the world for it to be probed or

discussed. To achieve this, learners have to create a 'public artefact' e.g. their own computer network design using computer tools, such as simulation software. Constructivism theory was developed at the time when learning was not impacted by technology. Therefore, according to Ackemann (2001), constructivism overlooks the significance of computer-aided tools and individual preference in human learning and development and that's where constructionism phenomenon comes to address that.

The ideas of constructionism are drawn from the construct of constructivism that children have a natural ability to conceptualise (Knowles, 1970). Knowles (1970) argues that except Piaget and Bruner who "discovered that children have a natural ability to conceptualise", […] "most educational psychologists gave attention to studying the reaction of children to teaching […] and training teachers how to control students' reaction to their teaching" (p. 19). Knowles' argument is "focusing on what happens inside the learner rather than what the teacher does" (p.19). This is the concept of internationalisation. On another hand, the concept of externalisation is based on the use of external tools e.g. computers (Papert, 1991) which a student may use to create or produce an 'artefact' that may probe discussion and further emerge ideas. Papert added the value of constructivism by emphasising on the use of external aided computer tools and media in learning (Ackermann, 2001) resulting in the concept known as constructionism. A learner is able to step back and use computing tools to ask questions like, 'how does this work?' In trying to answer that question, a learner is able to 'learn by making' (Papert & Harel, 1991) thereby students are able to decompose an abstract into something they can understand and solve. Papert's approach to constructionism is the use of computer-based technologies en-route to enabling a learner to construct his/her own knowledge. As alluded to before, externalisation is linked to the idea of learners creating a 'public artefact' e.g. their own computer network design. Students have to create something to project out or externalise their current understanding of whatever they are learning. In this study, students are using simulation

software as a computing tool to project out their computer network design artefact. This is the concept of externalisation.

Computing tools put learners at the centre of learning (Cole, 2009) enhances critical thinking (Papert, 1993; Schneckenberg, 2014) and promotes their problem-solving skills (Pulimood et al., 2016; Wing, 2006). Computing tools also provide rich environment for collaborative and cooperative learning (Harris et al., 2013; Schneckenberg, 2014). Therefore, learners are able to develop their thought processes which enrich them to solve problems as they design artefact (Voogt et al., 2015). All this is the factor of externalisation that enhances the concepts of constructivism to form constructionism. The theory of constructionism is therefore based on the concept of producing "the most learning for the least teaching", (Papert, 1991, p. 139). It is in view of this understanding that this study is applying the tool of simulation software to enhance the development of students' computational thinking skills.

2.7.4 The case of Cisco packet tracer simulation software

Sun et al. (2013) conducted series of experiments investigating the effectiveness of teaching the concepts of computer networks to undergraduate students using physical equipment against simulation software in three consecutive academic years. The experiments were conducted at University of Science and Technology Liaoning in China. For instance, in 2007-2008, students used physical devices; in 2009-2010, students used Packet Tracer while in 2011-2012, students used Dynamips and GNS3. Their results show that students achieved more when using simulation software than physical devices. For instance, 90% of students completed their course on time using Packet Tracer than the use of physical devices (70%) and Dynamips and GNS3 (85%) respectively. They also found out that simulation software played an effective role in helping students understand abstracts of computer networks, increased students' learning interest and achieved better grades.

The study of Zhang et al. (2012) show how they used Packet Tracer to develop students' creativity and innovation in designing basic to advanced networks. Their main approach was using mini-projects in which students were not provided with "student do-as-told exercise" (p. 507). Students were given problems which needed solving with specification clearly in detail without giving them guidance. After completing their mini-projects, according to Zhang et al. (2012) students' feedback showed that they were interested to do mini-projects than given the "do-as-told" exercise, students understood the abstracts of computer networks easily and it was convenient for student to see where they did well and/or wrong in their problem-solving process. Packet Tracer provided a platform for easy visual learning. On another hand, others, such as the work of Musheer et al. (2012) have used the multiuser feature in Packet Tracer to create a collaborative and interactive learning environment to enhance students' understanding of complex networking concepts. The multiuser feature provides the ability to use games, such as Cisco ASPIRE Beta 4 and Cisco myPlanNet (Musheer et al., 2012) in which students are asked to complete series of technical requirements such as, choosing the right hardware, applying correct configuration schemes, etc thereby enhancing students understanding of complex networking concepts. All these studies show the effectiveness of using Packet Tracer in teaching and learning the complexity of computer networks which would be difficult on physical devices (Sun et al., 2013; Ruiz-Martinez et al., 2013). Nonetheless, all these studies did not focus on how students may apply their CT skills in designing and troubleshooting computer networks via simulation software. It is in view of this observation that this study focuses on such gaps. Figure 2.1 provides an interface of Cisco Packet Tracer simulation software with basic annotation of key features used in designing and troubleshooting simulated network systems.

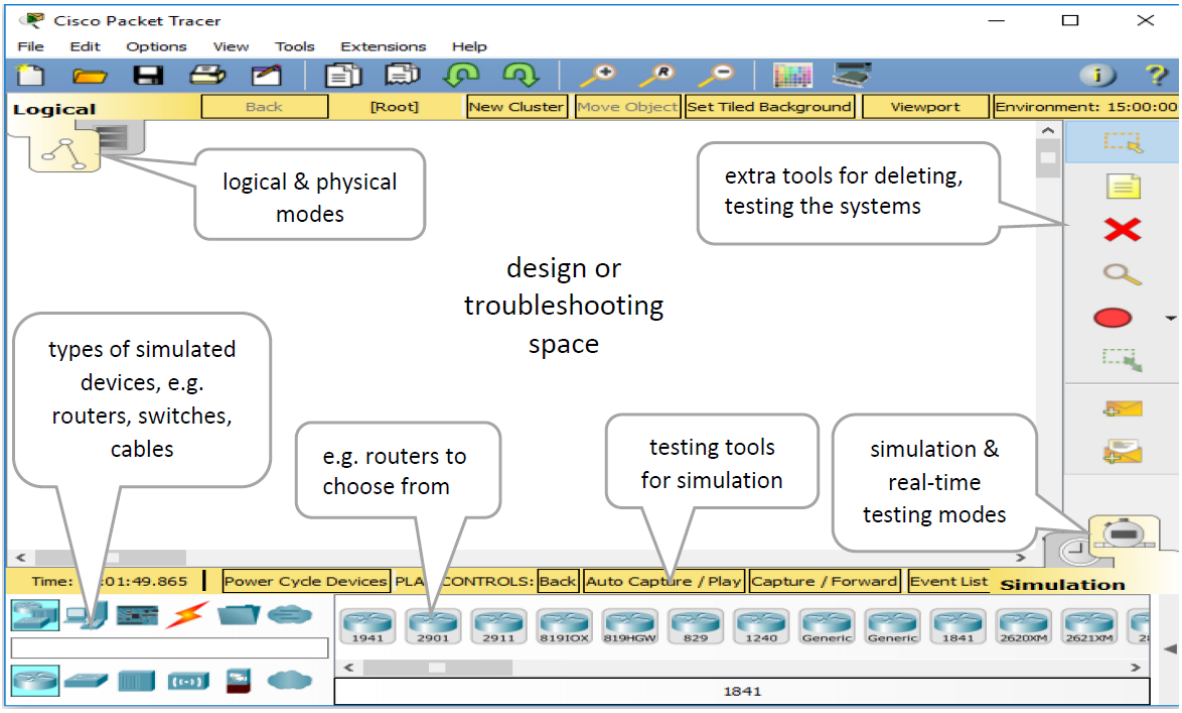


Figure 2.1. Packet Tracer Design Interface

2.8 **Rationale for computer networks**

Literature has revealed that CT has been applied mainly at primary and secondary school levels. However, there is little evidence showing how CT has been applied at university level particularly on learners studying on Computer Networks course. In instances, where CT has been applied at university level, the major focus has been on the use of programming (Selby, 2012, 2015). Nonetheless, the use of programming to develop students' CT has not gone away without debate. Some scholars (e.g., Grover & Pea, 2013; Mannila et al., 2014; Repenning et al., 2016) strongly argue that CT is best applied by the concepts of programming while Lu and Fletcher (2009), Yadav et al. (2011) and Voogt et al. (2015) argue that programming is not essential in teaching CT. Teaching CT through programming may even deter students from being interested in Computer Sciences (Voogt et al., 2015). Furthermore, there is notable and insightful work in New Zealand (http://csunplugged.org/), demonstrating how CT can be developed without the use of computers (i.e. programming). However, their focus is at primary

and secondary school level. Wing (2006, 2011) advocates that CT is applicable across all disciplines and at all levels. However, there is no evidence how CT has been applied in the field of Computer Networks. There are many tools and strategies which have been applied and implemented to facilitate the teaching and application of CT such as explained in Section 2.6. However, all these tools and strategies are based on the concepts of programming, games and robotics (Mannila et al., 2014; Repenning et al., 2016). However, there is no clear research to evidence how CT has developed learners studying on Computer Networks course. This is an interesting observation because the study of Computer Networks involves understanding the abstraction of computer networks (Janitor et al., 2010; Wing, 2011), decomposing abstracts of computer networks into the level that they can be understood, solved, developed and evaluated separately (Csizmadia et al., 2015), identifying patterns of similarities, commonalities or differences to troubleshoot network systems besides prior knowledge and experiences (Bocconi et al, 2016). All these concepts (abstraction, decomposition and generalisations) described are the concepts of CT (Angeli et al., 2016) which appear to fit in well in the Computer Networks curriculum.

Besides the gap in literature on the use of CT in the field of Computer Networks, networking in Computer Science is inevitable (Janitor et al., 2010; Zhang et al., 2012). For instance, programmers depend on computer networking to test, populate, share and run their programs; forensic computing requires networking to cover a wider geographical scope for data investigation; network security requires networking to provide a wider protective mechanism for an organisation; businesses, require computer networking to advertise their services and products across the world. Therefore, computer network forms a fundamental base for computer science learners where the application of CT will cover a wider area.

2.9 **Conceptual framework**

2.9.1 Introduction

The conceptual framework of this study is adopted from the conceptualisation of *technology, pedagogy,* and *content knowledge* (TPACK) for the concepts of CT as argued by Angeli et al. (2016). Angeli et al. (2016) have shown how the CT curriculum within the context of primary school level with a focus on programming-related subjects fits in the TPACK framework.

2.9.2 The framework of pedagogical content knowledge (PCK)

TPACK, which was originally known as *technological pedagogical content knowledge* (TPCK) (Koehler, Mishra, & Cain 2013), is the conceptual framework formed by Mishra and Koehler (2006) extending from the teaching and learning framework that consists of *content knowledge* (CK) and *pedagogical knowledge* (PK) to form what was known as *pedagogical content knowledge* (PCK) coined by Lee Shulman (Shulman, 1986, 1987). Shulman defines CK as teachers' subject knowledge while PK as teachers' knowledge about teaching and learning (Shulman 1987). However, Shulman's initial claim was that teachers' subject knowledge and pedagogy were often treated as mutually exclusive (Mashra & Koehler, 2006). For example, in teaching practices, teachers focused on either subject matters or pedagogy. The teaching practice of excluding subject matters from pedagogy was criticised by many scholars (e.g. Cochran, King, & DeRuiter, 1993; van Driel, Verloof, & De Vos, 1998). To bridge this dichotomy, CK and PK were integrated to form what was known as PCK framework (Shulman, 1986) as illustrated by Mishra and Kohler, (2006) in Figure 2.2.

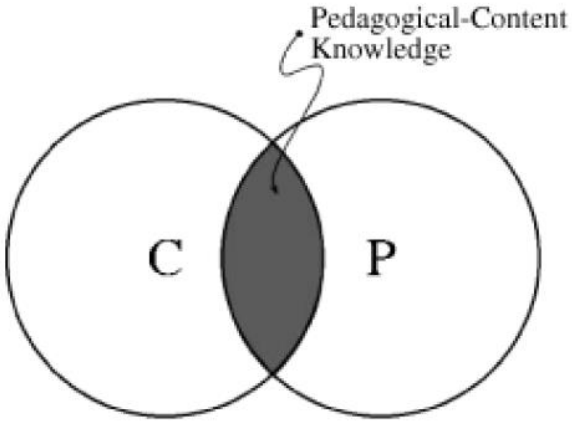


Figure 2.2. PCK Adopted from Mishra and Kohler (2006)

Additionally, the PCK framework included supplementary elements, such as curriculum knowledge, context knowledge, learner's knowledge and knowledge of educational goals and beliefs (Ertmer & Ottenbreit-Leftwich, 2010; Koehler & Mishra, 2006; Shulman, 1987). However, the PCK framework was criticised too by many scholars (e.g., Angeli & Valanides, 2009; Ertmer & Ottenbreit-Leftwich, 2010; Koehler & Mishra, 2006) that it lacked the integration of technology in this digital era.

Many scholars (e.g., Hughes, 2005; Neiss, 2005; Zhao, 2003) argue that in the digital era good teaching requires that teachers understand how technology relate to their subject matter and pedagogy. Therefore, from early 2000, Mishra and Kohler (2006) conducted several studies with educational scholars (Ferdig, Mishra & Zhao, 2004), researchers (Koehler & Mishra, 2005; Vya & Mishra, 2002) and practitioners (Koehler & Mishra, 2002) to form theories and practices on how teachers can integrate the use of technology in their teaching practice. In 2006, Mishra and Kohler (2006) officially integrated *technology* to the PCK framework to form what was known as TPCK as shown in Figure 2.3 and later it was known as TPACK as shown in *Figure 2.4*.

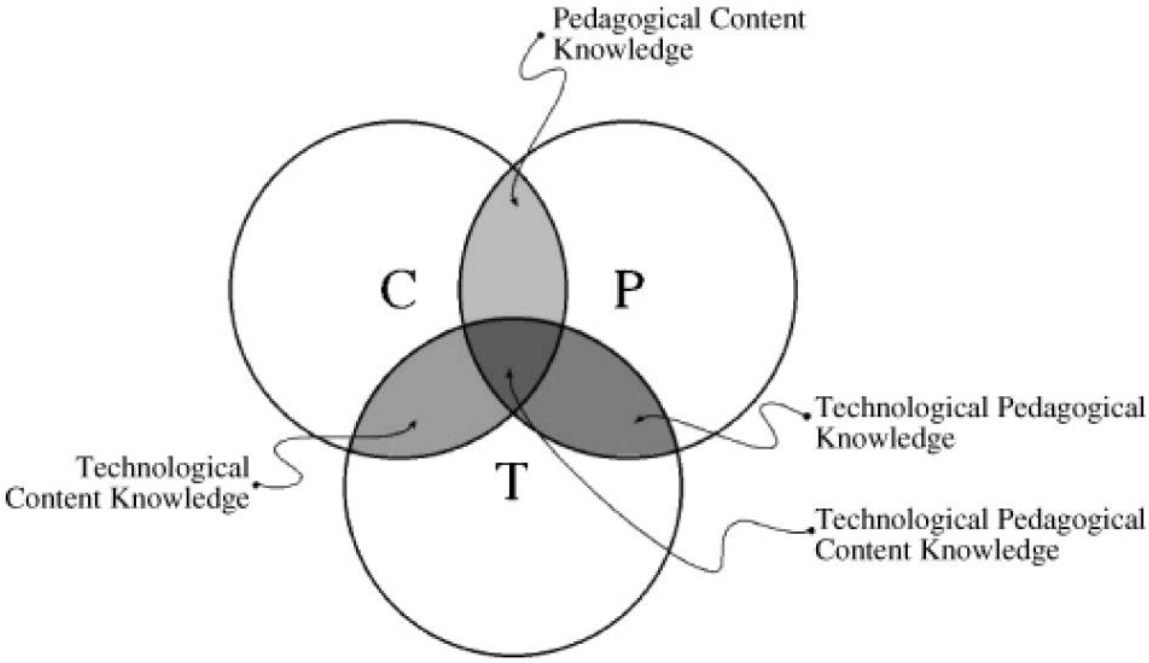


Figure 2.3. TPCK Adopted from Mishra and Kohler (2006)

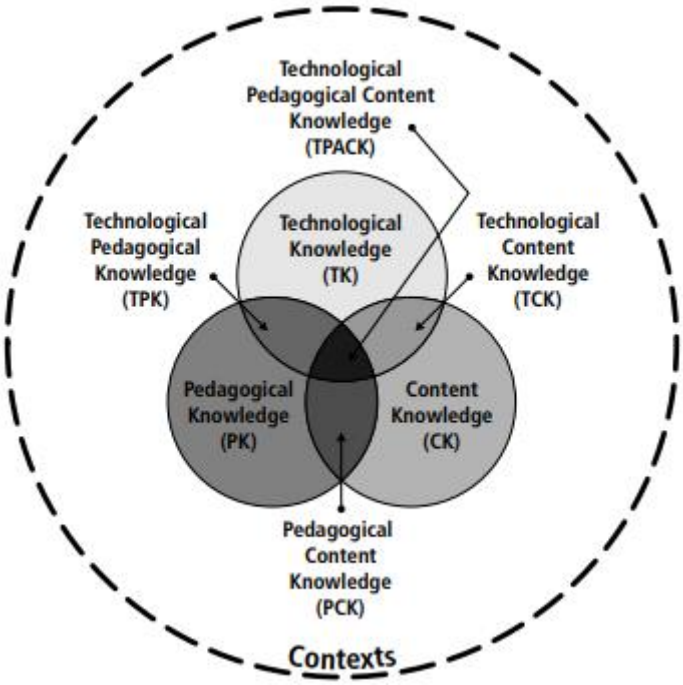


Figure 2.4. TPACK adopted from Koehler, Mishra and Cain (2013)

2.9.3 The concept of TPACK

The integration of technology to the PCK framework has resulted in the formation of three distinct but interdependent relationships – i.e. *technological content knowledge* (TCK); *technological pedagogical knowledge* (TPK); and *technology, pedagogy, and content knowledge* (TPACK). TCK is how the subject matters can be represented, taught, assessed using technology. The argument of Mishra and Koehler (2006) is that not only should teachers know the subject matter but also know how to use technology to represent the subject matter in a manner that is easy and effective for teaching and learning experiences. TPK involves knowledge of various forms of technologies which can be used to facilitate teaching and learning. This includes the ability to choose the right technological tool to fit in the right pedagogical strategies. For example, the use of simulation software to simulate complex problems which could otherwise be difficult to teach and learn within the specific teaching context. TPACK is therefore, the framework of using technology in presenting content knowledge and facilitating the construction of pedagogical strategies to address teaching and learning concepts which may appear difficult to students. Therefore, the distinct conceptualisation of TCK and TPK is interconnected to show the dependence of the two concepts (*TCK and TPK*) to form TPACK. In other words, Koehler, Mishra and Cain (2013) show that the three relationships described above are distinct but interdependent for good teaching and learning practice.

2.9.4 Conceptualisation of TPACK in other studies

Drawing from the TPACK framework, Angeli et al. (2016) argue that TPACK is significant to the field of Computer Sciences because technology is the vehicle for teaching, learning and above all producing technologies in Computer Sciences. Therefore, Angeli et al. (2016) used

the conceptualisation of TPACK to show how CT can be taught within the context of programming-related subject at primary school level.

Angeli et al. (2016) give typical examples on how each element of TPACK can be applied in practice in the context of CT curriculum. For instance, they argue that *content knowledge* for CT is the knowledge about CT such as, understanding the skills of *abstraction, decomposition, generalisation* among other core concepts of CT. Additionally, they argue that *pedagogical knowledge* for CT comprise teaching strategies which model how to solve problems, present and explain solutions in step by step processes, and also show how a complex problem can be decomposed into simpler problems. Furthermore, Angeli et al. (2016) link the *context knowledge* to the programming related subject at primary school level. They contend that *technology knowledge* for CT comprise skills about how to operate and use variety of technologies, produce new technologies and how to solve tasks using technologies while *learners' knowledge* for CT comprise knowledge about learners' difficulties in developing abstractions, decompositions, generalisation among other core concepts of CT.

2.9.5 Conceptualisation of TPACK framework in this study

In this current study, TPACK is not adopted as a means of demonstrating how technology can be used by teachers as argued by many scholars (e.g., Angeli et al., 2016; Angeli & Valanides, 2009; Niess, 2005), but rather provides a framework on investigating students' and lecturers' understanding (*knowledge*) of CT (*content knowledge*) and their perceptions of using simulation software (*technology*) to facilitate students' CT skills within the *context knowledge* of Computer Networks course as shown in Figure 2.5. The findings of this study provide appropriate *pedagogical knowledge* recommendations to lecturers. The components of the TPACK helped to frame the research focus of this study as explained below:

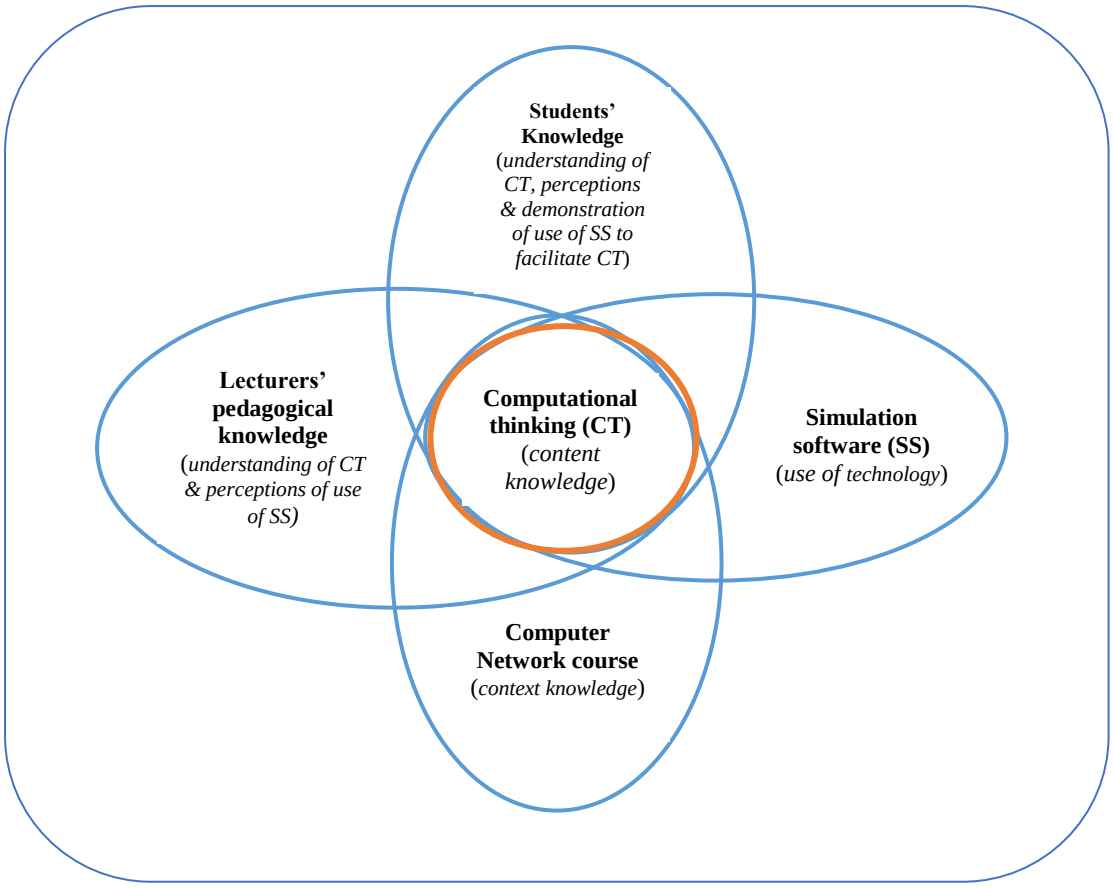


Figure 2.5. Conceptual Framework of this Study

2.9.5.1 Computational thinking (content knowledge)

Literature reviewed in Section 2.2 has shown that there is no consensus definition of CT but rather scholars appear to agree on some common core concepts of CT, such as abstraction, decomposition and generalisation. In view of the commonly agreed core concepts of CT, the investigation of students' and lecturers' understanding of CT is based on the content knowledge of the core concepts of CT such as, abstraction, decomposition and generalisation. Students' content knowledge about CT is furthermore investigated by solving computer network design problems via simulation software (*technology*) within the context of computer networks at university level. Similarly, in the CT framework of Angeli et al. (2016), they argue that content knowledge is students' and lecturers' understanding about the skills of abstraction,

decomposition, generalisation among other core concepts of CT. Furthermore, many studies (e.g. Bocconi et al., 2016; Lee et al., 2011; Selby & Woollard, 2013) have shown how the core concepts of CT, such as abstraction, decomposition and generalisation define the conceptualisation of CT. For example, Lee et al. (2011) defines abstraction as the ability to strip off the complexity of problems to level that they can be understood; Selby and Woollard (2014) define decomposition as the ability to breakdown complex problems into smaller solvable tasks; while Bocconi et al. (2016) define generalisation as the ability to identify patterns of similarities, commonalities, differences to solve a problem. Therefore, CT becomes the focal point of this study with close relationship to Computer Network students and lecturers (*context*), and simulation software (*technology*) which fits in well with the TPACK framework.

2.9.5.2 Students' understanding (learners' knowledge)

Computer Network students are examined on their understanding (*knowledge*) of CT, perceptions of using simulation software (*technology*) and how they can apply the concepts of CT (i.e. abstraction, decomposition, generalisation) by solving computer network design problems on simulation software. In Angeli et al's (2016) conceptualisation of CT framework, they argue that learners' knowledge of CT includes learners' difficulties in developing the core concepts of CT, such as developing abstractions which are beyond programming language, decomposing complex problems to simpler ones and use of generalisation by identifying common pattern in solving problems. Computer Network students are the main target audience in this study in contrast to many studies (e.g. Barr & Stephenson, 2011; Curzon et al., 2014; Seiter & Foreman, 2013) which have focused on students pursing programming-related subjects.

2.9.5.3 Lecturers' pedagogical knowledge

In consistent with many scholars (e.g. Hughes, 2005; Mishra & Koehler, 2006; Neiss, 2005) who argue that not only should a good teacher be adept in the subject matter (*content knowledge*) but also be able to choose the right technological tool to fit in the pedagogical strategies. In this study, Computer Networks lecturers are examined on their understanding of CT and their perception of using simulation software to facilitate students' CT skills.

2.9.5.4 Computer network course (context knowledge)

As allured to in previous discussion many studies have focused on programming-related subjects therefore this study has focused on Computer Networks in Computer Science discipline. The main significant reason of choosing Computer Networks course was the gap in the literature showing no straightforward evidence how the concepts of CT have been applied within the context of Computer Networks course (see Section 2.8). There is, however, little discussion about the lesson on how CT can be applied in networking and communications by Curzon et al. (2014), but the lesson is on a small scale and within the context of primary school level. Secondly, it was feasible for my own professional development in teaching CT in the Computer Networks curriculum within the university-level context. Therefore, the Computer Networks course was used in the current study as a *context knowledge* upon which students were investigated their understanding on how to apply CT skills via simulation software.

2.9.5.5 Simulation software (technology)

Students and lecturers are being examined on their perceptions of using simulation software to develop students' CT skills. Students are further investigated on how they can apply their CT skills via simulation software. Simulation software has been chosen because many studies (e.g., Galan et al., 2009; Su et al., 2013, Zhang et al., 2012) have shown that simulation software provides conducive and flexible platform to solve basic to complex problems. For instance, via

simulation software, students can design, build, modify, test, redesign, any network of their choice thereby enhancing their imagination, creativity and innovation (Ruiz-Martinez et al., 2013). Detailed discussion on the rationale of choosing simulation software has been provided in Section 2.7.2.

In summary, the TPACK framework helped in framing the investigation of students' and lecturers' understanding (*knowledge*) of CT and how may the use of simulation software (*technology*) facilitate the application of students' CT skills within the context of Computer Networks course at the university level. The framework also helped in understanding the relationships that exist in this study between Computer Network students, lecturers, and technology in understanding the concepts and application of CT.

In view of the literature review and the discussion of the conceptual framework of this study, the following are the research questions to be answered which are discussed further from Section 2.10:

1. What are Computer Networks students' and lecturers' understanding of computational thinking?
2. What are Computer Networks students' and lecturers' perceptions of the use of simulation software to facilitate the application of students' computational thinking?
3. How might the use of simulation software facilitate the application of students' computational thinking?

2.10 **The current study and research questions**

Using popular search engines and library databases from well-known and international recognised academic publishers, such as Education Research complete (EBSCO), Institute of Electrical and Electronics Engineers (IEEE), Association for Computing Machinery (ACM)

digital library, Springer, Google scholar, etc., there is hardly any empirical study showing how CT has been applied in Computer Networks courses rather than in Programming-related subjects. For instance, using terms such as: *development of computational thinking in computer networks, the concepts of computational thinking in computer networks, computational thinking and computer networks, application of computational thinking in computer networks, etc* yield no tangible results from these databases. However, repeating the same terms by replacing computer networks with programming, hundreds of results are yielded showing how CT is applied using programming-related subjects or tools. The latter is particularly showing how programming tools have been used to facilitate CT at primary and secondary levels while this construct is sparingly noticed at university level.

2.10.1 Research Question1: 'What are Computer Networks students' and lecturers' understanding of computational thinking?'

2.10.1.1 Rationale

Considering that there is no empirical study showing how CT is applied for learners studying on Computer Networks course, it became an interesting area of research to investigate students' and lecturers' understanding of CT. Research has shown that CT can be applied in computing and non-computing disciplines (Wing, 2011) and that it is majorly applied as a problem-solving skill (Barr & Stephenson, 2011; Hambrush et al., 2009; Kalelioglu et al., 2016; Yadav et al., 2011) it was interesting to notice how CT has not been applied in the discipline of Computer Networks. Computer Networks are abstract by nature (Janitor et al., 2010; Sun et al., 2013) and that studying for computer networks involves a great deal of practical experiments and research (Dobrilovic & Odadžic, 2006; Ruiz-Martinez et al., 2013) which includes network design, problem-solving and refining network performance (Galan et al., 2009). All these cover the concepts of CT (Wing, 2006, 2008) requiring the skills of abstraction, decomposition,

generalisation, just to mention a few as discussed in Section 2.8. Therefore, this study endeavours to investigate students' and lecturers' understanding of the concepts of CT and if they do, what are their own definitions and application?

2.10.1.2 Existing empirical studies

Yadav et al. (2011) conducted a pilot survey study at Purdue University in USA, on how CT could be integrated in their "Learning and Motivation" course. A total mixture of 115 Educational Studies and Computer Science students participated in pre- and post-survey assessment investigating their understanding and attitudes of CT. Results from their pre-survey indicated that 20% of students had no idea of CT; 33% of students described CT as the process of solving problems, and 31% believed that CT can be integrated in classroom. After the pre-survey, students were provided with an overview information of CT such as (problem identification and decomposition, abstraction, logical thinking, algorithms and debugging) without the use of computers. Students also participated in necessary activities which indicated the concepts of CT. Thereafter results on their post-survey assessment showed that 0% of students indicated they did not know what computational thinking was; 86% of students described CT as the process of solving problems and 86% indicated that CT should be integrated in classroom. Their post results indicated students' improved awareness of CT and how it might be applied and integrated in teaching and learning. However, their study was conducted for a week only therefore their results are to be treated with caution.

Yadav et al.'s study focused on the use of non-computing tools to show how CT may be applied in secondary school and clearly showed that very few students had idea about CT and how it can be applied in educational discipline. However, the study failed to show how CT can be applied with the use of computers. It is in view of that gap that this research question endeavour

to investigate Computer Networks students' and lecturers' understanding of computational thinking.

2.10.2 Research Question 2: 'What are Computer Networks students' and lecturers' perceptions of the use of simulation software to facilitate the application of students' computational thinking?'

2.10.2.1 Rationale

Section 2.7.2 has shown the significance of using simulation software in developing students' CT skills. For instances, simulation software provides unlimited devices and software that students can simply drag, drop and design complex network design. Simulation software also provides visual representation of abstracts which may otherwise be difficult for students to understand (Repenning et al., 2016; Willis & Miertscchin, 2005) thereby helping them to develop their creativity and innovation (Ruiz-Martinez et al., 2013). Simulation software also provides flexibility in that there is no geographical nor hardware restriction therefore students spend more time on problem-solving than being limited with other factors like connecting hardware devices, cabling, etc (Zhang et al., 2012). Many studies (e.g., Ruiz-Martinez et al., 2013; Hwang et al., 2014; Xu, Huang, & Tsai, 2014) have shown students' satisfaction in problem-solving using simulated devices and that students achieve more compared to the use of limited and constrained physical hardware devices. However, these studies did not focus on how simulation software may facilitate students' CT skills. Therefore, this study investigates Computer Networks students' and lecturers' perceptions of the use of simulation software in facilitating the application of students' CT skills.

2.10.2.2 Existing empirical studies

Ruiz-Martinez et al. (2013) conducted a three-year survey study in Spain to explore how the use of simulated environment could enable flexible but versatile approach to students' hands-

on experience when learning advance network concepts. Their study showed that students studying for computer networks perform better when given practical tasks than theoretical tasks. They used simulated environment than real physical device because of their flexibility to design complex networks without limitation to devices and software. The study focused on designing and solving complex computer network problems on simulated environment. Their study found out that students were satisfied and achieved higher scores than they would normally get when using limited and constrained physical devices. However, they did not investigate how simulated environment could facilitate students' CT skills as they were designing and solving complex network problems.

Hwang et al. (2014) conducted a 3-month survey study exploring the effectiveness of collaborative learning through simulated environment. Students were splits into two cohorts: the experiment cohort which conducted their tasks using a simulated environment while the controlled cohort conducted their tasks on normal physical devices with one to one support. They also measured their effectiveness based on higher achievement score. Students using simulated environment achieved more and had better scores than the other cohort. However, their findings did not focus on how simulated environment could facilitate students' CT skills.

The survey study of Xu et al. (2014) which was conducted in the USA, (for 4 consecutive academic years) focused on students' motivation, knowledge, collaboration, creativity, among others on simulated environment. Not much was observed or assessed on CT. From their findings, only 5% of students demonstrated their creativity on the tasks and only 2% of students found the tasks challenging. Their main final assessment was based on higher achievement scores. They did not measure students' CT skills whilst designing their computer networks on simulated environment.

The cited studies have shown the effectiveness of using simulated environment in providing a better platform for students to work on complex computer network design and achieve higher scores. Based on these findings, literature review and the author's own experiences, this study endeavours to explore students' and lecturers' perceptions on the use of simulation software in facilitating the application of students CT skills.

2.10.3 Research Question3: 'How might the use of simulation software facilitate the application of students' computational thinking?'

2.10.3.1 Rationale

Carrying on what has been discussed in Section 2.10.2.2, many studies (e.g., Hambrusch et al., 2009; Liu, Cheng & Huang, 2011; Ruiz-Martinez et al., 2013) have shown how complex network designs (abstracts) can be taught and learnt on simulated environment. One of the key elements of CT is problem solving (Barr & Stephenson, 2011; Kalelioglu et al., 2016). Literature reviewed has shown that simulation software is able to provide the platform that students can design, visualise and solve complex network designs (Musheer et al., 2012; Zhang et al., 2012) thereby motivating their innovation, creativity and critical thinking (Ruiz-Martinez et al., 2013) leading to computational thinking (Voskoglou & Buckley, 2012). As discussed in Sections 2.5 and 2.8 there is no empirical study showing how CT has been applied and developed on students studying on Computer Networks course. However, there are some studies on students studying for Programming which have been cited from Section 2.5. Therefore, to find out how computational thinking can be applied, this study attempts to investigate how might the use of simulation software facilitate Computer Networks students' CT skills.

2.10.3.2 Existing empirical studies

Hambrusch et al.'s (2009) survey study in the USA explored the application of CT using programming in the following subjects: Physics, Chemistry and Statistics. The study found only significant increased achievement and interest of undergraduate learners in CT skills but failed to show how computational thinking helped learners to develop their problem-solving skills. However, the study shows that learners were able to develop their own codes but there is nothing much we know about how this may have helped in developing their CT skills. Besides, their study was conducted on a small scale therefore their results cannot be conclusive.

Using railroad simulation computing game, Liu et al. (2011) conducted a survey study to investigate 117 first year undergraduate students' ability to programming using CT skills. These students were novice in computer programming and were enrolled on Introduction to Computer Science course at a university in Northern Taiwan. The study only found out that the game enhanced students' motivation to learning programming but failed to show how the study developed students' computational thinking.

The following chapter explores on the research methodologies espoused in this study to address the three identified research questions.

CHAPTER 3: METHODOLOGY

3.1 Introduction

This chapter starts by providing a brief overview of philosophical paradigms which underpin the research framework, particularly how my ontological and epistemological stances influence the choice of adopted mixed methodology. This is followed by a critical discussion on methods adopted in this study as well as the rationale and challenges associated with each method. Furthermore, the chapter provides an illustration of a stage-by-stage flow of data collection and analysis that has been followed in this study. Finally, ethical considerations are discussed.

3.2 Rationale for my philosophical research approach

3.2.1 Ontology and epistemology

The philosophical research approach that a researcher espouses assumes a number of factors. Some among others are researcher's view of the world, values, culture, political reasons, policies governing the research area, nature of the research, the kind of research in question, a sponsor's agenda for the research, publisher and many more (Cohen, Manion, & Morrision, 2009). The researcher's philosophical stance will be also driven by particular interest in between his or her understanding of knowledge and the process in how that knowledge is developed (Saunders, Lewis & Thornhill, 2009). For instance, a researcher who is particularly interested in the *what, where* and *when* will approach it differently from a researcher who is interested in the *why* and *how* (Cohen et al., 2009). The two researchers will have different strategies and methods in collecting data. Their point of view of what is important and significant will considerably be different. In other words, their *ontological* and *epistemological* point of views are different in choosing the type of methodologies in addressing their research questions.

Ontology is the nature of reality which "raises questions about assumptions researchers have about the way the world operates and the commitment held to particular views" (Saunders et al., 2009, p.110). Ontology deals with questions which relate to matters of real existence or real action (Guba & Lincoln, 1994), such as how social entities relate with social actors (Saunders et al., 2009). Objectivism and subjectivism are some of the aspects of ontology that a researcher may align himself or herself to understand how knowledge is developed (Saunders et al., 2009). Objectivism is an ontological position that states that social phenomena and their meanings are independent of social actors (Bryman, 2015). For instance, objectivism aligns with a researcher whose interests are the *what, where* and *when.* His or her focus is observable facts or data, typical example is a natural scientist. On another hand, subjectivism, is an ontological position that states that social phenomena and their meanings are constructed within society entity and that are dependent of social actors (Saunders et al., 2009). I, as a researcher, stand on the subjectivism paradigm where my interests are the *why* and *how*. My position is interested in the process and meanings (Cohen et al., 2009).

Epistemology is a philosophical view of how we come to know what is in existence (Saunders et al., 2009). Therefore, epistemologies are concerned with what constitute acceptable knowledge in a particular field of study. This is where we draw research philosophical views like positivism, interpretivism, pragmatism, among others. For instance, positivists work with an observable social reality and their interests are the *what, where* and *when.* They use highly structured methodology to facilitate replication (Gill & Johnson, 2002). Interpretivists, whose interests are understanding differences between humans as social actors (Bryman, 2015; Crotty, 1998). These are interested in the *why* and *how*? Pragmatists are interested in *what works* in order to constitute what is acceptable knowledge (Guba & Lincoln, 1994). I stand as a pragmatist whose interest is to find what works to answer the *why* and *how* questions.

In conclusion, while ontology asserts the nature of reality, the relationship between social entities and social actors, epistemology asserts the *how* we come to know what is in existence. We therefore see the interdependence (Cohen et al., 2009) between ontology and epistemology in developing what is in existence and how we come to know. The symbiotic relationship between ontology and epistemology gives rise to the questions of what methodologies and methods to espouse in collecting data (Cohen et al., 2009; Yeganeh, Su, & Chrysostome, 2004).

Other researchers, for example Hitchcock and Hughes (1995), believe that research is best conducted when ontological assumptions are defined first followed by epistemological assumptions leading to the choice of methodologies and methods espoused. However, Crotty (1998) believes that ontology and epistemology emerge together. The views of Hitchcock and Hughes have shaped my own philosophical point of view that the assumptions of ontology and epistemology respectively give rise to methodologies and methods chosen.

Nonetheless, it is worth noting though that there are some research philosophical frameworks, such as that of Crotty (1998), who believes that objectivism, constructionism, subjectivism are epistemological stances while positivism, post-positivism, postmodernism, interpretivism are 'theoretical perspective stances'. Crotty (1998) argues that theoretical perspective is "our view of the human world and social life within that world, wherein such assumptions are grounded" (p. 7). Crotty further quotes the work of Blaikie (1993, p. 6) arguing that "ontology is the claims or assumptions that a particular approach to social enquiry makes about the nature of social reality". Crotty (1998) further argues that "it would seem preferable to retain the usage of 'theoretical perspective' and reserve the term 'ontology' for those occasions when we do need to talk about 'being'" (p. 11). This definition aligns closely to what Guba and Lincoln (1994) and Saunders et al. (2009) refer to as *ontology*. In contrast to other researchers' framework (e.g., Bryman, 2015; Hitchcock & Hughes, 1995; Saunders et al., 2009), objectivism and subjectivism are placed under ontologies while positivism, post-positivism, and interpretivism

are placed under epistemology in Crotty's (1998) framework. Crotty (1998) argues that a researcher needs to start with their epistemological stance, then followed by *theoretical perspective*, leading to a range of methodologies which give rise to specific methods to use in collecting and analysing data related to research questions. However, considering the symbiotic relationship drawn above between ontology and epistemology and that the two are interdependent according to Cohen et al. (2009) and Yeganeh et al. (2004), my philosophical stance in this study is that ontology and epistemological assumptions are distinct but complementary in helping me to decide the methodology to espouse.

There are several philosophical views underpinning the research philosophical framework. However, the common ones are positivism, post-positivism, interpretivism and pragmatism which are briefly explained in the following section:

3.2.2 Positivism

Positivists, who stand on the positivism spectrum, align themselves with quantitative methodologies (Decrop, 1999). Their ontological phenomenon is that there is one reality (Cohen et al., 2009; Sale, Lohfeld & Brazil, 2002). They believe that a researcher is external to social realities (Saunders et al., 2009). This argument contradicts Crotty's (1998) framework which argues that positivism, constructionism, subjectivism, are epistemological stances. One of the positivist tenets is that where there is a problem, there must be an existence of a solution - this can be a true or false statement as long as it holds the truth value (Saunders et al., 2009). They believe that if two researchers disagree it means one must be wrong, because ontologically there is only one truth (Tashakkori & Teddie, 2003). Positivists' research is dominantly scientific in nature (Kennedy & Lingard, 2006) and that their epistemological approach is deductive, and their research paradigm tend to be quantitative (Decrop, 1999).

They try to prove or refute a hypothesis and so their tenet is to be independent and objective of the research they conduct (Cohen et al., 2009).

However, positivism has been faced with some criticisms and shortfalls. Johnson and Onwuegbuzie (2004) and Mack (2010) argue that positivists seem not to consider the fact that tools or methods espoused are subjectively chosen by human. Interpretation and reporting of results are conducted by human too. All this is ontologically subjective to human involvement and interpretation. Some of the typical examples which are subjectively observed by the positivists conducting a quantitative research are well encapsulated by Johnson and Onwuegbuzie (2004, pp. 15-16):

> Deciding what to study […] choosing specific test and items for measurement, making score interpretations, selecting alpha levels (e.g. .05), drawing conclusions and interpretation based on the collected data, deciding what elements of the data to emphasize or publish and deciding what findings are practically significant...

Consequently, some researchers (e.g. Creswell et al., 2006; Tashakkori & Teddie, 2003) argue that any research conducted is not immune from human involvement neither can it purely be objectively conducted as the positivist claim (Denzin & Lincoln, 2005; Howe, 2004). It was therefore decided that positivism views were neither my stance nor would fit in well with this study because this study involves interpretation of the subjectivism of different human perceptions investigated in this study.

3.2.3 Post-positivism

However, within the positivism framework there is another philosophical approach known as post-positivism. Post-positivism, according to Crotty (1998), does not necessarily mean that there was once positivism and that this has been replaced by post-positivism; rather post-positivism stands on its own but remains and retains a number of the features of positivism. For instance, post-positivists still hold the ontology that there is one reality (Kennedy &

Lingard, 2006). However, post-positivists "contend that the whole truth is never fully apprehendable but is approached progressively through the process of research" (Kennedy & Lingard, 2006, p.102). In essence, although the post-positivists share the main assumptions of positivist they hold relativistic perspective. Their ontological and epistemological stance is that there is an objective world, but knowledge of it is filtered through subjective experience of individuals. Creswell (2014) defines post-positivism as thinking after positivism. Phillips and Burbules (2000) contend that post-positivism challenges the notion that there is absolute truth and that we cannot be absolute positive when we deal with human behaviour and their actions. Crotty (1998) further claims that according to the stance of post-positivism, observer and observed cannot be independent. Their methodological paradigm can either be qualitative, quantitative or mixed methods. Therefore, though the post-positivists retain a number of the features of positivism, their philosophical stances are between positivist and interpretivism (Kennedy & Lingard, 2006). In this study the main interest is to investigate human understanding of CT and perceptions of using simulation software therefore requires human interaction and interpretation to the inquiry. It was therefore decided that post-positivism philosophical approach did not fully fit in well with the objectives of this study therefore was not pursued further.

3.2.4 Interpretivism

This is a philosophical approach to studying people, particularly in social sciences. Crotty (1998) suggest that in human science the main concern of study is understanding humans. The ontological view of interpretivists is that the world depends on many subjective experiences and that it is socially constructed of that world; therefore, it does not exist independent of experiences (Saunders et al., 2009). Their epistemological view is the opposite end of positivists in that there is no possibility of 'objective' knowledge of the world; all we have are different experiences; therefore, the main interest becomes understanding the processes and

meanings of the area under investigation (Cohen et al., 2009). Interpretivists tend to focus on "the details of situation, a reality behind these details, subjective meanings motivating actions" (Saunders et al., 2009, p. 119). Therefore, there are no predefined dependant and/or independent variables but rather focuses on human interpretation (Myers, 1997). Their research paradigm is mainly qualitative and that their approach is inductive in formulating new ideas and theories (Cohen et al., 2009). Subjectivism is well aligned with interpretivism (Crotty, 1998). Although interpretivism fits in well in this study since the research questions explores participants' understanding of CT and their perceptions of simulation software, pragmatism is a better option for the reasons outlined in the following section.

3.2.5 Pragmatism

Pragmatism is a philosophical view whose belief is the use of mixed research (Creswell, 2014). The ontological and epistemological stance of pragmatism is the use of pluralistic approaches to derive knowledge about the problem (Crotty, 1998). Their focus is on the problem rather than on the methods (Creswell, 2014). Pragmatic researchers will use any or all approaches to understand the problem. Their belief is what works to bring solution to the problem (Guba & Lincoln, 1994). Based on Creswell (2014), quoting the work of Cherryholmes (1992) and Murphy (1990), the claims of pragmatists are that they do not commit to one system of philosophy and reality hence the use of mixed methods in their investigation; they are free to choose the methods, techniques and procedures of research which satisfy their needs and purpose to solve a problem; they do not see the world as an absolute unity and that truth is what works at the time (Creswell, 2014).

Pragmatism fits in well in the investigation of this study following the mixed methods approach adopted in addressing research questions. For instance, students online survey (SoS) and lecturers online survey (LoS) were conducted to investigate students' and lecturers'

understanding of CT and their perceptions of how simulation software may facilitate students' CT skills. Online surveys were followed-up with one-to-one and focus groups interviews which were used to triangulate the findings from these methods as means of validating findings. Problem solving tasks were also implemented to address the question of how simulation software may facilitate students' CT. Students provided reflective reports based on their three problem-solving tasks. Students' reflective reports were significant in this study as they provided the raw experiences (Koro-Ljungberg, Cavalleri, Covert & Bustam, 2012) of students' perceptions on problem solving via simulation software. Collectively, results from all these different methods were analysed and triangulated for validity and reliability of results in formulating emerged ideas.

Therefore, concurring with Creswell (2014), methods in this study were chosen based on what works at the time to satisfy the need of the investigation. The use of different methods was adopted with the view of finding out the right answer to the inquiry of this study without being confined to one specific method, technique and procedure of research.

3.3 **Mixed methods approach**

There is substantial literature (e.g., Creswell et al., 2006; Lund, 2012; Mason, 2006; Onwuegbuzie & Johnson, 2006; Pole, 2007; Sale et al., 2002) that argue in favour of the mixed methods approach. The main argument highlighted in the literature is the strength and richness that the mixed methods approach brings to the investigation of the study. According to Sale et al. (2002) and Lund (2012), qualitative and quantitative approaches study different phenomena. For instance, qualitative research emphasises on the meanings of words in data collection and analysis, and a researcher tends to be inductivist, constructivist or interpretivist while quantitative research emphasises on numerical data analysis where the researcher tends to be deductivist, objectivist or positivists (Bryman, 2015). Therefore, mixing the two helps to

produce a complete picture of the area under study. Onwuegbuzie and Johnson (2006) further argue that mixing qualitative and quantitative methods does not exacerbate their weaknesses, but rather enhances and enriches their strength. Lund (2012) also argues that mixed methods can be divergent and contradictory thereby helping in drawing upon further reflection of knowledge necessary to form theories and produce good practice. One of the key purposes for mixed method is for developmental reasons (Johnson & Onwuegbuzie, 2004). For instance, the results of one method can be a source of feeding information to the other method. In this study, SoS and LoS provided information necessary for in-depth one-to-one and undergraduate students' focus group interviews. Problem-solving tasks provided information necessary for postgraduate students' focus group interviews and their reflective reports. Additionally, the strength of one method may overcome the weakness of the other (Pole, 2007). In this study, the weaknesses of surveys (i.e. not providing prompts for participants to give more rich data), were enhanced by using one-to-one and focus group interviews which provided in-depth dialogue to and by the participants to clarify surveys' responses (Creswell, 2014). Johnson and Onwuegbuzie (2004) argue that mixed methods can add insight and understanding that might be missed when only a single method is espoused. In this study, different methods i.e. surveys, in-depth interview, problem-solving scenario, focus groups, observation and reflective reports were applied for triangulation and complementarity.

3.4 **Application of mixed methods in this study**

The application of mixed methods in this study was based on the nature of research questions as stipulated in Table 3.1.

Table 3.1

Research Questions

Research question 1	What are Computer Networks students' and lecturers' understanding on computational thinking?
Research question 2	What are Computer Networks students' and lecturers' perceptions of the use of simulation software to facilitate the application of students' computational thinking?
Research question 3	How might the use of simulation software facilitate the application of students' computational thinking?

They were a few considerations which had to be made when applying mixed methods (Bryman, 2015; Creswell, 2014) in this study. The first consideration was the approach in which mixed methods could be applied. Mixed methods can be conducted, either as dominant qualitative methods with less quantitative methods or vice versa or both qualitative and quantitative methods conducted at equal weight (Bryman, 2015; Creswell, 2014; Johnson & Onwuegbuzie, 2004). In this study, dominant qualitative methods were conducted with less quantitative methods as illustrated in Table 3.2

Table 3.2

Mixed Methods Approach in this Study

Quant ⟶ QUAL	QUAL
1. Questionnaires (SoS, LoS): • CLOSED-ENDED questions • Open-ended questions 2. In-depth Interview	• Observation (video) • Reflective report • Focus group Focusing on: (i) Abstraction (ii) Decomposition (iii) Generalisation
Mainly for research questions *1 & 2; partially for research question 3*	*Mainly for research question 3; partially for research questions 1 & 2*

Note: Table 3.2 shows the mixed methods adopted in this study. The left column shows the initial open and closed ended questionnaires via online and then followed by in-depth interviews. Letters in capital letters denote the dominant element and the arrow (⟶) denotes sequential data collection. It is only in this column where quantitative method was mixed with qualitative method. To the left of the column, it shows series of qualitative methods adopted thereafter in this study

The quantitative methods via the use of Likert scale on SoS and LoS were included as part of the initial investigation on students' and lecturers' understanding of CT and their perceptions of how simulation software may facilitate the development of students' CT skills. The SoS and LoS were also purposefully helpful in selecting participants for qualitative data via the use of one-to-one and focus group interviews (Creswell, 2014). See Figure 3.1 illustrating how SoS and LoS fed into one-to-one and focus group interviews.

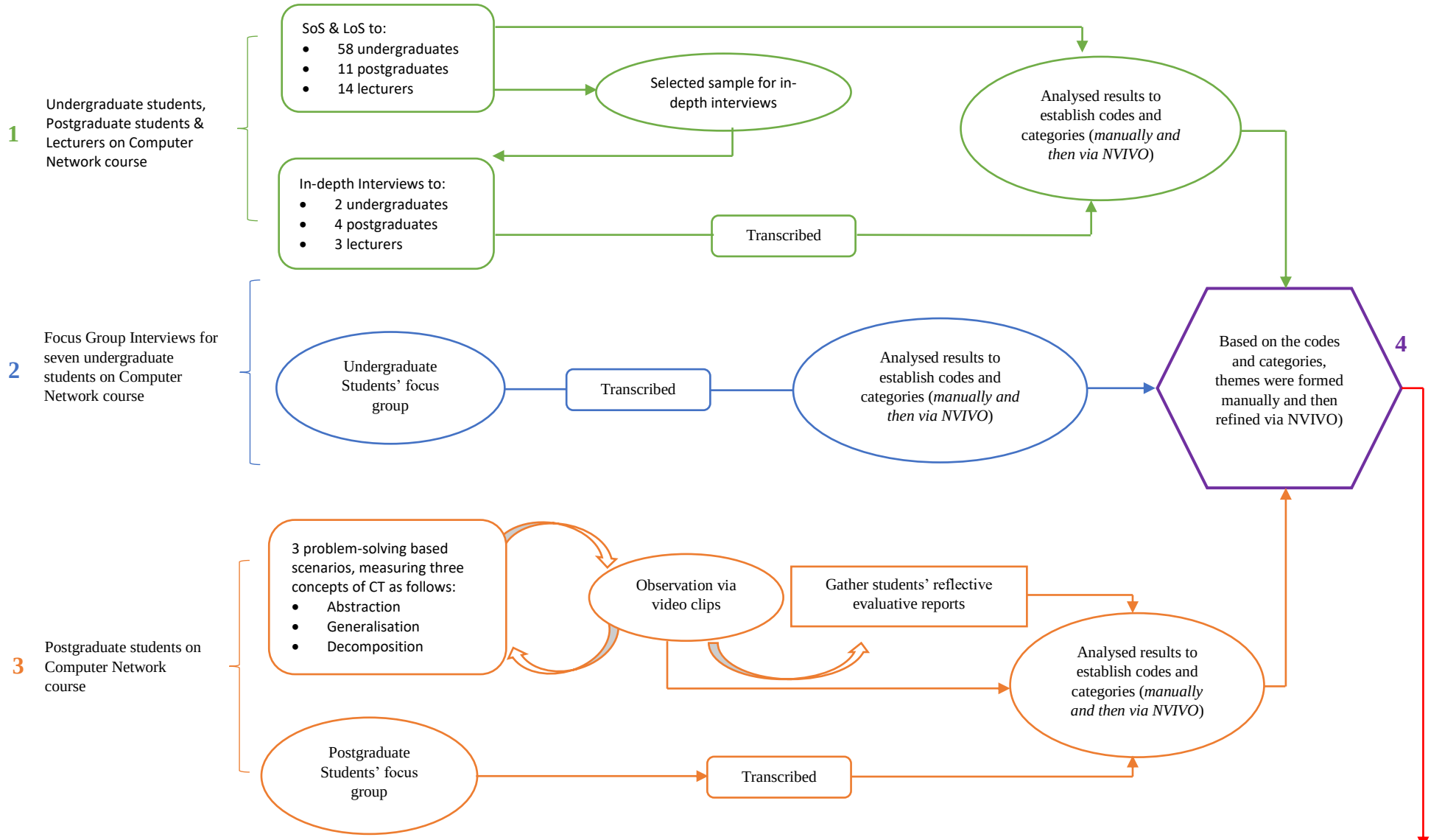


Figure 3.1. Flow of data collection and analysis.

Note: The numbers indicate the sequential order in which data was analysed. Stage 1, data was collected and analysed sequentially from surveys to one-to-one and focus group interviews and were both analysed feeding into stage 4. Stage 2, only undergraduate students were involved in the focus group interviews which were compared with results from stage 1, feeding into stage 4. Stage 3, postgraduate students were involved in problem-solving tasks, then provided reflective reports and participated in focus group interviews. All the data in stage 3, was analysed and results were fed into stage 4. Stage 4 involved consolidation and analysis of all results gathered from stages 1, 2 & 3 to form refined codes and themes

The second consideration in the mixed methods approach was whether the two methods needed to be conducted concurrently or sequentially (Johnson & Onwuegbuzie, 2004). In this study, quantitative and qualitative methods were applied sequentially to triangulate quantitative results with qualitative results (see the illustration on Table 3.2). Initially, quantitative data collection via surveys (with a mixture of open- and closed-ended questions) were conducted. For example, SoS and LoS were conducted on a larger scale, targeting a bigger sample of students and lecturers studying and teaching on Computer Networks course respectively. Creswell (2014) argues that sequential mixed method helps qualitative data to shade more light based on the quantitative results. Usually the initial stage involves collecting data via surveys and the results from surveys form the basis for qualitative data collection that explains the surveys' responses. This approach fitted in very well with this study as explained above and illustrated in Figure 3.1, stage 1.

The third consideration was the characteristics of individuals to be sampled for qualitative data analysis. In a mixed methodology paradigm, Creswell (2014) argues that the individuals to be sampled for qualitative data collection and analysis should be from those who participated in the quantitative data collection because the intent of qualitative data analysis is to follow up the quantitative results which explores the results in more depth. In this study, individuals who were sampled to participate in the qualitative data collection were from those who participated in the SoS and LoS following the argument of Creswell, (2014). After the initial stage of data collection via surveys, the rest of the methods were qualitative approaches as illustrated in Table 3.2. The process of data collections and analysis has been illustrated in Figure 3.1, stages 1, 2, 3 and 4.

The fourth consideration in mixed method was to decide at which phase of methodology should a researcher conduct the "mixing" (Johnson & Onwuegbuzie, 2004) and how was the mixing done (Creswell, 2014). In this study, Table 3.2 shows how quantitative data and qualitative

data was sequentially mixed. And, the table shows the predominant use of qualitative data that was used for the rest of the study. Figure 3.1 illustrates how mixed methods were conducted and how the data was collected and analysed otherwise detailed discussion has been provided in Section 3.8.

3.5 **The university setting**

The university where data were collected was from the Department of Computing that has six subject group areas, namely: *Applied Computing, Business Information Systems and Technology, Games and Information Systems, Software Engineering and Computer Science, Computer Systems and Networks, and Cyber Security*. This study focused on only Computer Systems and Networks subject group area for reasons provided in Section 1.5 of the Introduction chapter.

As an applied university since 1992, the focus has always been developing students' practical and technical skills ready for employment. For instance, students' learning time-table is scheduled in such a way that one hour is allocated for teaching theories of computer networks, four hours are scheduled for developing students' practical skills in designing and troubleshooting computer networks with the support of tutors. The labs are then open for 24 hours for students to use them to further develop their own independent practical skill. Therefore, the curricula of courses in the Department of Computing, e.g. Computer Networks course have been designed to develop students' practical and technical skills, such as designing network infrastructure, supporting and managing network systems, diagnosing and repairing malfunctioning network devices or systems rather than focusing on studying and developing theories for further research. There are several simulation and emulation software which the Department of Computing use in teaching students. For example, in Computer Networks course, some of the simulation and emulation software used are GNS, OPNET (currently

known as Riverbed Modeler), Cisco VIRL and Cisco Packet Tracer. However, the dominant used simulation software by Computer Networks students is Cisco Packet Tracer because it is linked to the Cisco curriculum which is embedded in their course to practice network design before working on the real Cisco physical hardware devices. The simulation software also helps students to work on online min-tasks via the Cisco academy website (www.netacad.com). As such, using Cisco Packet Tracer simulation software and physical hardware devices in working out students' practical tasks and coursework have been lecturers' predominant elements in teaching and assessing students since the university changed from polytechnic to applied university. Although students use simulation software to practice their network designs, Cisco physical hardware devices are the main equipment lecturers use to assess students' technical skills in network design and troubleshooting.

Furthermore, to enforce practical and technical skills needed for employment, the curriculum for Computer Networks course follow the Cisco, Microsoft, Linux and Virtualization curricula. These curricula are designed to train employees who may want to upskill themselves whilst working in the field of computer networking. However, since 2015, the ethos of the university shifted to combining the development of students' practical and research skills. The current vision of the university is to become an outstanding applied university that produces outstanding calibre of students who are well equipped with their practical skills for employment as well as research skills that they can further design and produce network systems.

3.6 **Sampling strategies**

There are two main categories of sampling strategies namely: *probability and non-probability sampling* (Bryman, 2015; Creswell, 2014; Saunders et al., 2009). Probability sampling is a random selection of participants that gives the same chance for everyone on that population to be sampled. This provides a representative sample that reflects the total population (Saunders

et al., 2009). Probability sampling aims at keeping sampling error (i.e. the error due to difference between sample size and population size) to a minimum (Bryman, 2015). Stratified random sampling is one of the examples of probability sampling. Stratified random sampling is the selection of specific characteristic (e.g. female and male, undergraduate and postgraduate students) reflecting their true proportional representation of the population under research study (Fowler, 2002). Bryman (2015) argues that if a researcher can stratify his/her data collection by faculty and gender or undergraduate and/or postgraduate students then stratified sampling is feasible. Although stratified random sampling would have been a better sampling choice to provide generalisability of findings, it was a challenge in this study to gather a true proportional representative of the total population of targeted Computer Networks students and lecturers (see Table 3.3 and the discussion that follows):

Table 3.3

Sampled Computer Networks Students and Lecturers in the Case-study Institution

Number of undergraduate students		**Participated in:**	**Percentage (%)**
Total population	110	SoS = 58	53
From those participated in SoS	58	FG interviews = 7	12
From those participated in SoS	58	1-2-1 interviews = 2	3
Number of postgraduate students		**Participated in:**	**Percentage (%)**
Total population	20	SoS = 11	55
From those participated in SoS	11	FG interviews = 6	55
From those participated in SoS	11	1-2-1 interviews = 4	36
Number of lecturers		**Participated in:**	**Percentage (%)**
Total population	20	LoS = 14	70
From those participated in LoS	14	1-2-1 interviews = 3	21

The process of data collection was conducted at the time when most of the students were preparing for their various assessments. However, based on the research questions pursued in

this study a non-probability sampling known as *purposive sampling* was espoused. Purposive sampling is an idea that unit of sampling depends on the research question (Bryman, 2015; Creswell, 2014) and objectives (Saunders et al., 2009). In this study, the research questions mainly focused on Computer Networks students' and lecturers' understanding of CT, their perceptions and how may the use of simulation software facilitate the application of students' CT. As such, samples to participate in interviews were purposively based on Computer Network students and lecturers learning and teaching on Computer Networks course respectively. Therefore, from the six subject group areas in the computing department as explained in Section 3.5 only one subject group area (*Computer Systems and Networks)* was sampled, giving a fraction of 1/6 population size. Undergraduate and postgraduate students and lecturers were sampled as shown in Table 3.3 and the summarised explanation is provided below:

For the sake of increasing chances for a better response rate (Saunders et al., 2009; Bryman, 2015), initially all 110 undergraduate and 20 postgraduate students were invited to participate in students' online survey (SoS). However, there were some challenges in collecting the true proportional sample size since not all targeted students participated in SoS. For instance, as shown in Table 3.3, only 58 of 110 undergraduate students (53%) and 11 of 20 postgraduate students (55%) participated in SoS. From the total of 69 undergraduate and postgraduate students who participated on SoS, only 12 put down their emails addresses to participate on one-to-one interviews. However, when they were contacted by their emails only a total of six (i.e. two undergraduates and four postgraduates) accepted and participated on one-to-one interviews. Seven of the 58 undergraduate students (12%) and six of 11 postgraduate students (55%) participated in the focus group as discussed in Section 3.8.2. These small numbers could be because most of the students were busy preparing for their examinations. All 20 lecturers within the subject group area of Computer Systems and Networks were also invited to

participate on lecturers' online survey (LoS). However, 14 of 20 lecturers (70%) accepted an invitation and participated. Likewise, lecturers were requested to put down their names at the end of their LoS if they would like to participate on one-to-one interviews. From the 14 lecturers, only four lecturers put down their email addresses. However, when they were contacted three accepted and participated on one-to-one interviews.

Additionally, there is another non-probability sampling which covers a broad spectrum of sampling which includes, *convenience sampling, snowball sampling and quota sampling* (Bryman, 2015; Saunders et al., 2009). According to Creswell (2014), convenience sampling is less desirable because it is based on the convenience and availability of the researcher. Therefore, Bryman (2015) argues that it is not possible to generalise findings because their findings do not have a true proportional representation of the population. However, for the sake of maximising possible response rate (Saunders et al., 2009) convenience sampling strategy (Saunders et al., 2009) was also espoused in this study. At the point of data collection, I had no access to undergraduate students, because I was not teaching them, therefore convenience sampling strategies were adopted by narrowing down the focus to only postgraduate students I was teaching. Therefore, the available six postgraduate students on Computer Networks course participated in both problem-solving tasks and focus group interviews. The focus group interview lasted for forty-five minutes.

3.7 **Characteristics of the participants**

The main participants on this study were undergraduate and postgraduate students who were enrolled from different academic and cultural backgrounds to study on Computer Networks courses. Computer Systems and Networks subject group area recruits a total average of 110 undergraduate students and 20 postgraduate students. In this study, the participants were a mixture of students who progressed from college studying for IT, accredited by the Business

& Technology Education Council (BTEC), sixth form studying for A-levels, polytechnic studying undergraduate degree in IT courses, and mature students studying for Access to Higher Education. Most of the undergraduate students progressed from colleges where the focus of their teaching and learning style is vocational (the emphasis is on more practical than theories) (Elliot, 1996). The postgraduate students were ex-employees who came to gain an academic qualification. In total, 58 undergraduate students (46 males, 11 females and one undisclosed gender) and 11 postgraduate students (11 males and no female) participated in the SoS. Some of these students furthermore participated in other data collection methods, such as focus groups and problem-solving tasks as outlined later in this section. The characteristics of these participants have been provided in Table 3.4.

Table 3.4

Characteristics of Student Participants (N = 69)

Gender	**Actual Number**	**Percentage (%)**
Female	11	16
Male	57	83
Prefer not to say	1	1
Level of Course		
Undergraduate	58	84
Postgraduates	11	16
Progressed from		
College studying for BTEC	27	39
College studying for A-levels	7	10
Sixth form studying for A-levels	17	25
Others	18	26

The other participants were lecturers in the Computer Systems and Networks subject group area. The overall ratio of male to female lecturers in the Computer Systems and Networks subject group area is 18 to 2. In this study, 14 of 20 lecturers (12 males and two females) who predominately lecture on the Computer Networks course participated in the LoS. Three of these

lecturers (only male) participated in a 60-minutes one-to-one interview. None of the female lecturers put down her email address via LoS to be interviewed further. Most of the lecturers involved in this study (particularly those without a doctorate degree as shown in Table 3.5) have extensive practical industrial experiences, such as designing network infrastructure, managing and maintaining network systems, diagnosing and repairing malfunctioning network systems gained from industry. The characteristics of these participants are provided in Table 3.5.

Table 3.5

Characteristics of Lecturer Participants (N = 14)

	lecturers	**Percentage (%)**
Teaching Experience in HE (years)		
< 2	2	14
2 to 4	1	7
5 to 7	2	14
8 +	9	64
Computing Industrial Work experience (years)		
< 2	2	14
3 to 5	7	50
6 to 8	2	14
9 +	3	21
Highest Qualification		
Doctorate degree	6	43
Master's degree	6	43
Bsc. degree	2	14

All lecturers without doctorate degree qualifications have extensive computer networks industrial experiences and professional qualifications, such as CCNP (Cisco Certified Network Professional) which enrich the skill set needed on this course.

All participants have been given pseudonyms for confidentiality. Undergraduate and postgraduate students in this study have been referred to as UGstudX and PGstudX

respectively, where X represents a random number. For instance, undergraduate participants on one-to-one and focus group interviews range from UGstud1 to UGstud9, while postgraduate participants range from PGstud1 to PGstud6. Focus groups for undergraduate and postgraduate students have been referred to as UGFG and PGFG respectively. Lecturers have been referred to as LecX where X represents a random number too. All comments from students' and lecturers' surveys have been referred to as SoS or LoS respectively.

3.8 Types of data collection methods

3.8.1 Students' and lecturers' online surveys

Students' and lecturers' online surveys were initially conducted to primarily address the first and second research questions. As discussed in Section 2.5 many scholars (e.g., Barr & Stephenson, 2011; Bocconi et al., 2016; Grover & Pea, 2013) have advocated the application of CT at school levels with the assumptions (e.g. Wing, 2008) that at university level, students already understand and are able to apply the concepts of CT. Therefore, at this initial stage, data was collected across a wider range of students (*N=69*) and lecturers (*N=14*) to ascertain students' and lecturers' understanding of CT at university level and their perceptions of using simulation software to facilitate the application of students' CT skills.

There were two types of survey items – one for the students (SoS) and the other one for lecturers (LoS). Detailed information about SoS and LoS are provided in Appendices 3.01 and 3.02 respectively. Both survey items had similar questions for easy comparison of inferences. The questions focused on three main fundamental areas, namely: the general personal information of participants, understanding of their knowledge and application of CT, and finally their perceptions and application of simulation software to facilitate students' CT skills. These three areas with their corresponding question numbers from SoS and LoS have been provided in Table 3.6.

Table 3.6

Structure and Sections of Online Surveys

Main area of inquiry	**Specific interest of inquiry**	**The specific No. of questions from SoS**	**The specific No. of questions from LoS**
General personal information	The characteristics of the participants	1-3	1-4
Understanding participants' knowledge and application of CT	(i) Their general understanding and knowledge of CT	5-7	6-8
	(ii) Using Likert scale, finding out their application of CT	8-12	9-13
	(iii) using open-ended questions	4, 13, 14,	5, 14-16
Participants' perceptions and application of simulation software to facilitate CT	(i) Their general perceptions	15-16	17-18, 20
	(ii) using open-ended question	17	19
	(iii) Using Likert scale, finding out their application of simulation software to facilitate CT	18-21	21-23

There were five questions via SoS (Q4, Q5, Q7, Q10, & Q11) and LoS (Q5, Q6, Q8, Q11 & Q12) which were adapted from the study of Yadav et al. (2011) who investigated teachers' understanding of CT at primary and secondary school levels (see Section 2.10.1.2 of the Literature Review chapter). These questions were around *understanding participants' knowledge and application of CT* (see Table 3.6 and Appendices 3.01 & 3.02). Apart from those five questions adapted from Yadav et al. (2011), all questions were designed from scratch to suit the investigation of this study. These questions were piloted through expert lecturers in Computer Systems and Networks subject group area, students and the supervisors before giving them to participants. Additionally, all data collection methods in this study went through the ethical committee at University of Reading's Institute of Education where the study was administered as well as at the Department of Computing at the University were data was

collected. The feedback after piloting questionnaires and going through ethical committee shaped the final questionnaires sent via SoS and LoS. For instance, there were more than 40 questions for both SoS and LoS however after the piloting process, they were trimmed down to 21 for SoS and 23 for LoS. Students, lecturers and supervisors felt that they were many questions and would easily put off participants leading to giving false responses. Additionally, there were some questions which were repetitive which ended up giving the same results. The questions were also trimmed to focus on the research questions – covering students' and lecturers' understanding of CT and their perceptions of using simulation software to facilitate students' CT skills (see the first column of Table 3.6). The same piloting process via lecturers, students, supervisors and ethical committee was conducted for all other data collection methods espoused in this study.

The online surveys were chosen because they were easy, cheaper and quicker way of collecting information from participants. Above all, online surveys provide an instant way of collecting and presenting data (Bryman, 2015). It was also easy to get a quick general overview of students' and lecturers' understanding of CT and their perception of using simulation software in facilitating students' CT skills. Furthermore, most of the targeted participants in this study were technologically adept and that almost everything they do (i.e. learning and assessments) is done electronically therefore participants found it easy to complete the survey.

However, online surveys were not able to capture in-depth information. For instance, there were no instances where participants were prompted for more answers based on their response and that participants were not able to probe the interviewer for more clarity based on their responses. These are some of the limitations of online surveys (Bryman, 2015). However, to avoid the problem of prompting and probing from participants, the questions were designed in a way that they were easy to follow and answer. For instance, there were fewer open-ended

questions than closed-ended questions, and the surveys were scheduled to last for only 10 to 15 minutes.

However, the results obtained from online surveys probed some further area of inquiry (Johnson & Onwuegbuzie, 2004) leading to sampling participants for one-to-one in-depth interviews. To maximise credibility of findings, the results obtained through surveys were compared and analysed with those results from interviews (see Figure 3.1). Additionally, some of the results from the survey also helped in addressing the third research question (see Table 3.2).

3.8.2 One-to-one and focus groups interviews

Following on from SoS and LoS, two focus groups consisting of seven undergraduate students and six postgraduate students were conducted separately (see Appendix 3.03 for sample questions). Additionally, six students (four postgraduates and two undergraduates) and three lecturers participated in one-to-one interviews (see Appendices 3.04 and 3.05 for sample questions). One-to-one and focus groups questions were structured to follow up with the questions posed from SoS, therefore similar but open-ended questions were formulated. Additionally, there were some questions which emerged during one-to-one and focus group interviews which furthermore enriched data collection. The focus group conducted for the undergraduate students covered their understanding of CT and their experiences of using simulation software when designing computer networks in their day-to-day practical lab activities. Furthermore, all students who participated in undergraduate focus group had just come back from their work placement year and were enrolled on the final year of their degree programme. On the other hand, the focus group conducted for the postgraduate students covered their understanding of CT, experiences and perceptions of using simulation software based on the three problem-solving tasks as explained from Section 3.8.3. All those students who participated in postgraduate focus group interviews were ex-employees who were enrolled

on the course to gain an academic qualification. Therefore, all students who participate in both focus group interviews had industrial and practical skills in computer networks.

Most of the challenges encountered via surveys were overcome by the one-to-one and focus group interviews. For instance, the problem of prompting and probing that were faced on SoS and LoS were made much clearer because of the presence of the interviewer (*myself*). Participants were able to ask for clarity of the questions before attempting an answer. The interviewer was also able to probe more questions that were salient to respondents. For online questions, according to Bryman (2015), a researcher is not sure who answered the questions, there is greater risk of missing data since questions are 'fixed', and there are higher chances for lower response rates. All these challenges were overcome by interviews which were conducted on face-to-face between the interviewer and interviewees.

Nonetheless, in-depth interviews are not immune from challenges, for example, interviews can be too long causing the participant to give false information to get away with the process. On another hand, participants have no much time to think about their responses which may end up giving responses which may not really reflect their subjective view (Creswell, 2014). The presence of the interviewer can prevent honest opinion from respondent. However, in this study, to overcome such challenges some participants were interviewed twice, (via one-to-one and focused group interviews) which is normal in qualitative interviews (Bryman, 2015). The process of different types of inquiries helped to elicit participants' subjective views which were triangulated if their views were congruent as illustrated in Figure 3.1.

3.8.3 Problem-solving tasks

After the investigation of students' and lecturers' understanding of CT, six postgraduate students, who also participated on SoS, one-to-one and focus group interviews were furthermore investigated using three problem-solving tasks via simulation software to address

the third research question (see Section 3.6 which discusses the sampling strategies and the rationale for choosing these students). These six postgraduate students progressed from different undergraduate courses, from different institutions and countries with varied computer networking knowledge, experiences and skills. They were all enrolled on Msc Professional Networking course which covered the following modules, among others: *Local area network (LAN) and design implementation, Wide area network (WAN) and design implementation,* and *Network security*. These three modules are designed to develop students' practical skills in designing and implementing LANs, WANs and securing network infrastructure. The modules were chosen because they cover aspects of some core concepts of CT, such as abstraction, decomposition and generalisation when designing and implementing network infrastructure, therefore they fit in well with the inquiry of this study.

The problem-solving tasks were fictitious but related to specific and important curriculum topics requiring students to develop the skill of designing and troubleshooting network systems. Therefore, the tasks required students' ability to demonstrate their problem-solving skills in designing and implementing network infrastructure. Literature (e.g., Aho, 2012; Barr & Stephenson, 2011; Voogt et al. 2015) has shown that problem-solving tasks give the ability to measure students' reasoning and thought processing thereby providing a better instrument in studying students' CT skills (Wing, 2008). Students worked on their problem-solving tasks for a period of six weeks per each task between the teaching-and-learning period of January to June 2017. Students used Screencastomatic software desktop capture program (https://screencast-o-matic.com) to capture video recording of every problem-solving activity they were working on (see sample video screenshots captured in Appendices 4.7 and 4.8). After each problem-solving task, students were requested to write an individual reflective report. Students were briefly given an outline on the areas of CT to focus on when providing their reflective reports (see the sample form provided in Appendix 3.06). However, although

students were explained about the basic overview of the elements of CT, such as abstraction, decomposition and generalisation, they were not taught how the concepts are applied. The information was provided to students to help them provide appropriate reflective feedback based on the form provided. Therefore, students had basic overview of what is involved in CT but not taught. The detailed discussion about the three tasks have been provided from Section 3.8.3.1.

3.8.3.1 Problem-solving task 1

On their first problem-solving task, students were requested to redesign (normally known as *reverse engineering*) the entire enterprise network infrastructure (topology) based on the routing outputs (see Appendix 3.07). Routing outputs show the networks that are directly connected and those networks which can be accessed remotely. Using routing protocols, routers are able to advertise (populate) networks which are directly connected to them and able to share those networks advertised by their neighbouring routers (Empson, 2009). This helps to establish the map of the entire enterprise network infrastructure. As briefly noted, reverse engineering is the ability to trace back the advertised networks so that the entire network infrastructure can be re-built. In this task, not only were students requested to rebuild the entire enterprise network infrastructure but were also required to troubleshoot network design problems which were embedded in the routing outputs and provide appropriate solutions with recommendations. To verify if students had redesigned the correct enterprise network infrastructure, they were asked to show their routing outputs which matched with those as shown in Appendix 3.07 including the corrected errors.

3.8.3.2 Problem-solving task 2

In their second problem-solving task, students were requested to design another enterprise network infrastructure from scratch based on the requirements given on the scenario (see the

scenario in Appendix 4.08). This was another challenging task as it required students' prior knowledge and skills on LAN design and implementation concepts to produce an enterprise network infrastructure. The task involved 4 simulated major cities connected to an Internet Service Provider (ISP). Each city represented a LAN. The links connecting to all these cities (*LANs*) and an ISP represented WAN. There were some specific requirements which students were required to meet, for instance, the provision of required bandwidth, throughput, response time, accessibility of users' appropriate resources, confidentiality, integrity, use of different IP (Internet Protocol) addressing scheme (IPv4 and IPv6) and use of appropriate routing protocol to facilitate network communication.

3.8.3.3 Problem-solving task 3

The final task involved distinct set of tasks which required the skills of combining the design of LAN and WAN to implement a secured enterprise network infrastructure. In this task, the key element of assessment was students' ability to design and implement a secured enterprise network infrastructure. Besides their individual reflective reports from task one and two, students were asked to write an individual overall reflective report encompassing all three problem-solving tasks, highlighting their thought processes and strategies in coming up with their solutions and recommendations (see Appendix 3.09 for a sample reflective form). Furthermore, each student was asked to record video clips showing how they solved their problems and demonstrated their solutions. Table 3.7 provides a summary of the key core concepts which were measured against each task.

Table 3.7

Summary of Key Core Concepts of CT Measured via Problem-Solving Tasks

Problem-solving task	**Concepts of CT abstraction (AB), decomposition (DE), and generalisation (GE)**	**Approaches to investigate the core concepts of CT based on literature review (e.g. Angeli et al., 2016; Bocconi et al., 2016; Csizmadia et al., 2015; Grover & Pea, 2013)**
Task 1: Reverse engineering	AB, DE, GE	Understanding and interpreting routing outputs to design a network infrastructure (AB). Breaking down smaller tasks, such as working on each LAN, Vlan, which are connected to form WAN (DE). Identifying similar connections of routes, IP addresses, subnet mask and routing protocols in solving errors (GE).
Task 2: building a network infrastructure from given requirements	AB, DE, GE	Breaking down the complexity of given requirements by working on each branch before connected each branch to form the entire WAN infrastructure (DE). Using previous knowledge in designing and troubleshooting LANs (from task 1) should give them a clue to identify patterns of connections, similarities in building up a network infrastructure (GE). Hiding some technical information but important details such as IP addressing schemes, routing protocols, network services such as DHCP, DNS, PPP and Frame relay protocols (AB).
Task 3: designed a secured network infrastructure	AB, DE, GE	Securing each VLAN, LAN per branch before connecting with other branches (DE). Using previous knowledge in designing and troubleshooting LANs (from task 1 and 2) should give them a clue to identify patterns of connections, similarities in building up a secured network infrastructure (GE). Hiding some technical information but important details such as IP addressing schemes, routing protocols, network security protocols (AB).

3.9 Data Analysis

Thematic analysis is one of the commonly used approaches for qualitative data analysis (Cohen et al., 2009). In this approach, through consistent reading and re-reading of data collected, codes, categories, themes were developed. Table 3.8 provides detailed explanation of thematic analysis and how it was adopted in this study. Bryman (2015) argues that "in qualitative data analysis there is a constant interplay between conceptualization and reviewing the data" (p. 589). Therefore, the stages explained below may not necessary follow that order in all studies, some stages may merge.

Table 3.8

Thematic Data Analysis in This Study

	Stages of thematic analysis adopted from the work of (*Bryman, 2015 p. 587-288*)	**What is involved in each stage and how were these stages applied in this study**
1.	*Read through at least a sample of the materials to be analysed*	This is the stage where a researcher needs to be acquainted with data collected, whether a sample of the material or the whole data. In this study, at this stage, as a process of familiarization of data collected, all data collected from SoS, LoS, interviews and reflective reports were read through for several times.
2.	*Begin coding the materials*	At this stage the researcher develops codes as he/she reads through material. Since this is the initial stage of coding, proliferation of codes is inevitable. In this study, at this stage, there were initial open codes which were generated. Some of the codes were mapped to the predetermined themes based on the core concepts of CT while others were 'hanging' around to be linked to other codes to form themes in the third stage.
3.	*Elaborate codes into themes*	At this stage, a researcher reduces the number of codes by consolidating similar codes into respective themes. In this study, at this stage, codes were broadly consolidated into either predetermined themes or emerged themes, waiting for a further refined process of allocating codes.
4.	*Evaluate the higher-order codes or themes*	There are two points of views at this stage. Other researchers look at combining codes from the third stage into higher-order-code while others seek to develop sub-themes. In this study, at this stage, sub-themes were developed from predetermined themes and emerged themes
5.	*Examine possible links and connections between concepts and/or how the concepts vary in terms of features of the cases*	A researcher may decide to link the concepts with the themes and sub-themes developed. In this study, there were three key concepts of CT, namely *abstraction, decomposition* and *generalisation* which were linked to the predetermined themes and sub-themes developed. Otherwise emerged themes were linked to participants' own understanding of CT and perceptions of using simulation software to facilitate the application of CT skills.
6.	*Write up a compelling narrative about the data*	A researcher writes up about the meaning and interpretations of the themes developed to make sense to the audience/readers In this study, there is a detailed discussion about all themes in the Discussion and Analysis chapter

Strauss and Corbin (2008) argue that themes can be formed based on the terms used in the existing theories or literature of the area of research. In this study they were some predetermined themes (*abstraction, decomposition and generalisation*) which were deliberately chosen based on the core concepts of CT. These became the primary themes in trying to find out students' and lecturers' understanding of CT, their perceptions of using simulation software to develop students' CT skills, and finally investigated how may simulation software facilitate the application of CT using problem-solving tasks. The second set of themes were those which emerged outside the predetermined themes, e.g. *problem-solving approaches* and *algorithmic thinking* (see the Discussion and Analysis chapter).

3.10 Coding process

There were two stages for coding data in this study. In the first stage all data collected from SoS, LoS, one-to-one and focus group interviews were printed off and read through several times to identify codes and themes as explained in Table 3.8. The identified codes and themes were tabulated as shown in Appendix 3.10. The second stage of data coding involved the use of a qualitative data analysis software known as *NVivo* (see Appendix 3.11 with some samples of coding in NVivo). NVivo software, as in any computer-assisted qualitative data analysis software (CAQDAS), does not help in decision making about coding and/or interpretation of textual material (Weitzman & Miles, 1995) or retrieving all the coded data, but rather the researcher must read through, understand and analyse all the necessary data to be able to code and interpret it (Bryman, 2015). Therefore, in this study the first stage when all data collected were printed off and used pen and paper to fully familiarise with data was essential for the second stage of coding data. Lewins and Silver (2009) argue that CAQDAS facilitates qualitative data analysis study by providing structure, consolidation and organisation of data in one place; ability to explore the data by searching and interrogating data (e.g. using queries);

graphical and tabular presentation of output and project managements. In this study, all data from the first stage of coding were input into NVivo software to easily refine codes, themes and subthemes. Besides refining codes and themes, NVivo software was helpful in creating mind mapping, coding comparison, code book, and it was easy for word frequency search. Concurring with Lewins and Silver (2009), NVivo also helped in structuring, organising and consolidating codes and themes which resulted into producing graphical representation of data analysed.

3.11 Reliability and validity of methods

Reliability and validity are terms often used in research, but their meanings and interpretations are often applied differently in quantitative and qualitative approaches (Golafshani, 2003; Lincoln & Guba, 1985). For instance, reliability and validity in quantitative research are essential tools which are used to measure the extent at which results are consistent (reliability) and accurate (validity) over a period of time (Creswell, 2014; Golafshani, 2003; Lincoln & Guba, 1985). However, in qualitative research, reliability measures the extent to which the research quality can "generate understanding" (Stenbacka, 2001, p. 551); and can persuade the audience to pay attention to (Lincoln & Guba, 1985). Validity in qualitative research is the authenticity, goodness, adequacy, trustworthiness of the study (Creswell & Miller, 2000). Furthermore, other scholars (e.g., Leung, 2015) argue that validity in qualitative study is the appropriateness of tools, process, sampling, methods adopted, and conclusions made in answering research questions. Validity is affected by the researcher's perception of the study of his/her choice model or assumptions made (Creswell & Miller, 2000).

The use of triangulation in mixed methods to corroborate findings (Golafshani, 2003; Mathison, 1988) can validate the findings in qualitative research (Patton, 2002). Many scholars (e.g. Golafshani, 2003; Lincoln & Guba, 1985; Onwuegbuzie & Johnson, 2006) argue that in

qualitative research, validity has been more contentious and therefore many terms, such as 'rigour' and 'trustworthiness' have been emerged. Bryman (2015) and Rolfe (2006) argue that the contentious of reliability in qualitative research could be as a result of the ontological phenomena that qualitative research has multiple realities. In view of that, scholars (e.g., Graneheim & Lundman, 2004; Guba & Lincoln, 1994; Onwuegbuzie & Johnson, 2004; Rolfe, 2006; Sandelowski, 1993) have argued that reliability and validity in qualitative research are treated as the *trustworthiness* of the research procedure.

Trustworthiness is the criteria for assessing a qualitative study to achieve *credibility, dependability* and *transferability* of findings through the steps of research procedure (Graneheim & Lundman, 2004; Lincoln & Guba, 1985). Credibility is the consistence (reliability) and accuracy (validity) of research findings which can be auditable, verified (confirmed) and transferable (Crang & Cook, 2007; Creswell, 2014; Guba & Lincoln, 1994). Dependability is "the degree to which data change over time and alterations made in the researcher's decisions during the analysis process" (Greneheim & Lundman, 2004, p.110). Transferability shows the extent to which findings from one setting can be applied in other different setting (Polit & Hungler, 1999). The key difference is that in quantitative research, credibility depends on the construction of the instrument while in qualitative research credibility depends on the researcher as the instrument (Patton, 2002). Nonetheless, both quantitative and qualitative researchers must demonstrate the credibility of their studies and how dependable and transferred are their studies (Johnson & Onwuegbuzie, 2004).

3.12 **Trustworthiness of methods adopted in this study**

In this study, to the best of my knowledge, methods have been selected in such a way that they are dependable, credible and transferable in addressing the research questions. Bryman (2015) argues that credibility can be maximised by cross-checking research findings with participants

to obtain confirmation that the researcher has understood participants' responses. In other words, confirming if researcher's findings and impressions are congruent with the views of participants, Bryman (2015) calls this technique as '*respondent validation'*.

In this study there were four steps which were considered to triangulate results in an attempt to yield the credibility of findings (Guba & Lincoln, 1994). Firstly, all tools designed for data collection (i.e. SoS, LoS, problem-solving tasks, questions designed for one-to-one and focus group interviews, observation forms) were cross-checked with peers and students within the Department of Computing prior to sending them to participants. Secondly, the refined tools were sent to the university ethics committee for approval. This was to find out whether the questions were clear, simple to follow and answer, and if they were capturing the intended desire and needs. Any ambiguities were clarified and edited before used by the intended participants. Thirdly, the transcribed data were shared back to the participants to verify their input and double-check if the data reflected their perceptions, understanding and opinions. Fourthly, there were three main processes of triangulation of data collection and analysis (Johnson & Onwuegbuzie, 2004) which took place as illustrated in Figure 3.1. This was consistent with the argument of Mathison (1988) that triangulation is the way out to improve validity and reliability of findings in qualitative research.

The findings from this study are transferable to other computing courses such as Network Security and Forensic because all these courses are under computer science and that Computer Networks course provides the fundamental basis to study other Computer Science courses. Conducting such an investigation to both undergraduates and postgraduate students pursuing Computer Networks course gave dependable results which can be transferable to other disciplines.

3.13 **Ethical considerations**

Ethical clearance for the current study was sought and approved by the University of Reading's Institute of Education (see Appendix 3.12). Information sheet was sent out to the Head of Computing Department of the participating University as well for their permission to allow their students and lecturers to participate in the study (see Appendix 3.13). Separate information sheets and consent forms were issued to all participants (see sample consent form in Appendix 3.14). Participants were made aware that any form of interviews conducted would be audio-recorded and transcribed. All participants were also given an option to opt out from the study without any repercussion to them. They were ensured and encouraged that their option to opt out would not in any way affect my commitment in supporting them as a lecturer in their normal day-today lessons. All the three problem-solving tasks which the six postgraduate students were given to work on in this study were part of their assessment for their degree. They were assured that their individual marks were not going to be published in this study. Any issues or discussion which triggered any ethical issues were avoided unless instigated by participants themselves, of which at that point, they were not recorded.

3.14 **Summary**

The ontological and epistemological stance of a researcher leading to the choice of methodology are significant. These become a vehicle to the research findings. Additionally, the philosophical position of a researcher plays a significant role in driving the research. Therefore, my philosophical stance in this study was pragmatism where mixed methodologies were espoused. The study has adopted predominately qualitative methods. This was because of the nature of the research questions which focused on human (students and lecturers) understanding and perceptions of this study. With the pragmatism approach any or all available methods which best address research questions become the right candidates to be used

therefore, falls on mixed methods approach (Guba & Lincoln, 1994). Literature discussed in Sections 3.3 has revealed the strength and richness of mixed methods over other methods for this study. For instance, mixing different methods does not exacerbate their weakness but rather enhances and enriches their strength (Onwuegbuzie & Johnson, 2006). If the results of different methods contradict, it becomes a good opportunity for further investigation thereby enriching the investigation in formulating new theories. Thematic analysis, one of the qualitative data analysis approach, was applied in developing codes and themes. The process of triangulation was espoused for credibility and dependability of data analysis.

The next chapter provides findings and discussion of this study.

CHAPTER 4: DISCUSSION AND ANALYSIS

4.1 Introduction

The overarching aim of this study was to investigate Computer Networks students' and lecturers' understanding of CT and how students may apply their CT skills via simulation software. Therefore, the chapter outlines the findings, discussion and analysis of this study based on the following research questions:

1) What are Computer Networks students' and lecturers' understanding of computational thinking?
2) What are Computer Networks students' and lecturers' perceptions of the use of simulation software to facilitate the application of students' computational thinking?
3) How might the use of simulation software facilitate the application of students' computational thinking?

The chapter begins by presenting the findings, discussion and analysis of the first research questions covering the predetermined and emerged themes, this is followed by the second research questions which focuses on students' and lecturers' perceptions of the use of simulation software and followed up with the third research question which shows how students apply their CT skills via simulation software. The chapter concludes with the overarching summary of the findings leading into the next chapter which discusses the conclusions and implications of this study.

4.2 Findings in relation to the first research question ('What are Computer Networks students' and lecturers' understanding of computational thinking?')

4.2.1 Introduction

The primary concern of this question was to find out students' and lecturers' understanding of computational thinking (CT). Students and lecturers were asked series of open-ended questions via SoS, LoS, one-to-one and focus group interviews which explicitly investigated their own understanding of CT, such as *what is your understanding of computational thinking*? (see Section 3.8 of the Methodology chapter). The findings of these explicit questions were cross-examined with non-explicit questions, such as *how can computational thinking help in solving computer network systems*? The findings of the results were put into themes and arranged into two categories, namely: *predetermined* and *emerged* themes which are explained and discussed on Sections 4.2.2 and 4.2.4 respectively.

4.2.2 Predetermined themes

The primary investigation of students' and lecturers' understanding of CT was based on the predetermined themes, namely: *abstraction, decomposition and generalisation*. The predetermined themes were chosen to remain consistent with many scholars (e.g., Angeli et al., 2016; Barr & Stephenson, 2011; Grover & Pea, 2013) arguing that there is no consensus definition of CT however, there are some common core concepts which describe CT, such as *abstraction, decomposition and generalisation*. Therefore, in this study, students' and lecturers' responses to their understanding of CT were based on their understanding of the concepts of CT, such as abstraction, decomposition and generalisation. These predetermined themes have been shown in Table 4.1, discussed from Section 4.2.2.1 and their sources of codes and occurrences have been provided in Appendix 4.1a.

Table 4.1

Predetermined Themes in Answering the First Research Question

Themes	Sub-themes	Sources	Codes	Aggregate codes
Abstraction				
	• Representation of problems which can be understood	11	26	40
	• Sets of abstracts and abstraction	10	14	
Decomposition				
	• Breaking problems to solvable level	15	68	82
	• Structure of the problem	7	14	
Generalisation				
	• Previous knowledge and experiences	8	18	26
	• Pattern of similarities to problem-identification	4	7	

Note: Table 4.1 shows themes and subthemes generated from codes related to research question one. The main themes are the ones highlighted in bold (aligned to the left column) while sub-themes are listed corresponding to each theme. Sources shows the number of different sources where the codes have been generated from (i.e. SoS, LoS, one-to-one and focus group interviews). Codes mean the actual number of coding occurrences from sources. Aggregate code means the total number of codes on that theme (e.g. abstraction had 40 aggregate codes).

4.2.2.1 Abstraction

Abstraction was the first predetermined theme that describes the concept of CT. Students' and lecturers' responses from explicit and non-explicit questions about their understanding of CT were cross-examined and analysed, the following two sub-themes were formed, *representation of problems which can be understood* and *set of abstract and abstraction***.** These sub-themes are shown in Table 4.1 and are discussed from Section 4.2.2.1.1.

4.2.2.1.1 Representation of problems which can be understood

There were 26 of 40 aggregate codes (65%) which referred to *representation of problems which can be understood* from the 11 sources of coding occurrences via SoS, LoS, one-to-one and focus group interviews. When both students and lecturers via SoS and LoS were explicitly asked to explain *what* their understanding of CT were, there were three responses from students and three responses from lecturers which were closely related to the concepts of abstraction (see Appendix 4.1a). For instance, one of the students' responses via SoS was, "[CT is] *the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out*". Concurring with this student's response, one of the lecturers, in response to the same question via LoS, said, "*Its* [CT] *a very valid approach which helps with understanding of what computers are actually doing or need to do*". Both responses are closely related to the concept of abstraction (Csizmadia et al., 2015; Wing, 2008) however, there were not many students and lecturers who were able to give clear responses related to the concept of abstraction in their understanding of CT until when they were asked non-explicit questions. For example, when students and lecturers were asked on *how* can computational thinking help in solving computer network systems, more ideas related to the concept of abstraction emerged. Ten students and five lecturers via SoS and LoS respectively gave responses closely related to abstraction. Students

responded that they believe that CT help them simplify the complexity of problems making them clear to understand and solve them which concur with literature (e.g., Bocconi et al., 2016; Lee et al., 2011) arguing that abstraction is the process of making complex problem easy to understand and think about. Students furthermore believe that CT help them understand technical terms and components which would otherwise be difficult to understand.

Furthermore, concurring with students' responses, lecturers believe that CT help students understand core components of computer systems making complex systems clear and easier to work on. Perhaps the ideas which are closely related to the concept of abstraction emerged when asked the *how* than the *what* of CT because the focus of this course is on the *how* students understand the concepts of computer networks by means of *doing* than memorising and citing theories. In other words, students in this course do not focus more on studying and understanding of the theories but rather in the application of already known and existing theories. As discussed in Section 3.5 of the Methodology chapter, these students spend more than four hours in the lab doing practical tasks than in lectures where they spend only less than an hour learning about theories. Therefore, these students understand and explain better via practical-oriented tasks in answering how systems work rather than what they are. For instance, these were some of their responses when they were asked the *how of CT*:

> "Helping you to understand the computer and all the technical terms and components" (*SoS*)
>
> "students need to understand what is going on from the lowest level, understanding the core components and how they work to appreciate what is going on" (*LoS*)

Students in this study were deliberately not taught the theories of CT as part of investigating their own understanding of CT. However, they were taught the theories of designing and troubleshooting computer networks as part of their normal learning outcomes of the curriculum. Furthermore, when students were asked to make any comments about CT via SoS,

most of their ideas were skills-based approaches than the concept of abstraction. They preferred practical demonstration to understand a concept. For instance, this is one of the comments one student made:

> "I think that while the theory can be taught to someone, it is something that one can only get a strong understanding through worked examples and practical work" (*SoS*)

Therefore, students understood better on the *how* than the *what* concepts of abstraction in explaining their understanding of CT.

Cross-examining students' understanding of CT via one-to-one interviews about their strategies in designing network systems more ideas describing the concepts of abstraction emerged too. This question was closely related to their normal day-to-day tasks they do in the lab; therefore, it was perhaps easier for students to explain and relate the question to their own experiences. Three of the six students who participated on one-to-one interviews gave their responses which were closely related to the concept of abstraction. One student indicated that he uses mind-mapping via paper and pencil to understand the complexity of network systems which concur with the work of Computer Science Education Research Group at the University of Canterbury in New Zealand (http://csunplugged.org/) focusing on non-computing tools to demonstrate the conceptualisation of CT. These are some of the responses students gave which closely related to the concept of abstraction when they were asked to explain the strategies they use when designing computer network systems in their understanding of CT.

> So, what I do is to design a network on the piece of paper as I understand it then after that that's when I jump into software design to help me just realise it, ya!! […] In other words, how can I make an abstract into concrete to the software that I develop (*PGstud2 – one-to-one interview*)

> Generally, when we solve a problem, the idea is to make it easy for everyone to understand even the layman. With that in mind, when solving networking problems and designing, you need to ask questions like, can I explain to a layman who doesn't know about networking what this networking is all about (*PGstud1 – one-to-one interview*)

> "You can bring an abstract to concrete by testing every area of your design before actually deploying it" (*UGstud9 – one-to-one interview*)

Furthermore, students and lecturers commented about the details which are hidden when designing network systems in response to their understanding of CT concurring with the concept of abstraction (Wing, 2011). There were three students from SoS, one student from one-to-one interview and two lecturers from one-to-one who commented about the process of hiding and highlighting important details in designing and troubleshooting systems in response to their understanding of CT. They used phrases such as *"remove things which are important," focus on details which are important"*. For example, in response to a non-explicit question via SoS which required students to explain how CT may help in solving network systems, one of the students responded and said:

> Networks are incredibly complex systems, in order to manage and design large scale systems effectively, you need to concentrate your efforts on what matters most to the system. Once you have identified those issues and dealt with them then the smaller problems should fall in place (*SoS*)

These responses show the concepts of abstraction where the complexities of networks are stripped off to the level that they can be understood concurring with the argument of Lee et al., (2011). A few students and lecturers believe that CT is the process of simplifying the complexity of network systems where a human and computer can easily understand and think about concurring with the conceptual ideas of abstraction discussed in Section 2.2.1 of the Literature Review Chapter. It was apparent that not many ideas emerged describing the concepts of abstraction in explaining their understanding of CT when students and lecturers were explicitly asked what their understanding of CT were. However, findings became different when non-explicit questions about their understanding of CT were asked. More ideas on the concepts of CT emerged revealing the nature of the students that they understand and explain better on the *how* by *doing* than the *wha*t by reciting and memorizing theories.

4.2.2.1.2 Sets of abstract and abstraction

Sets of abstract and abstraction was another sub-theme that described the concept of abstraction in response to students' and lecturers' understanding of CT. They were 14 of 40 aggregate codes (35%) in references to *set of abstract and abstraction* as shown in Table 4.1. In response to students' and lecturers' understanding of CT via SoS and LoS respectively, they described abstraction as the process of removing abstracts from their raw incomprehensible state to the level that they can understand and work on concurring with Bocconi et al. (2016). They believe that sets of abstracts exist in the logical state and not physical. They further believe that abstracts are hidden from users but known and understood by the technical designers concurring with literature (e.g., Csizmadia et al., 2015; Wing (2011). For example, they believe that a technical designer can collect customer's (user) system requirements which are understood by the customer but when the system is designed a customer cannot see the hidden details which make the system functional. Lecturers also believe that abstraction is the concept of understanding the components and their operation. They emphasised that computer networks are abstract complex systems therefore students need to understand important details (i.e. components, protocols) that they are able to work at a suitable level. Cross-examining their responses via one-to-one interviews when students were asked to describe how they remove unnecessary details when designing computer networks their responses resonated as follows:

> "Abstraction, it has to do with things that are not seen or not easily understood or are not real; they exist logically but are not physical – you cant relate to them in some form of thinking" (*PGstud1 one-to-one interview*)

> So abstraction is basically all the information that you collect from the users and then draw them on the paper ready to deploy them. Most of the technical issues are hidden from the customer but only those are needed to a customer are the ones that are shown and demonstrated. (*PGstud9 one-to-one interview*)

These responses are consistent with many scholars including those of (Angeli et al., 2016; Csizmadia et al., 2015; Wing, 2008) arguing that abstraction is an intellectual skill of removing

complex details from an object or entity to the level that one can understand and/or reuse without compromising its functionality. In other words, it is a thought process in deciding what details of an object to hide and/or highlight to reduce to the level of its fundamental characteristics (Angeli et al., 2016; Wing, 2011). Wing (2008) defines abstraction as the essence of CT and that abstraction is working with different layers to get the desired solution. For example, an abstract is decomposed to work at a level of understanding its components (Angeli et al., 2016), leading us to the concept of decomposition.

4.2.2.2 Decomposition

Decomposition was the next predetermined theme that describes the concepts of CT. Similar to Section 4.2.2.1, students and lecturers were asked both explicit and non-explicit questions about their understanding of CT. Analysing students' and lecturers' responses in reference to the concept of decomposition when describing CT, there were two main sub-themes which emerged, namely *breaking problems to solvable level* and *structure of the problem.* These sub-themes have been shown in Table 4.1 and discussed from Section 4.2.2.2.1.

4.2.2.2.1 Breaking problems to solvable level

On this sub-theme there were 68 of 84 aggregate codes (83%) in reference to *breaking problems to solvable level* as shown in Table 4.1. There were more evidences via all methods of data collection used in this study that students' and lecturers' main understanding of CT was the ability to breakdown complex problems to solvable level. For example, 30 of 69 students (43%) via SoS, five of six students (83%) via one-to-one interviews, 10 of 14 lecturers (71%) via LoS and all the three lecturers via on one-to-one interviews (100%) commented about the concept of decomposition in describing their understanding of CT (see Table 4.2).

Table 4.2

Number of Participants Against Those Who Commented About the Predetermined Themes

	Predetermined Themes						
	Number of participants	Commented about abstraction (AB)	**%age**	Commented about decomposition (DE)	**%age**	Commented about generalisation (GE)	**%age**
SoS	69	13	**19**	30	**43**	4	**6**
Stud 1-2-1	6	3	**50**	5	**83**	3	**50**
LoS	14	8	**57**	10	**71**	1	**7**
Lec 1-2-1	3	2	**67**	3	**100**	2	**67**

Note: Table 4.2: shows the number of different data collection methods (i.e. SoS, LoS, students' one-to-one interviews, LoS, and lecturers' one-to-one) as shown on the first right column; then the number of participants on the corresponding data collection method followed by the number of participants who commented about the concepts of AB, DE, GE and their percentages respectively

Additionally, among the three predetermined themes, it was only the *breaking-down-to-solvable-level* sub-theme that had the highest number of coding occurrences (15) (see Table 4.1 and Appendix 4.1a). This could be because the main method of learning in this course is practical-based as discussed in Section 3.5 of the Methodology chapter involving designing and troubleshooting arrays of small networks to build up complex network infrastructure. There were no differences when students and lecturers were asked either explicit or non-explicit questions about their understanding of CT; their responses remained consistent that the process of breaking down computer network problem into smaller solvable task is CT. Literature reviewed in Section 2.2.2 has shown that the concept of breaking down a complex problem to solvable task is decomposition. It became clear that their understanding of CT is based on the concept of decomposition. They cited several examples to complement their understanding of CT using the concept of decomposition. For example, these students believe that if complex problems are split into logical self-contained parts they are easy to solve and eventually help them to build up the entire network infrastructure. Therefore, they believe that the concept of

splitting a problem into logical self-contained parts is CT. When students where asked a non-explicit question via SoS, such as *how* CT may help in solving computer network systems, their responses were in line with the concept of decomposition as follows:

> "By breaking systems down into logically self-contained parts, it makes it easier to conceptualise the processes behind the inherent A to B to C to D platform independent design philosophy that enables computer networking" (*SoS*)

> I think it [CT] is a very logical way of thinking, and if a problem is broken down into smaller issues, for example if the designer can test each stage of designing a network and then find one part isn't working then this breaks down the problem quite a bit, and allows the designer to focus on the area in which they believe the system has broken down (*SoS*)

Furthermore, when lecturers were asked via LoS to provide any comments about CT, they believe that it should be apparent for students to breakdown problems to the level they can understand to solve them. However, on the other hand, some lecturers believe that the skills to break complex problems to smaller tasks (i.e. decomposition) is not obvious to students. Students are supposed to be taught how to decompose problems so that they can be thinking at a suitable level rather than trying to understand the entire complex system at once. When these lecturers' ideas were cross-examined with other lecturers via one-to-one interviews, their responses resonated with each other that students are supposed to be taught how to break down problems. For example, when lecturers were asked to make any remarks about CT during one-to-one interviews one of the lecturers said:

> Am sure its [CT] probably one of the main key elements that they were taught that when they have a problem they need to break it down for instance in programming into easy steps and make sure that it works and then you can build on top on what it has worked [...] Because the issue is that students do not always have the skills of how to break the problems into small solvable solutions (*Lec3 – one-to-one interviews*)

Students and lecturers in this study believe that the concept of decomposition means CT. For example, their understanding is that if a student is able to break a complex problem to the level

that she or he can think about and solve, it means she or he has applied the skills of CT. This was evident across all different questions which students were asked too about their understanding of CT. Furthermore, when students were asked via SoS to select the first strategy they apply when solving a complex computer network design problem, 51% (see Figure 4.1) selected the concept of breaking down the main problem into smaller manageable problems – which is a concept of decomposition.

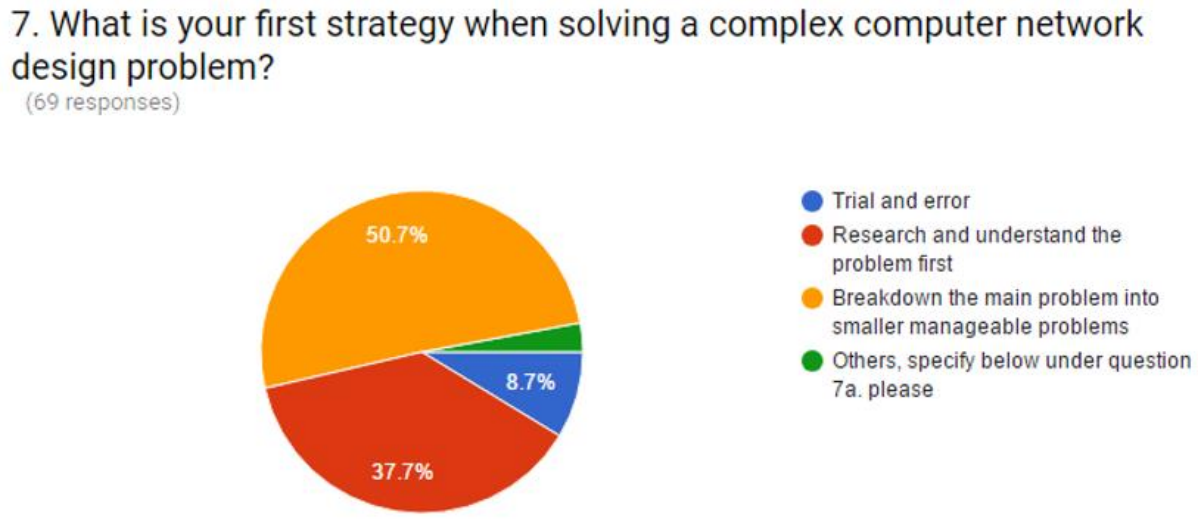


Figure 4.1. Students First Strategy when Solving Complex Network Design Problem

Additionally, one of the lecturers via LoS gave an example that designing a computer system involves different technologies, such as, L2 (layer 2 switches), L3 (layer 3 switches or routers), wireless, security and many more other technologies. Therefore, all these technologies should be examined *separately* leading to solving and building up the entire network infrastructure together. In essence, the lecturer was describing the concept of decomposition (Bocconi et al., 2016). Another student during one-to-one interview, in response to his understanding of CT, described the concept of decomposition by giving an example of how he designed a complex problem with different technical needs. For instance, this is how he said:

> I think the difference is that in computational thinking you break down problems into chunks what you can understand like on networks, you may first of all look at interfaces and then you may need to do VPN. One thing I have learnt today, like now we were building a network together, so we said, let's do the interface first and then do the VPN and then we can add some security features later. So that is almost what computational thinking is all about (*UGstud6 – one-to-one interviews*)

Students commented that they were able to troubleshoot and solve complex problem by breaking them into smaller chucks which they could understand, solve and evaluate separately before building them together, which in essence, they were applying the concepts of decomposition. Other students further commented that by breaking down problems it helped broaden their minds that complex problems became simple to solve. It was effortless to students to give examples how they applied the concepts of decomposition to explain their understanding of CT. Examining the lecturers' experiences and qualifications, 12 of 14 lecturers (86%) have industrial experiences between the range of three to more than nine years and eight of 14 lecturers (57%) without doctorate degrees have extensive industrial experiences and professional networking skills to enrich students industrial networking skills (see Table 3.5 in the Methodology Chapter). Therefore, it is evident that students are taught based on the extensive industrial experiences of these lecturers in designing and troubleshooting networks. For instance, one of the students during their focus group interviews made it clear that they are not taught the concepts of CT instead they are taught skills which will enable them to thrive and excel in networking industries:

> I don't feel our course teaches us any computational thinking, I feel our course is designed to in-cooperate workforce processes. The course is designed to introduce work place processes, best practices from hardware, best practices from enterprises. It is designed ready to get you in the work place; 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 (*UGstud1 – UGFG*)

Despite not explicitly taught the concepts of CT as the students commented, evidence in this study has shown that students' and lecturers' main understanding of the concept of CT is based on students' practical skills to breakdown computer network components into logical self-contained parts which they can solve, manage and evaluate separately concurring with many scholars (e.g., Bocconi et al., 2016; Csizmadia et al., 2015; Grover & Pea, 2013). These students and lecturers used words, such as "*decomposition from abstract*", *"abstracting a big problem into solvable ones", "breaking down problems into solvable tasks",* in trying to describe their understanding of CT – which in essence, revealed the concept of decomposition. Furthermore, the results have shown that lecturers' industrial experiences and professional skills in networking have had direct impact and influence in the way student solve networking problems leading to the views of their understanding of CT as the concept of decomposition.

4.2.2.2.2 Structure the problem

On this sub-theme, there were 14 of 84 aggregate codes (17%) in references to the *structure of problem* as shown in Table 4.1. They were explicit and non-explicit questions which were used to solicit students' and lecturers' understanding of CT. It did not make any difference from the type of the question posed, their understanding of CT is that it helps to structure problems in such a way that it is easy and simple to solve complex problems. They believed that CT is taking a complex problem and solving it in manageable pieces (which in essence refers to the concept of decomposition). Some lecturers via SoS, believe that when students are shown how to abstract away complexity and generalise problems they will be able to re-use their knowledge and experiences to enable them focus on small parts of the problem leading to more complex tasks. They all believed that the ability to work on smaller tasks to yield complex tasks is the concepts of CT. Some of the other typical examples which students gave were the ability of solving one piece of a problem at a time or in bits before building them together to

form the final network infrastructure. For instance, students gave their comments such as this one:

> "Divide the tasks just to small tasks and solve a problem from one step and then go to the other step and then conclude one step before into the next step and then come to the final conclusion" (*UGstud6 – one to one interviews*)

They believe that the ability to work on one piece of a problem at a time, in such a structural manner, is the concept of CT. Again, this could be contributed from the fact that these students spend much of their learning period building up and troubleshooting networks in the lab. There are more familiar with working on small pieces of work at a time to build up a complex network infrastructure. For instance, during focus group interviews when students were asked to explain the strategies they use to identify and solve a network problem one of the students said:

> A network design is in such a way that it comes from a bit to bits to be a holistic giant topology so in solving issues in a network you have to go from little bit. You have to look at the smallest bits (*PGstud1 - PGFG*)

Therefore, to them solving bits and bits of network problem at a time comes as the second nature in identifying and solving problems. Perhaps thus why it was easy to relate their problem-solving strategies to the concept of decomposition in explaining their understanding of CT. As alluded to in Section 4.2.2.2.1, students and lecturers were very clear on the concept of decomposition in explaining their understanding of CT.

4.2.2.3 Generalisation

Generalisation was the last predetermined theme that describes the concept of CT focused in this study. Similar to Sections 4.2.2.1 and 4.2.2.2, students and lecturers were asked both explicit and non-explicit questions about their understanding of CT. When students and lecturers were explicitly asked to explain *what* their understanding of CT were, there were

almost no relevant responses via SoS, LoS, one-to-one and focus group interviews which referred to the concept of generalisation. There was only one student via SoS whose response related closely to the concept of generalisation in explaining his understanding of CT based on literature discussed in Section 2.2.3. His response was that "*Computational thinking is identifying a problem and find ways to solve it based on your general computing knowledge"*. Table 4.2 and Appendix 4.1a show the number of students and lecturers who commented about the concept of generalisation based on SoS, students' one-to-one interviews, LoS and lecturers' one-to-one interviews. Because of lack of tangible responses from the explicit questions about their understanding of CT in relation to the concepts of generalisation, all two sub-themes formed under generalisation were based on non-explicit questions about their understanding of CT.

When students and lecturers were asked non-explicit questions about their understanding of CT, such as *how* CT may help in solving network systems, *how* may CT be integrated in network design, and *what* strategies students use to identify problems and solve them, ideas on the concepts of generalisation emerged. For examples, they commented about the use of prior knowledge and experiences in solving problems and the ability to identify problems by using the patterns of similarities, concurring with literature discussed in Section 2.2.3. The last non-explicit question which they were asked about *what* their strategies were, as discussed in Section 4.2.2.1.1, students may have found it related to their daily practical tasks in designing and troubleshooting networks in the lab, therefore ideas about the concepts of generalisation emerged too. These findings furthermore enforce the fact that these students easily relate practical tasks to their understanding of concepts more than reciting theories. In other words, they respond better by means of demonstration than explaining.

The following were the two sub-themes formed, namely: *previous knowledge and experiences* and *pattern of similarities to problem identification* as shown in Table 4.1 and discussed from

Section 4.2.2.3.1 and their sources of codes and occurrences have been provided in Appendix 4.1a.

4.2.2.3.1 Previous Knowledge and experiences

The first sub-theme had 18 of 25 aggregate codes (72%) related to *prior knowledge and experiences*. Lecturers believe that if students are taught and shown how to generalize problems, it will be easy for students to apply their previous knowledge and experiences in solving problems. For example, via one-to-one interview another lecturer gave an example that routing protocols are basically the same things therefore if students are taught on how one routing protocol functions students can apply the same knowledge on a different routing protocol. Concurring with the lecturers' comments in describing the concept of generalisation, students, via SoS, one-to-one and focus group interviews, believe that prior knowledge to networking plays a significant role in problem solving. For instance, when students were asked to explain how CT may help in solving network systems via SoS, one of the students commented that:

> "It [CT] can help because in order to solve a complex network problem you first need to have the general knowledge of computing" (SoS)

Students believe that without prior knowledge they cannot abstract away the complexity of a problem and cannot decompose a problem to the level that they can think about solving them. Basically, they were saying without their prior knowledge and experiences, students can not apply the concepts of CT. One of the students during focus group encapsulated this point well here:

> So your knowledge to the problem you are solving is significant. I think it's not a matter of computational thinking but the prior knowledge that helps you in breaking down that problem. Background knowledge helps in understanding the similarities and differences which will help in making appropriate decision in solving that problem (*UGstud6 – UGFG*)

On another hand, based on the comments made by this student, it appears that some students are not clear what CT is all about. For example, this student (*UGstud6*) appears to say that the ability to use prior knowledge in breaking down problems is not CT. Additionally, three students and two lecturers during one-to-one interviews admitted that they had to search on Google to find out the meaning of CT. Therefore, these conflicting ideas about their understanding of CT show their uncertainty about the definition of CT which concur with many scholars (e.g., Angeli et al., 2016; Barr & Stephenson, 2011; Grover & Pea, 2013) arguing that there is no consensus about CT. Students and lecturers seemed unaware of the meanings and concepts of generalisation in answering their non-explicit questions about their understanding of CT. However, it was during coding process that their responses showed that they were explaining the concepts of generalisation. The findings from this study show that very few students' and lecturers' responses related to the concept of generalisation when explaining their understanding of CT.

4.2.2.3.2 Pattern of similarities to problem-identification

The second sub-theme had seven of 25 aggregate codes (28%) related to *pattern of similarities to problem identification*. Codes for this sub-theme emerged only from five students via SoS, one-to-one and focus group interviews. There were no codes which emerged from lecturers' responses either via LoS or one-to-one interviews. They were limited responses related to this sub-theme. However, the five students who explained their understanding of CT related to the concept of generalisation. For instance, these students believe that the ability to identify patterns of similarities, differences and connections help in their learning experiences on designing and troubleshooting network systems. They related their responses to their own experiences in how they troubleshoot network problems. Some went further to explain how they use non-computing tools to identify similarities in trying to troubleshoot a network

problem before they solve them. For instance, this is how some students explained their strategies in identifying similarities when troubleshooting network designs:

> you can print out some configurations say you want to see VLANs between switches. You can see by using vlan brief command to check from switch 1 and then go another switch and so the same to see if you can see where VLANs are not properly configured (*UGstud9 – one-to-one interview*)

> "I then looked at the similarities of serial interfaces against other routers and then I was able to connect them together. I saw that there were networks which were sharing the same network hence I connected them together" (*PGstud4 – PGFG*)

Although students were able to explain the concept of generalisation via non-explicit questions about their understanding of CT there are three observation made. Firstly, these students were not aware that they were explaining the concept of CT bearing in mind that they did not know what CT was and that largely their responses did not emerge from explicit questions about their understanding of CT. Secondly, there were limited responses related to the concepts of generalisation as observed by the number of codes emerged from SoS, LoS, one-to-one and focus group interviews. Thirdly, no code emerged from lecturers' responses via LoS or one-to-one interviews. This observation was different when analyzing their understanding of CT based on other concepts of CT, e.g. the concept of decomposition as discussed in Section 4.2.2.2.1. Therefore, these findings show that students and lecturers did not have convincing ideas related to the concept of generalisation in their understanding of CT.

4.2.3 Summary for predetermined themes

The discussion and analysis provided from the predetermined themes show that students and lecturers were mainly able to explain the concepts of abstraction and generalisation when they were asked the non-explicit questions of their understanding of CT. They were able to relate the concept of abstraction and generalisation when questions covered the areas which they were familiar with, such as strategies they use in designing and troubleshooting network systems.

Even then, they were not aware that they were describing the concepts of abstraction and generalisation in explaining their understanding of CT. It was only made clear during the coding process that they were describing the concepts of abstraction and generalisation in explaining their understanding of CT. However, this was not the same when describing the concepts of decomposition when explaining their understanding of CT. There were more evidences when explaining the concept of decomposition regardless whether they were asked explicit or non-explicit questions about their understanding of CT. Almost every explanation about their understanding of CT was related to the technical skill of breaking down complex problems to the level that students can be able to solve. For instance, via SoS, LoS, one-to-one and focus group interviews, 84 coding occurrences related to the concept of decomposition, while 40 and 26 coding occurrences related to the concepts of abstraction and generalisation respectively (see Table 4.1 and Appendix 4.1a). Additionally, it was clear that lecturers' industrial experiences and professional practical skills had direct influence on the style of students' application of designing and troubleshooting network systems. Students made some remarks that the course is mainly designed to teach them skills which will enable them to work in the networking field once employed and not focusing on CT. Therefore, based on findings from the predetermined themes, Computer Networks students and lecturers who participated in this study largely believe that CT is the practical skill of breaking down complex problems to solvable tasks that students can solve, manage and evaluate separately.

4.2.4 Emerging themes

Besides the predetermined themes, there were additional two themes, namely: *problem-solving approach* and *algorithmic approach* which emerged in investigating students' and lecturers' understanding of CT. These themes and their sub-themes have been shown in Table 4.3 and discussed from Section 4.2.4.1 with coding occurrences and sources provided in Appendix 4.1b.

Table 4.3

Emerged Themes in Answering the First Research Question

Themes	Sub-themes	Sources	Codes	Aggregate codes
Problem-solving approach				
	• Problem-solving technique	8	35	82
	• Logical thinking	7	23	
	• Thought process in problem-solving	6	15	
	• Think like computers	2	9	
Algorithmic approach using steps and flow chart		7	12	12

Note: Table 4.3 shows themes and subthemes generated from codes related to research question one. The main themes are the ones highlighted in bold (aligned to the left column) while sub-themes are listed corresponding to each theme. Sources shows the number of different sources where the codes have been generated from (i.e. SoS, LoS, one-to-one and focus group interviews). Codes mean the actual number of coding occurrences from sources. Aggregate code means the total number of codes on that theme (e.g. problem-solving approach had 82 aggregate codes).

4.2.4.1 Problem-solving approach

Problem-solving approach was the first theme that emerged from data analysis and is not part of the three predetermined themes. Problem-solving approach had more coding occurrences than any theme including predetermined themes in answering the first research question. There were more students and lecturers who explained about problem-solving approach as their understanding of CT (see Table 4.4).

Table 4.4

Number of Participants Against Those Who Commented About the Emerged Themes

	Emerged Themes				
	Number of participants	Commented about problem solving (PS) approaches	%age	Commented about algorithmic (AL) approaches	%age
SoS	69	52	75	3	4
Stud 1-2-1	6	3	50	2	33
LoS	14	7	50	2	14
Lec 1-2-1	3	1	33	-	-

Note: Table 4.4 shows the number of different data collection methods (i.e. SoS, LoS, students' one-to-one interviews, LoS, and lecturers' one-to-one) as shown on the first right column; then the number of participants on the corresponding data collection method followed by the number of participants who talked about the concepts of PS and AL and their percentages respectively.

An aggregate of 82 codes were formed from different sources of data, such as SoS, LoS, one-to-one and focus group interviews in response to the explicit and non-explicit questions about students' and lecturers' understanding of CT which referred to *problem-solving approach*. Four sub-themes emerged, namely: *problem-solving techniques, logical thinking, thought process in problem-solving* and *think like computers*. These sub-themes are shown in Table 4.3 and discussed from Section 4.2.4.1.1.

4.2.4.1.1 Problem-solving techniques

There were 35 of 82 aggregate codes (43%) which described CT as a *problem-solving technique*. When students and lecturers were explicitly asked to explain their understanding of CT via SoS, LoS, one-to-one and focus group interviews, 19 of 35 codes (54%) emerged describing CT as problem-solving technique. Students' and lecturers' explanation of problem-solving technique revolved around the concepts of abstraction, decomposition and

generalisation. For instance, when students were asked explicitly to explain their understanding of CT, they believe that CT is a problem-solving technique that helps to make complex network problems clear for the computers and humans to understand and work on concurring with literature discussed in Section 2.2.1 about the concept of abstraction. They also believe that CT is a problem-solving technique that complex problems are broken down into smaller solvable tasks concurring with literature discussed in Section 2.2.2 about the concept of decomposition. Almost every student and lecturer who participated on one-to-one and focus group interviews commented on the view that CT is the concepts of problem-solving involving breaking down problems to solvable tasks (see Appendix 4.1a on decomposition). Even for those who were not sure about the term "*computational thinking*", they would guess by mentioning phrases, such as "*breaking down of problems*", "*splitting out of problems*", "*solving problems in bits*". For example, one of the students via SoS in explaining his understanding of CT said:

> "I am unaware what computational thinking actually is, I assume it is something to do with the breaking down of computer related tasks and simplifying problems" (*SoS*)

Furthermore, students believe CT is an approach of identifying and solving problems using prior knowledge and experiences concurring with literature discussed in Section 2.2.3 about the concept of generalisation. Additionally, students went on further to comment that CT is the approach of formulating problems and finding solutions; a technique where computers are involved to solve problems. One of the students during a focus group interview empathised that any method applied on solving a computing problem is CT. This is what he said:

> "where we apply methodologies in solving problems we apply computational thinking" (*UGstud5 – UGFG*).

Therefore, students' responses seem to indicate that CT involve problem-solving techniques in applying the concepts of abstraction, decomposition and generalisation. Concurring with

students' responses, lecturers believe that CT is a problem-solving technique that involves abstraction, decomposition and algorithmic thinking. For instance, one of the lecturers described his understanding of CT as follows:

> An approach to thinking, especially problem solving, which is based on a top-down decomposition from the abstract into small, solvable, concrete problems. Each concrete problem can be solved using a set of steps in a way that is analogous to a mathematical algorithm (*LoS*)

When students and lecturers were asked non-explicit questions about their understanding of CT, there were consistent responses emphasizing on CT as a concept of problem solving. They believe that CT help them develop good reasoning skills in solving computer problems involving computer networks, computer software development and IT projects. They strongly believe that CT is an approach to solving problems in computer networking. They often used words, such as "*complex thinking in problem-solving", "thoughts in problem-solving", "technical way of thinking in problem-solving", "artificial intelligence in problem-solving"*, in describing CT. For instance, one of the students emphasised the need of CT in problem solving as follows:

> In the computing world you have […] to solve a lot of computational problems and have to face computer network problems that you need to solve […]. I need computational thinking when am solving […] a computer networking problem, because computational thinking will help me think in solving problems as I go ahead solving a problem (*PGstud2 – one-to-one interviews*)

In summary, it was evident that students' and lecturers' understanding of CT is the problem-solving involving the concepts of abstraction, decomposition and generalisation as illustrated from Figure 4.2 concurring with many scholars as discussed in Section 1.2 of the Introduction chapter.

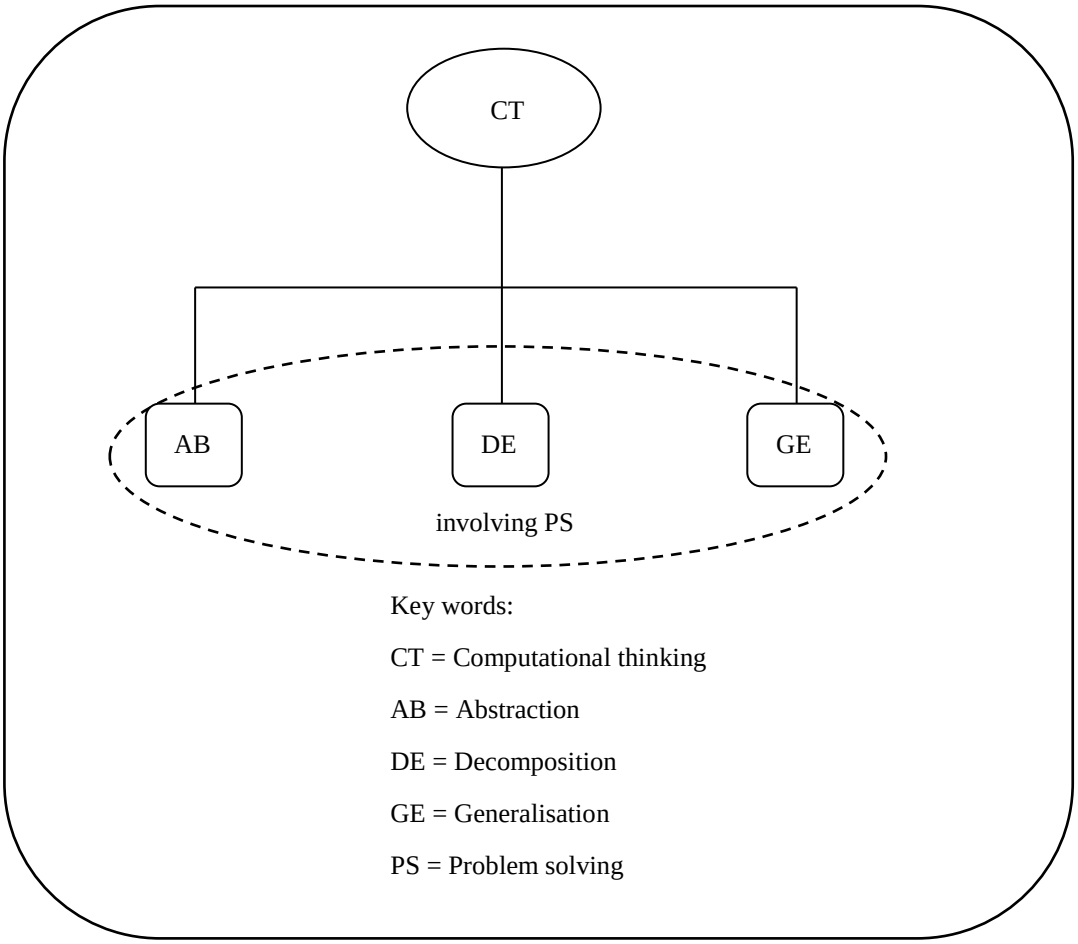


Figure 4.2. CT Involving Problem Solving Based on Abstraction, Decomposition, Generalisation

4.2.4.1.2 Logical thinking

There were 23 of 82 aggregate codes (28%) which described CT as a *logical thinking* in problem solving. Students and lecturers believe that any computing problem-solving task that requires logical-thinking approach means that it is CT. They cited examples, such as the ability to break or divide problems into solvable tasks requires logical reasoning; they also believe that fault-finding and exploring possible solutions require logical reasoning. Some lecturers further believe that through problem-solving approaches students become innovative and develop stronger logical reasoning in designing networks. These were some of their responses:

> "Thinking in a logical matter in order to solve a problem (possibly by breaking it down into smaller parts)" (*SoS*)
>
> "Computational thinking is logical thinking, […] And when we are dividing a problem into small bits, we are using our logical thinking hence it is computational thinking" (*PGstud6 – one-to-one interview*)
>
> "It will make the pupil more innovative and can even help in foreseeing and designing networks with a strong logical reasoning on why and how it happened or will happen" (*LoS*)
>
> "Solving problems based on your logical thinking is computational thinking" (*UGstud7 - UGFG*)

Although students' and lecturers' view of CT was based on the logical thinking in exploring ideas and coming up with solutions their responses to logical thinking in problem solving, appear to be describing the concept of decomposition. This observation shows that students' and lecturers' main understanding of CT is the skill of breaking or dividing problems into smaller solvable tasks.

On the other hand, Selby and Woollard (2010) argue that logical thinking is a broad term that may refer to many interpretations therefore they raise a debate as to whether logical thinking should be considered in the definition of CT. For instance, while other scholars (e.g. Wing, 2006) argue that logical thinking is the process of CT others (e.g., Selby & Woollard, 2010) argue that logical thinking can be applied to any problem-solving task and not only on CT therefore logical thinking is a broad term to be included in the definition of CT.

However, the findings from this theme seem to suggest that students' and lecturers' definition of logical thinking is based on problem solving in breaking and diving tasks to the level that they can solve.

4.2.4.1.3 Thought process in problem-solving

There were 15 of 82 aggregate codes (18%) emerged only from students' responses via SoS, one-to-one and focus group interviews describing CT as a *thought process in problem solving*. Students believe that CT is the thought process or "complex thinking" involved in solving

problems; they also believe that whenever they are solving a computing problem that involves their reasoning skills, it means it is CT. They furthermore commented that CT helps them to think when solving a computational problem. However, they were not clear as to how CT help them to think which might indicate the uncertainty of their understanding of CT. On another hand, there were two students via one-to-one interviews who defined CT in consistence with Wing's (2006) original definition of CT. For example, this is how these students said when asked what CT was:

> It is the thought process of solving computer problems but not only on computer problems but in other discipline too and our daily life – it is the thought process. It is formally defined as a thought process in solving problems in computer related problems (*PGstud1 – one-to-one interview*)

> it [CT] involves computers and therefore the computers should compute the information you feed it. And so when you are applying computational thinking it's now your process of thinking in the way the computer will solve your problems so as the definition stays "computational thinking" (*PGstud2 – one-to-one interview*)

Other students in explaining their understanding of CT via SoS, used phrases, such as "*thought process in formulating problems", "thinking that produces automated solutions to problems", "technical way of thinking", "complex thinking*". Therefore, from this theme, it appears that students believe that any thought process that triggers off their reasoning skill to solve a computing problem is CT. As alluded to from previous sections, the learning style of these students involve problem-solving in designing and troubleshooting network systems, therefore, students may have found it easy to relate their reasoning skills in solving computing problems to CT. Nonetheless, the definition of CT that these students have provided remain consistent with the proposal and argument of many scholars (e.g. Aho, 2012; Selby & Woollard, 2010; Wing, 2006) that CT is the thought process in problem solving though the definition is still under debate (Bocconi et al., 2016; Kalelioglu et al., 2016).

4.2.4.1.4 Think like computers

There was a total of nine of 82 aggregated codes (11%) only from students via SoS, in which they referred CT as *think like computers*. Seven of nine codes emerged when students were asked explicitly their understanding of CT. The remaining two codes emerged when students were asked non-explicit question about their understanding of CT. In describing their understanding of CT, students used words, such as "*thinking like PC*", "*thinking in a way a computer processes actions*", and "*think like a computer*". Students seem to position a machine as superior over a human in problem solving. However, Repenning et al. (2016) argue that CT involves the process of human and computers in problem-solving. For instance, Repenning and his colleagues argue that human formulate problems and express solutions but computers executive and present the solutions showing the consequence of the human thinking. Computers can only process what a human feed them otherwise on their own are dormant (Wing 2006, 2008). Therefore, a human thought process or his/her involvement in calculating or solving problems is more significant. However, data was only collected from SoS therefore the interpretation of students' understanding of CT on this theme cannot be generalised.

4.2.4.2 Summary for problem-solving approach

Students and lecturers referred CT as the problem-solving approach involving thought processes, logical thinking techniques which humans and computers can work on. There were more coding occurrences about problem-solving approach than any theme including those from predetermined themes in answering the first research question. However, students' and lecturers' understanding, and explanation of problem-solving technique was revolving around the concepts of abstraction, decomposition and generalisation. For instance, they mentioned about the thought process in formulating problems and solutions that a human and computer can understand and work on. That is the concept of abstraction (Wing, 2011). Additionally,

they mentioned of the problem-solving process that helps in decomposing an abstract to small solvable tasks; again, that is the concept of decomposition (Csizmadia et al., 2015). They furthermore commented on the importance of prior knowledge in computing to solve problems effectively. Other students discussed about using road maps in identifying problems. The two latter ideas are associated with the concepts of generalisation (Bocconi et al., 2016). Their ideas were focusing on what they referred to as "deep and complex thinking" involved in the way computers solve problems that a human can understand. Additionally, it was noted that their focus on problem-solving approach was based on the skill to break or divide complex problems to solvable tasks – which is the concept decomposition. Therefore, it became clear that their problem-solving approach referred more to the concept of decomposition than the concepts of abstraction and generalisation focused in this study.

The findings in this theme have shown that students' and lecturers' understanding of CT involve problem-solving approaches covering the concepts of CT as illustrated in *Figure 4.3*. These results remain consistent with literature discussed in Section 1.2 that many scholars (e.g. (Barr & Stephenson, 2011; Grover & Pea, 2013; Kalelioglu et al., 2016) argue that CT is a problem-solving skill that involves the concepts of abstraction, decomposition, generalisation among other CT concepts highlighted in Table 2.1.

4.2.4.3 Algorithmic thinking approach

Algorithmic thinking approach was the second theme that emerged from data analysis and is not part of the three predetermined themes. There was only one theme, namely: *algorithmic approach using steps and flow charts*. In response to students' and lecturers' explicit and non-explicit questions about their understanding of CT via SoS, LoS, students' one-to-one and focus group interviews, 12 codes about algorithmic thinking approach emerged (see Appendix 4.1b). Additionally, they were a total of seven students via SoS, students' one-to-one interview and

LoS who commented about algorithmic approach (see Table 4.4). Additionally, there was only one student who commented about algorithmic approach via focus group interviews.

When students and lecturers via SoS and LoS respectively were explicitly asked to explain their understanding of CT, three of 69 students (4%) believe that CT is the way humans construct step-by-step thinking ideas (e.g. by creating a flow chart of ideas) to solve problems. Concurring with students' responses, two of 14 lecturers via LoS believe that each concrete problem can be solved by using a set of steps in a way that is analogous to mathematical algorithms. Other two of six students who participated in one-to-one interviews believe that CT is the process of developing algorithms which can be fed to computers to solve computational problems. For instance, one of the students during one-to-one interviews when asked to explain his understanding of CT commented that:

> "[CT is] solving problems through some kind of algorithms which you feed to the computers and try to help people solve those problems […] in computational thinking you develop these algorithms and simulated software to solve problems" (*UGstud9 – one-to-one interviews*)

Students furthermore believe that one problem in networking creates another problem therefore creating a flow chart that provides step-by-step methods in solving problems refers to CT. These students believe that the use of algorithms or flow charts in solving computational problems is CT. On another hand, it was evident that some students understand CT in reference to how algorithms are applied in programming languages which concur with the study of many scholars (e.g., Kalelioglu et al., 2016; Repenning et al., 2016). For instance, when students were asked via focus group interviews to explain how CT can be applied to solve problems, one of the students commented in references to programming language as follows:

> if you take programming language problems, a lot of them use, "if statements", and you could set a string of "if statement" and design a network. Say if this is this, do this or do that. I would say that's how you would design networks through computational thinking or design a flow chart that helps you to design a network every time you want to design a network based on specific needs or you are able to discover a

network topology of which somehow you have gone into network configuration just by following flow chart (*UGstud1 – UGFG*)

Clearly there were few ideas from students and lecturers who explained their understanding of CT in relation to algorithmic thinking approach. Most of them, were not sure as they used words like "*I am guessing*…." For those who described the concept of algorithms, their focus was on the use of step-by-step to create flow chart in solving computational problems.

4.2.5 Summary of Emerged themes

There were two main themes which emerged, namely *problem-solving approach* and *algorithmic approach* in explaining their understanding of CT. While the latter theme did not come out with many coding occurrences and that not many students and lecturers commented on the concepts of algorithmic approach, there were more coding occurrences from the former theme than any of the themes, including those from predetermined theme where students and lecturers commented about problem-solving approach. Students' and lecturers' comments on the problem-solving approach focused on the technique involved in the application of CT skills, such as abstraction, decomposition and generalisation. Additionally, it came out clear that when students and lecturers were commenting on problem-solving approach, they consistently referred to the problem-solving skill in breaking down problems to solvable tasks. This implies that students' and lecturers' main understanding of CT revolves around problem-solving technique on the application of decomposition. On another hand, the few students and lecturers who mentioned about algorithmic approach, focused on the step-by-step approach of using flow-chart in problem solving. Some students made a reference to problem-solving approach in programming language which use, "*if statement*" to imply the concept of algorithms. The findings from the emerged themes have shown that Computer Networks students and lecturers relate problem-solving technique to the application of abstraction, decomposition and

generalisation, though they focused more on the concept of decomposition. On the other hand, they could only relate the algorithmic approach to programming-related subject. This could be because these students and lecturers have a focus mind on designing and troubleshooting computer networks which largely do not involve algorithms.

4.2.6 Summary of findings of the first research question

The findings have shown that students and lecturers in this study associate the definition of CT with the concepts of decomposition more than the two other predetermined themes of CT. Although when analysing the two emerged themes (i.e. *problem-solving approach* and *algorithmic approach*), students and lecturers commented a lot more about problem-solving approach than any theme including predetermined themes, their understanding of CT was based on problem-solving skill in applying the concept of decomposition. There were few ideas showing their understanding of abstraction but only became clearer when asked *how* CT might be applied than *what* it is. Insufficient information was shown about their understanding of the concept of generalisation. There was no idea to compare from one-to-one and focus group interviews which further evidenced their lack of understanding of the concept of generalisation. Only when students and lecturers were asked on *how* CT might be applied in problem-solving that meaningful ideas about the concepts of generalisation emerged too. Lecturers and students found it easier to explain their understanding of CT based on their practical approaches to problem-solving in network design and troubleshoot which addresses the *how* than the *what* questions. This could be because their teaching and learning style focus on developing their computer networking skills by *doing* in answering *how* network systems work than memorising the theories of networking system which may require them to answer the question *what*. On another hand, these findings are showing how Computer Networks students understand the application of CT via their own practical and skills-based experiences.

On another hand, the different opinions revealed by students and lecturers about their understanding of CT in this study are consistent with many scholars (e.g., Angeli et al., 2016; Barr Stephenson, 2011; Grover & Pea, 2013) arguing that there is not yet an agreed-up definition of CT. Some scholars (e.g., Angeli et al., 2016; Selby & Woollard, 2014) have further argued that since there is not yet agreed upon definition of CT makes it difficult to define elements of how to measure and assess the concepts of CT in teaching and learning. There is currently a great deal of work that has already been put in place in trying to provide information that may help in teaching and assessing CT (Angeli et al., 2016; Blikstein, 2018; Csizmadia et al., 2015). However, their focus is at primary and secondary school level. There is no much evidence how this has been covered at higher education particular on students studying for Computer Networks courses. The current literature shows that the focus of CT is on programming-related subjects. Their argument is that once students at primary and secondary levels have been well grounded in the concepts of CT they will be better at handling such concepts at higher education. In the UK, where this study has been conducted, CT has been introduced in schools in 2014 (Bocconi et al., 2016) meaning that there is a gap between the current students in HE and those who are being taught the concepts of CT at school. Therefore, would be recommendable to teach and apply the concepts of CT at university now particularly in the area of computer networks where the literature reviewed in this study has shown that gap. The results from this study suggest that the fundamental principles of examining students' and lecturers' understanding of CT and their application are vital for further research.

4.3 Findings in relation to the second research question ('What are Computer Networks students' and lecturers' perceptions of the use of simulation software to facilitate the application of students' computational thinking?')

4.3.1 Introduction

This question was primarily concerned with finding out students' and lecturers' perceptions of the use of simulation software with the view of understanding how simulation software may facilitate the application of students' CT skills. Initially using Likert scale, open-ended and closed-ended questions via SoS and LoS, students and lecturers respectively were investigated on their perceptions of using simulation software to facilitate the application of students' CT skills. To cross-reference students' responses from SoS, one-to-one and focus group interviews were followed up. However, lecturers' responses from LoS were followed up by only one-to-one interviews because lecturers did not participate in a focus group interview. It was practically difficult at the time to bring the lecturers together for a focus group interview because it was the period when students were being assessed, therefore all lecturers were fully engaged. Collectively data from SoS, LoS, reflective reports, one-to-one and focus group interviews were analysed and coded to form four main themes and their associated sub-themes as shown in Table 4.5 and discussed from Section 4.3.2 with coding occurrences and sources provided in Appendix 4.2.

Table 4.5

Emerged Themes in Answering the Second Research Question

Themes	Sub-themes	Sources	Codes	Aggregate codes
Simplicity of using SS				
	• Simple and flexible use of SS	9	41	52
	• Visual representation of complexity	4	11	
Effectiveness for Abstraction, Decomposition & Generalisation	• Provides network design platform	7	27	49
	• Complex problems are broken & understood	5	22	
Satisfaction of using SS		12	58	58
Challenges of using SS		8	21	21

Note: Table 4.5 shows themes and subthemes generated from codes related to research question one. The main themes are the ones highlighted in bold (aligned to the left column) while sub-themes are listed corresponding to each theme. Sources shows the number of different sources where the codes have been generated from (i.e. SoS, LoS, one-to-one and focus group interviews). Codes mean the actual number of coding occurrences from sources. Aggregate code means the total number of codes on that theme (e.g. simplicity of using SS had 52 aggregate codes). SS stands for simulation software.

4.3.2 Simplicity of using simulation software

The first theme that emerged in addressing the second research question was *simplicity of using simulation software.* Two sub-themes were formed, namely: *simple and flexible use of simulation software* and *visual representation of complexity*. These sub-themes have been analysed and discussed from Section 4.3.2.1.

4.3.2.1 Simple and flexible use of simulation software

There were 24 of 69 students (35%) via SoS, four of six students (67%) via one-to-one interviews, six of 14 lecturers (43%) via LoS and two of three lecturers (67%) via one-to-one interview besides those who participated on focus group interviews who commented about the simplicity and flexibility of using simulation software (see Table 4.6).

Table 4.6

Number of Participants Against Those Who Commented About the Emerged Themes for RQ2

	Emerged themes from RQ2								
	Number of participants	commented about simplicity	%age	commented about effectiveness for AB, DE & DE	%age	commented about satisfaction	%age	commented about challenges of SS	%age
SoS (*Perceptions of SS*)	69	24	35	29	42	19	28	11	16
Stud 1-2-1	6	4	67	2	33	5	83	3	50
LoS (*Perceptions of SS*)	14	6	43	7	50	-	-	-	-
Lec 1-2-1	3	2	67	2	67	2	67	2	67

Note: Table 4.6 shows the number of different data collection methods (i.e. SoS, LoS, students' one-to-one interviews, LoS, and lecturers' one-to-one) as shown on the first right column; then the number of participants on the corresponding data collection method followed by the number of participants who commented about simplicity of simulation software (SS), effectiveness of SS, satisfaction of SS and challenges of SS and their percentages respectively.

When students and lecturers were asked to explain their perceptions of using simulation software to facilitate their application of CT skills, there were 41 of 52 aggregate codes (79%) in relation to *simple and flexible use of simulation software*. Students' and lecturers' description of "*simplicity of using simulation software*" related a lot to the concepts of abstraction, decomposition and generalisation. For instance, they commented that simulation software help students to simplify unnecessary details which tallies with the concept of abstraction (Bocconi et al., 2016). They commented that simulation software help students to decompose large complex systems to the level that they can solve concurring with the concept of decomposition (Angeli et al., 2016). Additionally, they commented that simulation software makes it easy and simple to see trends and patterns of the network configuration to aid in problem-solving concurring with the concept of generalisation (Csizmadia et al., 2015). They furthermore added that simulation software makes it easier and simpler for students to apply networking theories learnt in lessons concurring with (Zhang et al., 2012). Therefore, they recommended that simulation software provides a simpler platform to apply and improve students' CT skills. These were some of the examples of what students and lecturers commented:

> definitely there and in a lot of those tools and simulations and abstractions is what they are largely built, and you abstract the way protocols are working – you know you abstract the model to the level that replicate the full operation of LAN – and so you are constantly conscious of abstracting a solution to a level that is quick and easy to build a solution (*Lec3 – one-to-one interview*)

> "packet tracer allows the student to apply decomposition into their design when tackling large complex network design" (*LoS*)

> "By providing a generalised image, it is easier to look at problems from multiple points of view" (*SoS*)

Students believe that the simplicity of using simulation software provides an easy platform to stir up their imagination and innovation to look at problems from multiple point of view which remain consistent with the concept of generalisation in identifying different patterns to solve problems. Cross-referencing students' responses with one-to-one interviews conducted by

lecturers and students, their inferences remained the same. Lecturers believe that using simulation software makes students easy and simple to constantly become aware of abstracting solutions when they design and troubleshot network systems. Furthermore, lecturers believe that via simulation software it is simpler to decompose a complex problem to the level that it is easier to solve a problem. For instance, one of the other lecturers commented that:

> "Simulations enables you much more to take and see, be able to examine the whole picture and able to split smaller chunks at time which would on average take more than times longer" (*Lec2 – one-to-one interviews*)

Concurring these ideas, students via one-to-one interviews commented that it is much easier for them to develop their CT skills via simulation software than on real physical devices. Students said that simulation software makes tasks less complicated, they do not waste time to design basic to complex network systems. For examples, students via simulation software simply drag and drop simulated devices and start configuring their networks.

Results from Likert scale shown on Table 4.7 and Table 4.8 show students' and lecturers' perceptions of using simulation software to facilitate the application of students CT skills. For instance, 66% of students and 93% of lecturers agree that simulation software provides more flexibility than real physical devices in designing complex network systems. Furthermore students commented that simulation software provides flexibility and freedom to design limitless network systems without restriction to physical hardware devices or their geographical location. For instance, one of the students via SoS made this comment:

> If i was given physical hardware i wouldn't be able to take my time to fully understand it as if i had to move around a lot such as from university to my home i'd have to take a lot of equipment with me as well as set up my home network to ensure that i could work on the equipment and configure them however in simulation software i can research and thoroughly understand the work whilst only needing one specific software and a way to transport the file if necessary (*SoS*)

Students' and lecturers' perception of the simplicity and flexibility of using simulation software to understand complex computer networks remain consistent with literature discussed in Section 2.7.2. Students in this study were not taught the concepts of CT however, their perceptions of how simulation software facilitates their problem-solving skills suggested that they are able to apply the concepts of CT.

Table 4.7

Perceptions of Students (N = 69) on the Use of Simulation Software in Facilitating Computational Thinking from SoS

Q. No on SoS	Likert scale question	Strongly agree	Agree	Combined (*Strongly agree and Agree*)	Strongly disagree	Disagree	Combined (*Strongly disagree and Disagree*)
18	Simulation software provides a good platform in facilitating my computational thinking when designing computer networks	**26** (38%)	**20** (29%)	**46** **67%**	**7** (10%)	**16** (23%)	**23** **33%**
19	Simulation software provides more flexibility than real physical device in designing complex networks which can help in developing my computational thinking	**19** (28%)	**26** (38%	**45** **66%**	**6** (9%)	**18** (26%)	**24** **35%**
20	Simulation software provides a good platform to troubleshoot network design thereby develops my computational thinking skills	**13** (19%)	**36** (52%)	**49** **71%**	**2** (3%)	**18** (26%)	**20** **29%**
21	Simulation software does not provide real practice and experiences as compared to physical devices therefore it is not ideal to develop my computational thinking	**7** (10%)	**21** (30%)	**28** **41%**	**13** (19%)	**28** (41%)	**41** **59%**

Note: Table 4.7 shows the results of students' perception on the use of simulation software in facilitating their CT. Questions from 18 to 21 retrieved from SoS, were based on Likert scale in which students indicated whether they agree or disagree to the questions given to them. Their responses were combined as shown under (**Combined** (***Strongly agree and Agree)***) and (**Combined** (***Strongly disagree and Disagree)***) respectively. The numbers without a percentage (%) represent the number of students who opted for that answer while the number in percentage shows the number of students in percentage who opted for that answer.

Table 4.8

Perceptions of Lecturers (N = 14) on The Use of Simulation Software in Facilitating Computational Thinking from LoS

Q. No on LoS	Likert scale question	Strongly agree	Agree	Combined (*Strongly agree and Agree*)	Strongly disagree	Disagree	Combined (*Strongly disagree and Disagree*)
21	Simulation software helps in facilitating students' computational thinking when designing computer networks	**3** (21%)	**10** (71%)	**13** **93%**	**0** (0%)	**1** (7%)	**1** **7%**
22	I don't think simulation software provides a platform to develop students' computational thinking in designing computer networks	**1** (7%)	**3** (21%)	**4** **29%**	**6** (43%)	**4** (29%)	**10** **72%**
23	Simulation software provides a good platform to troubleshoot network design and therefore develops students' computational thinking	**4** (29%)	**9** (64%)	**13** **93%**	**0** (0%)	**1** (7%)	**1** **7%**

Note: Table 4.8 shows the results of lecturers' perception on the use of simulation software in facilitating students' computational thinking. Questions from 21 to 23 retrieved from LoS, were based on Likert scale in which lecturers indicated whether they agree or disagree to the questions given to them. Their responses were combined as shown under (Combined (*Strongly agree and Agree)*) and (Combined (*Strongly disagree and Disagree)*) respectively. The numbers without a percentage (%) represent the number of lectures who opted for that answer while the number in percentage shows the number of lecturers in percentage who opted for that answer.

4.3.2.2 Visual representation of complexity

There were 11 of 52 codes (21%) in relation to *visual representation of complexity*. Students' and lecturers' perceptions were that simulation software help students to have a visual (graphical) representation of a complex problem that would not be possible with physical hardware devices. They believe that the visual representation of complex problems via simulation software makes it easier for students to understand the problem and decompose to solvable tasks. Other students commented that they are visual learners therefore they find simulation software providing a better platform to visualise, learn and abstract complex problems. In essence students believe that the use of simulation software helps in the process of abstracting the complexity of the network that they can easily understand. For example, two students, via focus group interview said:

> As for me, simulation software, also learning very well. For instance, am a visual learner so I like seeing things to understand better. Therefore, I feel confident and comfortable when I use simulation because it gives me that visual representation of what am working on. Otherwise implementing on real device (I don't know, It's just me), I have no control over it (*PGstud6 - PGFG*)

> "I feel confident and comfortable when I use simulation because it gives me that visual representation of what am working on" (*PGstud5 - PGFG*)

Students went on to comment that visual representation of complex problem help them to identify patterns (i.e. similarities, differences, commonalities) leading to solving problems and developing their CT skills. For examples, one of the students via SoS furthermore commented that:

> "Visual representations of data such as charts and graphs make it easy to see trends and patterns with the network to aid in problem solving" (*SoS*)

Additionally, one of the students when reflecting upon the problem-solving tasks they were given to work on via simulation software, he acknowledged that without the visual

representation of the problems (e.g. interpreting routing table to design a topology) would not have been possible to fix the problems. This is how he commented:

> "The first thing was to have a visual topology of the entire network based on the routing table given. When I had a visual presentation of that, then I was able to fix everything" (*PGstud2 – PGFG*)

Students' comments concur with lecturers' response in emphasising that simulation software provides a platform upon which students can visualise the entire picture of a problem to easily decompose and solve it. This is how one of lecturers commented via one-to-one interview:

> "It [simulation software] enables them to see the whole pictures and enables them to concentrate and splitting them unto tasks" (*Lec2 – one-to-one interviews*)

It became apparent that students' and lecturers' perceptions of using simulation software facilitates the application of CT skills easily. There are no hardware or geographical limitations to solve computer network problems via simulation software making it flexible and readily available to apply CT skills. Furthermore, simulation software provides a platform on which students can visualise a complex problem, making it easy and simple to abstract and decompose it. Additionally, visual representation of complex problem via simulation software help students to simply identify patterns of the problems leading to the application and development of their CT skill.

The results were consistent with literature (e.g., Hwang et al., 2014; Ruiz-Martinez et al., 2013) indicating students' and lecturers' opinions and experiences in the simple and flexible usage of simulation software. Simulation software provides a broader and deeper environment where students can build, modify and test their design thereby enhancing their problem-solving skills as discussed is Section 1.6 of the Introduction chapter. Many studies including those of (Galan et al., 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. The inherent flexibility of simulation tool means that it provides a platform upon which students can build almost any structure thereby extending their inventiveness and innovation (Ruiz-Martinez et al., 2013). Lecturers also believe that via simulation software students are constantly conscious of abstracting a complex network making it easy and quicker to solve problems. These findings remain consistent with the studies of many scholars (e.g., Exposito et al., 2010; Ruiz-Martinez et al., 2013; Zhang et al., 2012) showing the simplicity of using simulation software to enhance students on problem solving (Zhang et al., 2012) thereby enforcing the application their CT skills (Selby & Woollard, 2010). Selby and Woollard (2010) argue that simulation software can be used to manipulate abstraction by identifying anomalies, formulating hypothesis, coming up with emerging solutions and formulating patterns, similarities and commonalities in solving complex problem, thereby enhancing the application and development of CT skills.

4.3.3 Effectiveness for using simulation software for abstraction, decomposition and generalisation

The second theme that emerged in addressing the second research question was *the effectiveness of using simulation software for abstraction, decomposition and generalisation* with an aggregate of 49 codes (see Table 4.5). Table 4.6 provides the number of participants from SoS, LoS and one-to-one interviews who commented about the effectiveness of using simulation software and their percentages against the population sample of those who participated in this study. There were two sub-themes which emerged, namely: *provides effective network design platform* with 27 of 49 codes (55%) and *complex problems are broken down and understood* with 22 of 49 codes (45%). These two sub-themes are shown in Table

4.5 and discussed from Section 4.3.3.1 with coding occurrences and sources provided in Appendix 4.2.

4.3.3.1 Provides effective network design platform

The first sub-theme focused on the effectiveness of simulation software in providing platform for designing simulated networks. Similar to students' and lecturers' comments made in Section 4.3.2.1, they believe that simulation software provides an effective platform for students to design various and complicated simulated computer networks thereby facilitating the application of the concepts of abstraction, decomposition, generalisation in problem solving. They emphasised that the graphical representation of problems via simulation software, make students to effectively abstract the complexity of network problem to the level that they can understand (thus the concept of abstraction). In other words, they commented that students are able to understand complex network problems via simulation software than on physical hardware device which concur with Repenning et al. (2016) arguing that simulation, as a CT tool, can help in visual thinking where narrative approaches are not possible. Students and lecturers furthermore, mentioned that via simulation software students are able to break down pieces of network requirements into smaller manageable tasks which can be solved independently before constructing them together to form a complex network infrastructure (thus the concept of decomposition). These were some of students' and lecturers' comments:

> "I have learnt, to focus on building one part of a network at a time, so when it has come to build an entire network I am able to put all of those skills together, to build a whole network" (*SoS*)
>
> "It [simulation software] can help by showing students how all the components link together into a larger system" (*LoS*)
>
> "simulation software provides the best support for any learner to understand and learn about network design and also help in developing computational thinking" (*PGstud2 – one-to-one interview*)

Additionally, when it comes to troubleshooting network problems, lecturers commented that the visual representation of the network problems via simulation software helps them to easily identify patterns of similarities, differences or commonalities in solving problems (thus the concept of generalisation). For example, one of the lecturers via LoS said: "*[simulation software] also enable the easy testing of similar scenarios to examine commonalities".* Furthermore, Table 4.7 and Table 4.8 show that 71% of students and 93% of lecturers respectively believe that simulation software provides a good platform to troubleshoot network design thereby facilitate the application of CT skills. Moreover, 11 of 14 lecturers (79%) via LoS, indicated that simulation software can facilitate the demonstration of abstraction, decomposition and generalisation (see Appendix 4.5, question 18). Emphasizing on the effectiveness of simulation software in facilitating the application of CT skills and how it helps in building up students' confidence when working on complex network systems, students via one-to-one and focus group interviews made these comments:

> I guess you develop your computational thinking much better on simulation software than real kit […] for instance, on simulation you can build as big and complex network as you want there is no restrictions in terms of costing of the actual equipment so you can do more stuff in a sense that you can make your network as large as you want and use your computational thinking as over and over and over as you wish (*UGstud3 - UGFG*)

> The other major part of simulation software is that it helps in developing so many skills and also helps in developing confidence. You can also test a lot of things prior to deploying them on the real physical equipment. Therefore, it is very important that people learn about computational thinking though others really do that but they don't know that what they are doing is actually applying their computational thinking skills (*UGstud9 – one-to-one interview*)

Additionally, lecturers mentioned that simulation software is effective in helping lecturers themselves to quickly demonstrate some challenging networking concepts to students which would normally be difficult or take long on physical devices. Another lecturer also highlighted that simulation software is effective because student can break, build, modify and test their

network designs again and again thereby facilitating the application of their CT skills concurring with literature discussed in Section 1.6. Furthermore, lecturers highlighted that simulation software makes students always conscious of abstracting their network design thereby enhancing learning and enforcing problem-solving skills which involves the application of CT skills. These are examples of some of the comments made by lecturers via LoS and one-to-one interviews:

> "it's not possible to conduct a study on a lab with 1000 devices, but in simulation we can study for N devices and designing a complex MESH is very hard in real lab, but in simulation its possible" (*LoS*)

> It enables you do things that you could not be able to do in a classroom because it is much more easier to set up things and tear them down when you don't get what you expected. You have more freedom in your design especially with things like packet tracer, GNS, OPNET but at least packet tracer and GNS3 from a teaching point of view you can do things that you would not be able to do in the physical (*Lec2 – one-to-one interview*)

> The advantage of using emulation and simulation software is that you can build, modify, test, build, modify, test again and again … and you can improve on what you do a lot easier than you would do in the real network and not only that but you can go down and see how the actual thing operate and behave and also monitor what is happening which is not easy without specialist tools on the real network (*Lec3 – one-to-one interview*)

Furthermore, lecturers believe that simulation software provides students with simple and flexible unlimited simulated devices and protocols that they can design limitless networks from basic to complex network designs. Additionally, lecturers indicated that students are not constrained with the physical environment of network devices since they can work anywhere as long as they have simulation software installed on their laptops concurring with literature discussed in Section 2.7.2. Therefore, lecturers believe that students can understand the abstraction of a complex network, decompose the problem to smaller solvable solutions and apply their generalised ideas based on the theories, prior knowledge and experiences acquired in class. For example, Table 4.8 shows that 93% of lecturers believe that simulation software helps in facilitating students' CT skills when designing computer networks.

These results concur with many scholars (e.g., Janitor et al., 2010; Yalcin et al., 2015; Zhang et al., 2012) arguing that students can design complex and limitless networks on simulation software much better and easier than on physical hardware platform because of cost and inflexibility of physical hardware devices. Furthermore, students' and lecturers' perceptions of this study have shown that simulation software provides an effective platform upon which students can design complex network problems thereby enhancing their problem-solving skills leading to the application of their CT skills. Students acknowledge that simulation software can help them design, build, configure and troubleshoot complex networks which concur with the study of Wannous and Nakano (2010). They furthermore indicated that simulation software makes complicated network design easier that students and computers can understand to solve. Additionally, students acknowledged that simulation software provides a safer environment to solve network problems using CT strategies.

4.3.3.2 Complex problem are broken down and understood

The second sub-theme focused on the effectiveness of simulation software in breaking complex problems to the level that they can understand, solve and evaluate separately. This sub-theme re-enforced the fact that students' and lecturers' main understanding of CT in this study, as discussed in previous sections, is on the concept of decomposition. They often referred to almost everything concerning CT, as the ability of breaking down complex problem to a suitable level. In explaining their perceptions of using simulation software via SoS, students used phrases, such as "*makes complex problem easily solvable*", "*highlights problems which could otherwise be overlooked*", "*gives wider insight about the problem by breaking it down*". Students furthermore, highlighted that they can bring an abstract of a network design to concrete via simulation software and able to test every area of their design before deploying it.

> Simulation software is a good platform to develop your computational thinking because you can do everything you want, and its much different from hardware which you need to bring together. You can

> bring an abstract to concrete by testing every area of your design before actually deploying it (*PGstud9 – one-to-one interviews*)

Examining students' academic background based on their profile provided in Section 3.7 of the Methodology chapter and the teaching and learning style, which focuses on practical tasks than theories as discussed in Section 3.5 of the Methodology chapter, the inferences resonated very well with their perceptions of using simulation software to apply their CT skills. Students often use simulation software to understand the concept of networking before working on the physical hardware devices. Therefore, they are used to working on different simulation software and physical hardware devices in designing and troubleshooting network systems as part of developing their practical skills. When students and lecturers via SoS and LoS were asked to mention the type of simulation software which they are familiar with, 69 of 69 students (100%) and 12 of 14 lecturers (86%) mentioned Cisco Packet Tracer (see Appendix 4.4, question 15 and 4.5, question 17 respectively). However, these students never designed or troubleshooted network systems from the CT point of view. They often solved problems by trial and error as per the following comments:

> 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 (*LoS*)

> "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" (*PGstud3 – reflective report*)

> "Anyway, I don't dictate to students how they need to arrive at the solution. However, what I do is to give them particular problems which, by the end of their degree it should change them" (*Lec1 – one-to-one interviews*)

From Section 4.3.2 results have shown students' and lecturers' positive experiences in using simulation software to facilitate the application of CT. Students and lecturers have shared their opinions and experiences on how simplistic and flexible simulation software is compared to

physical hardware in facilitating CT. Both lecturers and students acknowledged the effectiveness of using simulation software in facilitating the application of CT skills. Furthermore, Table 4.7 and Table 4.8 show that 71% of students and 93% of lecturers respectively believe that simulation software provides a good platform to troubleshoot network design thereby facilitate the application of CT skills. Students' responses from SoS have indicated that simulation software helps them to look at a greater picture of a problem, understand wider insight about the problem, help visualise complex problem through graphical user interface (GUI) and help them guide through network design and management. Therefore, students' and lecturers' perceptions of the use of simulation software in this study shows that skills-based learning approach, which focuses on developing students' networking skills in designing and troubleshooting network systems via simulation software, facilitates the conceptualisation of CT.

Students' and lecturers' opinions and experiences are consistent with literature review (e.g., Galan et al., 2009; Hwang et al., 2014; Su et al., 2013) arguing that simulation software provides a highly realistic way of teaching computer networks.

4.3.4 Satisfaction of using simulation software

The third theme that emerged in addressing the second research question was *satisfaction of using simulation software* with 58 codes from 12 sources comprising SoS, LoS, students' and lecturers' one-to-one interviews, focus groups and reflective reports (see Table 4.6 and Appendix 4.2). This theme had the highest coding occurrences than any of the themes in answering the second research question. Although it is a dedicated theme, evidences discussed in Sections 4.3.2 to 4.3.3 contribute strongly to the satisfaction of students' use of simulation software to facilitate their application of CT skills. Moreover, thus the reason why this theme had the highest number of coding occurrences. For instance, the fact that students and lecturers

explained the simplicity, flexibility and effectiveness of using simulation software to facilitate the application of abstraction, decomposition and generalisation suggested that students and lecturers were satisfied. Therefore, this theme comes as a summary of the key themes already discussed in Sections 4.3.2 to 4.3.3. The keys areas have been summarised below:

Students and lecturers were positive about the use of simulation software in facilitating students CT skills. They commented that simulation software helps students to abstract complex network problems because it provides visual representation of a problem. Lecturers find it easier and simpler to demonstrate some difficult networking concepts to students which could otherwise be difficult via physical hardware devices. Literature (e.g. Repenning et al., 2016) has shown that visual CT tools improve students' thinking skills in areas where narrative text would not be possible. Furthermore, scholars (e.g., Ruiz-Martinez et al., 2013; Sun et al., 2013) have argued that networks are abstracts and can be easily explained by the use of practical demonstration which simulation software is able to facilitate that easily and effectively (Zhang et al., 2012). For example, two students via SoS, said:

> "It [simulation software] definitely makes complex details more understandable, such as through a GUI [graphical user interface] which makes the code with a simple interface. This allows inexperienced users to still carry out complex tasks" (*SoS*)

> "Very good, [simulation software] has given me the chance to understand some of the ins and outs of the systems using a virtualised version" (*SoS*)

Furthermore, all students and lecturers on one-to-one interviews expressed their satisfactory opinions that it is easier and simpler to break, build, test, again and again their network design via simulation software than using physical hardware devices, remaining consistent with literature (e.g., Janitor et al., 2010; Zhang et al., 2012). They believe that students can do more tests via simulation software thereby apply and develop their problem-solving skills. Additionally, students and lecturers often commented that via simulation software, students are

able to build their confidence to handle different levels of computer network designs (i.e. from basic to complex networks). They emphasised that because simulation software provides visual representation of complex problems, students can identity different patterns of similarities, differences, thereby giving them multiple view of to solve problems. Students and lecturers had so much to say on their satisfaction of using simulation software in designing and troubleshooting network systems. It was evident that lecturers and students in this study base their teaching and learning styles respectively on problem-solving skills.

Furthermore, students via focus groups commented that simulation software gives them over 70% of knowledge on how network systems are designed and implemented in real life. They believe that simulation software supplements their theoretical knowledge and practical experiences in designing and troubleshooting network systems. Students believe that using simulation software is less stressful than using physical hardware devices. Concurring students' idea that simulation software is less stressful, one of the lecturers via SoS commented that:

> "students go directly to the problem-solving stage instead of spending a lot of time configuring and cabling a system" (*SoS*)

Overall students often used words, such as *"it is wonderful", "very good and helpful", "very handy", "can develop more problem-solving skills", "provides clear representation of problems"* in expressing their satisfaction of using simulation software in designing and troubleshooting network systems. Students' and lecturers' perceptions clearly showed that simulation software is useful and helpful in improving their practical networking skills and mastering commands which they would use in configuring the physical hardware devices.

In this study, students were not taught the concepts of CT however, their explanation to their perceptions of using simulation software when designing and troubleshooting network systems indicated that they can apply the concepts of CT via simulation software.

4.3.5 Challenges of using simulation software

There were some challenges of using simulation software which students and lecturers commented about as they were explaining their perceptions of using simulation software to facilitate the application of CT skills. All ambiguous responses were discarded, and the remaining responses seem to highlight challenges of using simulation software as shown in Table 4.5. There were 21 codes suggesting various challenges and are discussed below:

One lecturer via one-to-one interview commented that students may lose the experience of working on physical devices because simulation software creates a false image of physical network. The other lecturer via one-to-one interview also commented that troubleshooting is problematic on simulation environment and does not allow students to think deeply. Furthermore, one other lecturer via SoS said that with simulation software students may not be able to observe and measure traffic passing through their simulated network therefore they lose some of the aspects which they would usually learn via physical hardware devices. One other lecturer, on one-to-one interview emphasised that simulation software focuses on only abstraction than any other concepts of CT. On the other side, students main challenge was that there are some *commands* (syntax of codes used to configure network devices) which cannot be applied via simulation software but rather on physical hardware devices. However, analysing the challenges which these students and lecturers commented, they do not show the limitations in applying the concepts of CT via simulation software. For instance, lecturers' major concern was that via simulation software, students may lose the objectivity of practical experiences in using real physical equipment which they would encounter in the industry.

Students' main concern was that there are some commands which may prevent them from successfully designing and troubleshooting network system. None of these limitations prevent the application of the concepts of CT on simulation software.

However, majority of students and lecturers interviewed on focus groups and one-to-one interviews, as evidenced from Section 4.3.2 to 4.3.4, strongly believe that simulation software help them understand complex problems (thus the concept of abstraction); help them to break problems to small solvable tasks (thus the concept of decomposition); help them to identify trends, patterns, similarities, connections of problems and able to visually see a complex problem which help them to easily and quickly solve the problem (thus the concept of generalisation). Therefore, according to the overall students' and lecturers' perceptions, they believe that simulation software can facilitate the application of their CT skills.

4.3.6 Summary of findings in relation to the second research question

The results have shown that students and lecturers believe that the use of simulation software facilitates the application of students' CT skills. The simplicity and flexibility offered by using simulation software in designing simulated networks were endorsed as effective means of providing a good platform to facilitate the application of students' CT skills. For instance, 66% of students agreed that simulation software provides more flexibility than real physical device in designing complex networks which can help in developing their CT skills. Students and lecturers mentioned how simple and easy they can break, build, modify, test their simulated network designs on simulation software than on physical network designs (concurring with their understanding of the concept of decomposition in Section 4.2.2.2.1 in answering the first research question). Additionally, students and lecturers commented that students can design and troubleshoot their networks via simulation software wherever and whenever they want

without geographical restrictions; thereby providing unlimited opportunities to practice their problem-solving skills, leading to the application of CT skills.

The simplistic and flexible use of simulation software leads students to easily and quickly abstract complex networks to the level they can understand, (concurring with their understanding of the concept of abstraction in Section 4.2.2.1.1 in answering the first research question); identify trends, patterns, similarities and commonalities in problem-solving (concurring with their understanding of the concept of generalisation in Section 4.2.2.3.2 in answering the first research question). Students can easily apply their prior knowledge and experiences in solving problems. Furthermore, students and lecturers indicated that simulation software provides a visual representation of problems which enables them to have a wider insight to problems leading to multiple views of problem-solving.

Lecturers believe that simulation software stirs up students' analytical and critical ideas causing them to be innovative, creative and imaginative in designing and troubleshooting network systems.

Only a few students and lecturers (i.e. three students and two lecturers via one-to-one interviews and six students via SoS based on data provided in Appendix 4.2) felt that simulation software may not be effective in designing network systems because students may miss out the industrial experience of working on physical network devices; and that there are some commands which cannot be used on simulation software. However, these limitations do not have direct effect on the application of CT skills. Overall, the results have shown that students and lecturers are satisfied that simulation software provides an effective platform for students to apply CT skills. Students and lecturers believe that students via simulation software can understand an abstraction of a complex network, decompose the problem to smaller solvable solutions and apply their generalised ideas based on the theories, prior knowledge and

experiences. These findings remain consistent with literature (Angeli et al., 2016; Csizmadia et al., 2015) demonstrating the effectiveness of facilitating the concept of abstraction, decomposition and generalisation via simulation software.

Section 4.5 leads into the practical application of how simulation software may facilitate the application of students CT skills.

4.4 Findings in relation to the third research question ('How might the use of simulation software facilitate the application of students' computational thinking?')

4.4.1 Introduction

The third research question was a follow-up on the second research question to investigate ***how*** students could demonstrate the use of simulation software to facilitate the application of CT skills. Therefore, six postgraduate students, who also participated in the investigation of the first and second research questions were given three different problem-solving tasks to solve via Cisco Packet Tracer simulation software for a period of six weeks per each task. After each problem-solving task, students, as an individual, produced reflective reports explaining their thought processes, strategies and challenges in solving problems (see Appendix 4.6 for an example of a student's reflective report). Besides students' reflective reports, they participated in a follow-up focus group which further explored their understanding of CT, experiences and perceptions of using simulation software. Detailed explanation of the focus group interviews and the three problem-solving tasks respectively have been provided from Section 3.8.2 of the Methodology chapter.

Carrying on from the findings of the students' and lecturers' understanding of CT as illustrated in Figure 4.2 and their perceptions of using simulation software to facilitate the application of CT as summarised in 4.3.6, four themes were formed, namely: the demonstration of *abstraction*, *decomposition*, *generalisation* and *problem solving* which are shown in Table 4.9 and discussed from Section 4.4.2.

Table 4.9

Themes in Answering the Third Research Question

Themes	**Code Occurrences**	
	Sources	Codes
Demonstration of abstraction (AB)	12	21
Demonstration of decomposition (DE)	14	29
Demonstration of generalisation (GE)	15	41
Demonstration of problem solving that involve AB, DE & GE	17	45

Note: Table 4.9 shows themes formed from codes related to the third research question. **Sources** shows the number of different sources such as students reflective reports, video clips and focus group interviews based on problem-solving task1, task2, tasks3 where the codes have been generated from. **Codes** mean the actual number of coding occurrences from sources (see Appendix 4.3 for more details).

Additionally, the number of students who participated in each problem-solving task (*PsTask1, PsTask2* and *PsTask3*) against those who commented about the concepts of CT focused in this study, followed by the percentage rating have been provided in Table 4.10.

Table 4.10

Number of Participants Against the Reported Themes for RQ3

	Themes in Investigating RQ3								
	Number of participants	Reported about AB	%age	Reported about DE	%age	Reported about GE	%age	Reported about PS	%age
PsTask1	6	2	33	2	33	6	100	5	83
PsTask2	6	3	50	4	67	2	33	3	50
PsTask3	6	3	50	4	67	2	33	4	67
ORR on PsTasks	6	4	67	4	67	5	83	5	83

Note: Table 4.10 shows the number of different problem-solving tasks and students' overall reports (i.e. PsTask1, PsTask2, PsTask3 & ORR on PsTasks) as shown on the first right column; then the number of participants on the corresponding problem-solving task followed by the number of participants who reported and commented about abstraction (AB), decomposition (DE), generalisation (GE) and problem solving (PS) and their percentages respectively (see Appendix 4.3 for more details)

4.4.2 Ways that simulation software facilitate students' computational thinking

4.4.2.1 Demonstration of abstraction

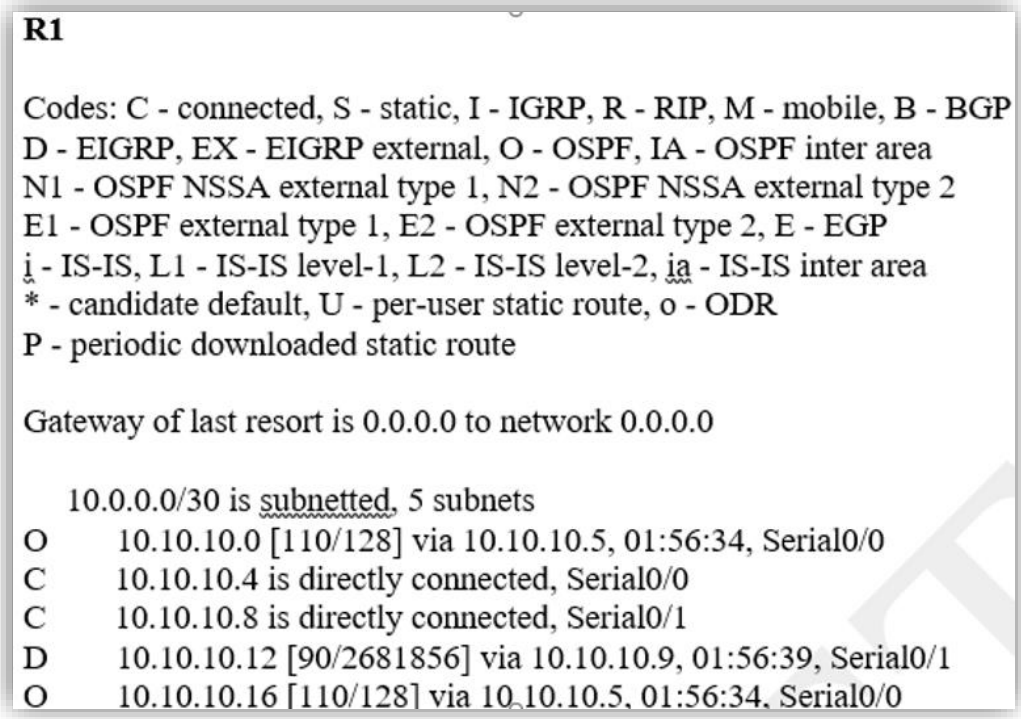
In the first problem-solving task, students as an individual, were supposed to show their CT skills via simulation software to abstract the given routing table (see Appendix 3.07) to form a concrete simulated enterprise network infrastructure. As part of the demonstration of the concept of abstraction, students were initially supposed to focus on identifying the important details, such as remote networks, directly connected routes, IP addresses with their appropriate subnet masks, routing protocols, misconfigured components and decide on the appropriate commands to configure their simulated devices leading to a concrete simulated enterprise network infrastructure. In the second problem-solving task, students were given an abstract of network design requirements (see Appendix 3.08). Students, as an individual, were supposed

to ‘interpret’ those requirements by hiding all unnecessary technical details from the user but design functional simulated enterprise network infrastructure. Students were furthermore meant to decide which IP addresses and routing protocols to implement for the effective operation of their simulated enterprise network infrastructure. Therefore, students were meant to make appropriate judgement of what important details to focus on and which ones to hide, which concur with the concepts of abstraction (Bocconi et al., 2016, Wing, 2008). Finally, in their third problem-solving task, students were only meant to apply security features to their existing simulated enterprise network infrastructure therefore there were little points of abstraction to investigate, such as identifying security protocols and areas to secure (i.e. hide).

To understand their thought processes, via reflective reports and video clips, each individual student was required to explain and show clear strategies applied on how he designed his simulated enterprise network infrastructure. As shown in Table 4.9, there were 21 codes from students’ reflective reports and video clips showing how the concept of abstraction was applied via simulation software. Table 4.10 shows that out of the six students who participated in the three problem-solving tasks, two (33%), three (50%), and three (50%) students commented about the concept of abstraction based on their solutions to PsTask1, PsTask2 and PsTask3 respectively. These results indicate an average percentage of 44% of students who commented about the concept of abstraction based on all the three problem-solving tasks. Although few students commented via their reflective reports about how they applied the concept of abstraction when solving their problems, all students via their video clips successfully demonstrated their application of abstraction. This shows that although all students were able to apply the concepts of abstraction but most of them were not conscious neither did they know the concepts of CT they were applying for.

In some instances, particularly on problem-solving task one (PsTask1) students applied the concepts of abstraction and generalisation concurrently which concurs with literature (e.g. Angeli et al., 2016) arguing that the concepts of abstraction and generalisation often work together to provide greater usability. For example, without prior knowledge and experiences of computer networking, students could neither understand nor be able to trace the routing table for PsTask1 and interpret the network requirements for PsTask2 to build a simulated enterprise network infrastructure.

Reading through students' reflective reports and watching their video clips, it showed that they all were able to apply the concepts of abstraction to build concrete and functional simulated enterprise network infrastructures (see sample simulated enterprise network infrastructure in Appendices 4.7 and 4.8). For example, based on PsTask1, students via simulation software were able to abstract the routing table by associating the IP addresses with their subnets IDs, subnet masks, interface type (fast ethernet or serial, or sub-interface), special characters such as "C", referring to directly connected networks, "O" and "D" referring to OSPF and EIGRP routing protocols respectively (as briefly shown in *Figure 4.3*).



R1

Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

10.0.0.0/30 is subnetted, 5 subnets
O 10.10.10.0 [110/128] via 10.10.10.5, 01:56:34, Serial0/0
C 10.10.10.4 is directly connected, Serial0/0
C 10.10.10.8 is directly connected, Serial0/1
D 10.10.10.12 [90/2681856] via 10.10.10.9, 01:56:39, Serial0/1
O 10.10.10.16 [110/128] via 10.10.10.5, 01:56:34, Serial0/0

Figure 4.3. An Extract from Routing Output (see Appendix 3.07 for details)

Note: Figure 4.3 shows different networks which are advertised from this Router (R1). Some networks are directly connected to the Router as denoted by "C" while others are advertised to the router because they can be seen from remote using routing protocol denoted by letters "O" and "D". Each directly connected network shows the type of interface they are connected to (e.g. serial0/0) and the remotely advertised networks can be traced by the next hop of IP address denoted by "via" (e.g. via 10.10.10.5)

Additionally, via their reflective reports two of the six students explained their thought processes in abstracting the routing table. They pointed out that they were able to focus on important details on both the routing table and network requirements to form a simulated network infrastructure. Another student via focus group commented that he used paper and pencil to sketch his topology in trying to interpret the routing table and network requirements to form simulated enterprise network infrastructure. For examples, these were some of the students' comments:

> "The directly connected routes in each routing table of the routers which is denoted by the letter C was the key to tracing and reproducing the network topology" (*PGstud1 - reflective report PsTask1*)

> From the routing table, the parameter 'C' indicates the direct connection of the network to the interfaces through that it is discovered that the network address of the serial interface of one router is similar to the network address of the serial interface on other router. This concludes that those two routers are connected with the same network address and all other similar connections are identified and noted down vice versa. (*PGstud5 - reflective report PsTask1*)

> when I approach a complex problem like that, when designing a simulation software like Packet Tracer. Especially like the problem we were given like routing output table of all routers, I first look at all networks which have been advertised, and then I will look at the IP addresses which have been advertised with their networks and then I will be identifying their interfaces, source and destination IP addresses. I will then design on a paper the whole topology according to the interconnected devices, from that point I will be jumping into configurations (*PGstud2 – PGFG*)

Students demonstrated their practical skills in applying the concepts of abstraction via simulation software. This could be because of their learning style which focuses more on developing their problem-solving skills involving designing and troubleshooting network systems as discussed in Section 3.5 of the Methodology chapter. These students also demonstrated how CT skills can be applied on a non-computing platform (i.e. by use of paper and pencil) concurring with the study conducted in New Zealand (http://csunplugged.org/).

When abstracting the network requirements based on PsTask2, three students (see Table 4.10) commented that without prior knowledge and experiences of networking they would not be able to interpret the requirement to form a simulated enterprise network infrastructure. Their previous knowledge and experiences helped them to easily identify key details and hide those which were not essential concurring with the concept of abstraction (Angeli et al., 2016). For instance, these were some of their comments:

> At this point, the necessary components required to objectively design any topology are network devices and accessories, hence routers […], switches […], firewalls […], Workstations […], Wireless AP[…],

> Cables […], Servers […] all the mentioned devices are useful and were utilized to represent and design a network topology based on a simulation platform or on physical set (*PGstud2 overall reflective report*)
>
> In conclusion, the challenges such as choosing the best protocols for connecting LANs together based on their requirement and limitations or configuring some technologies in a way that it shouldn't interfere with other configurations made me not to rely only on my current knowledge and experience and it also maid me to search and read more about the problem which helped me to expand my knowledge and skills (*PGstud6 - reflective report PsTask2*)

Therefore, students' ability to identify appropriate special characters, types of interfaces, subnets and protocols besides prior knowledge and experiences were some strategies they used to design functional simulated enterprise network infrastructure from the abstracts of network requirements. All students, as individuals, managed to produce a simulated enterprise network infrastructure by interpreting the routing table. Figure 4.4 is a sample diagram from one of the students' simulated enterprise network infrastructure that was produced from interpreting the routing table based on PsTask1 showing different subnets and their associated routers and switches (see Appendices 4.7 and 4.8 for more samples).

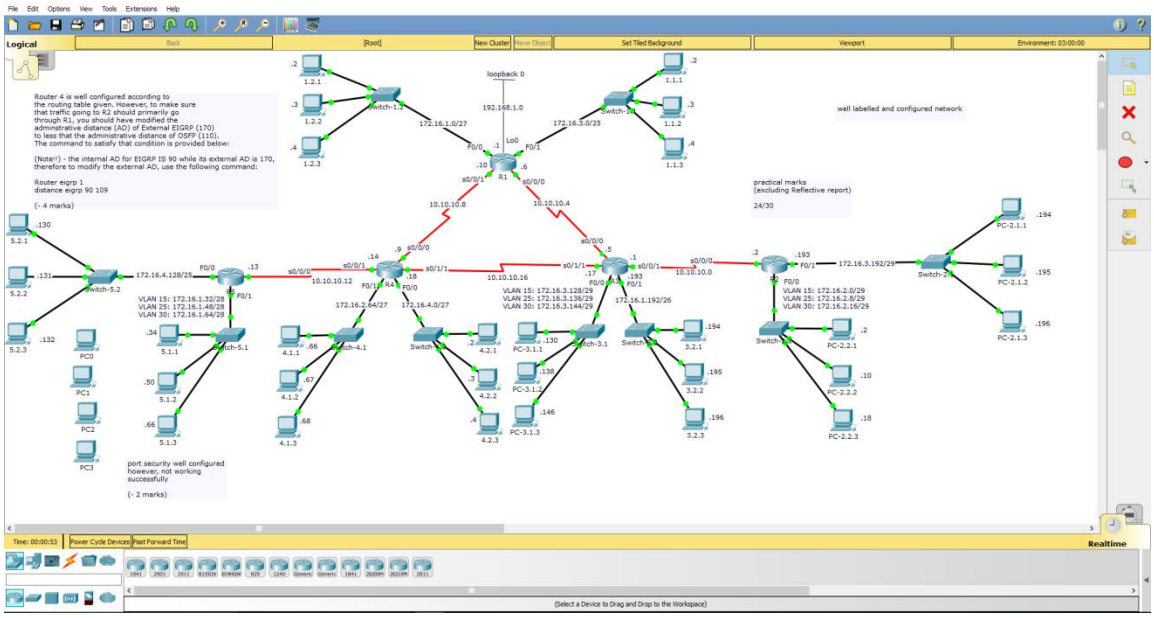


Figure 4.4. Students Sample Simulated Enterprise Network Infrastructure

There was however, one student who struggled in constructing his simulated enterprise network infrastructure because he joined the course late and that he did not have much experience knowledge working with networks. This is what he commented in his reflective report:

> "The majority of my challenges stemmed from the fact that I did not have much experience working with networks before this module" (*PGstud3 – reflective report PsTask1*)

This student's comment concurs with many scholars arguing that prior knowledge and experience in problem solving play important role. For instance, student's ability to identity patterns of similarities, differences, or commonalities are often applied based on one's prior experiences and knowledge – thus the concept of generalisation. Concurring with the argument of Angeli et al. (2016), a student can abstract a complex problem based on his or her prior knowledge and experiences therefore the concepts of abstraction and generalisation work concurrently.

Although via students' video clips, reflective reports and focus group interview show that students managed to apply the concepts of abstraction via simulation software, it appears that students were not conscious that they were applying the concepts of abstraction. For example, based on PsTask1, only two of six students (see Table 4.10) reported about how they applied the concepts of abstraction. The evidence for the rest of the other students was by means of watching their practical demonstration via video clips. However, even via the practical demonstration, students simply explained how they solved the problems without necessary mentioning how the concepts of abstraction helped in solving those problems. It was only made clear by interpreting students' strategies which they adopted in building their simulated networks via video clips that they applied the concepts of abstraction. Students' focus and interest were to get the problems solved. Therefore, they used any means and approach to get to their solutions. Following on that these students were not taught the concepts of CT, they

did not have a structured approach to thinking which could have quickly led them into their solutions and perhaps become more imaginative to show their innovation and creativity as argued by Wing (2006, 2011). Students did not add any innovation or creativity to their simulated enterprise network infrastructure neither did they add any recommendations on their reflective report demonstrating their imaginations. They only solved given problems by trial and error. For instance, in their reflective reports they mentioned that they were using trial-and-error approach to solve problem, others went on to comment that they used Internet to search for solutions. Cross-referencing lecturers' comments via LoS and one-to-one interviews, lecturers mentioned that most of the students solve their problems by trial and error. Additionally, lecturers acknowledged that if students are taught a structured approach to thinking around their problems it would help in their problem-solving approach. Many scholars (e.g., Angeli et al., 2016; Fluck et al., 2016; Goode, Chapman, & Margolis, 2012) as discussed in Section 1.1 of the Introduction chapter, argue that the application of CT to the Computer Sciences students will develop their problem-solving skills to become more creative and innovative that they can produce technologies in the 21st Century. This literature suggest that CT should enable students to go beyond mere solving problems. However, these students in this study who were not taught the concepts of CT only managed to solve problems without going beyond their own creativity and innovations.

In summary, students were able to apply the concepts of abstraction via simulation software when designing and troubleshooting their simulated network design but mainly became evident when observed via their video clips. Students were not conscious that they applied the concepts of abstraction, they only solved problems largely by means of trial and error. They were no evidence of their own creativity nor innovation to their simulated network designs.

4.4.2.2 Demonstration of decomposition

After the application of abstraction discussed in Section 4.4.2.1, students began to breakdown segments of the network problem-solving tasks and constructed their own networks to small solvable designs via simulation software (see samples in Appendices 4.7 and 4.8). At this time, it was easier for students to solve and evaluate areas which needed LANs and VLANs (Virtual local area networks). Additionally, students were able to assign appropriate IP addresses, configured routing protocols and identified areas which were misconfigured. For instance, the screenshot of a video clip shown in Figure 4.5 is an example of how students worked on PsTask2 at a time and finally joined subnets together – demonstrating the concept of decomposition (see Appendix 4.7 for more students' video clips).

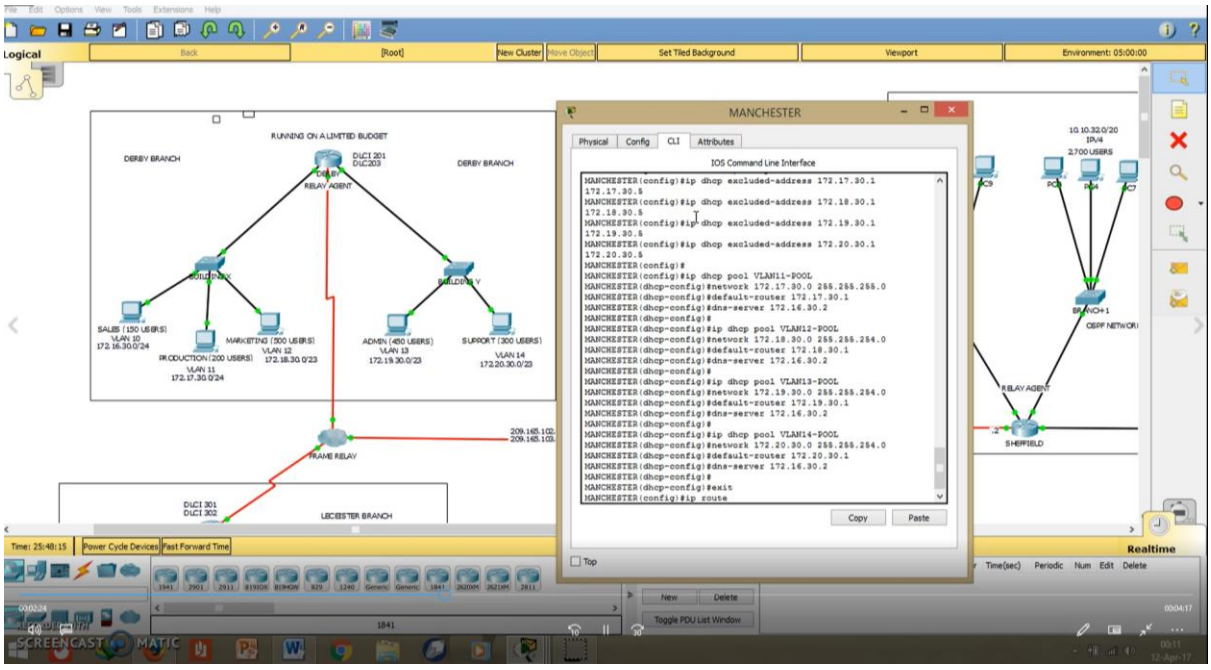


Figure 4.5. Student Sample Enterprise Network Infrastructure from a Video Clip

Note:

Figure 4.5 is a video clip screenshot showing how the student tackled the problem-solving task 2 requirements. The grey square drawings show the individual LAN for this infrastructure. The student showed each LAN was designed separately (i.e. decomposition) before joining them together to complete the full construction and operation of this simulated network. The min-window shows student's configuration on each LAN. (see Appendix 4.7 for more students' video clips screenshots)

Below are three comments made by students from their reflective reports. The first comment was made by the student who provided the video clip shown in Figure 4.5 followed by two comments from other students.

> "This entire network was built based on computational thinking, for example; the LANS with departments had to be broken into VLANS, and each VLAN was configured based on the necessary requirements given for the network" (*PGstud1 – reflective report PsTask2*)

> "Breaking down the whole network into LANs and designing the LANs with VLANs and the Inter-VLANs set is initially target and achieved it" (*PGstud4 – Reflective report PsTask2*)

> Another area where computational thinking really aided me was in the configuration of DHCP in Manchester and using the other branches as Relay agents to distribute IP addresses to the hosts. The configuration didn't work initially, but breaking the steps into bits, even writing it on paper and reapplying several configurations helped in resolving the whole situation (*PGstud3 – Reflective report PsTask2*)

Students believe that the process of breaking down network designs into small parts (subnets) helped them to fully design and fix problems of their simulated network infrastructures. It was evident that students applied the concepts of decomposition in isolating smaller solvable tasks which made their problems easier to solve, develop, evaluate separately concurring with the argument of scholars as discussed in Section 2.2.2 of the Literature Review about the application of the concept of decomposition. Some of the additional video-clips are shown in Appendix 4.7 demonstrating students' process of breaking down segments of WAN into LAN – demonstrating their application of the concepts of decomposition via simulation software.

All the six postgraduate students, who participated in these problem-solving tasks commented via their reflective reports and video clips that the strategy of breaking the complexity of the problem into solvable tasks was their main key concept applied in solving and building their simulated enterprise network infrastructure. Looking at the profile of these students (see Section 3.7 of the Methodology chapter), data revealed that some students progressed from practical-oriented institutions, such as polytechnics and colleges while others from industries

where solving technical problems are handled in scaffolding fashion (Blikstein, 2018). Therefore, it was not surprising that students solved their problems by means of breaking them into smaller partitions.

Based on the four of six (67%) students' overall reflective report who commented about the concepts of decomposition as indicated in Table 4.10, they emphasised that they were able to decompose complex problems easily to the level that they were able to solve them successfully concurring with Bocconi et al. (2016). They commented that the smaller subnets which they had broken down from both PsTask1 and PsTask2 via simulation software made it easier to visualise the entire simulated network problem concurring with the study of (Ristov et al., 2015; Zhang et al., 2012) arguing that simulation software facilitates in the visualisation of difficult networking concepts. Students outlined that the first process they did when solving problems was to decompose the problem on simulation software. One student commented that he first of all decomposed on the piece of paper before building and testing his design via simulation software. Other students commented that they were only able to realise that some networks were advertised differently after they had decomposed the network into small solvable tasks via simulation software. This gives further evidence on how decomposing an abstract via simulation software can help student visualise problems which could otherwise be overlooked. These findings remained consistent with students' comments in Section 4.3.2.2 in answering the second research question. Students, further said that after decomposing their network designs, it was easy and obvious to identify areas of vulnerability, loopholes, bottleneck and threats. The application of the concept of decomposition via simulation software facilitated students' thought processes to emerge ideas on how to overcome security vulnerabilities. They configured, tested, and corrected all areas of their network design to come up with their concrete functional simulated enterprise network infrastructures. Clearly their reflective reports and evidences via the video clips shown in Figure 4.5 and Appendices 4.7

and 4.8 demonstrate how students designed each LAN (i.e. into smaller manageable networks) before connecting to other LAN to form a WAN (simulated enterprise network infrastructure). Additionally, it was evident that students were able to troubleshoot and traceback the source of problems (routing outputs) easily to form a simulated enterprise network infrastructure shown in the Figure 4.5. Students' body language, voice and excitement shown when demonstrating their simulated enterprise network designs clearly indicated how they enjoyed solving problems as they applied the concepts of decomposition via simulation software. They strongly believed that the concepts of decomposition helped in solving their network designs and facilitated the process of troubleshooting problems. Students' reflective reports besides their video clips showed how they applied the concepts of decomposition. For example, they commented that:

> The process of computational thinking was extremely helpful in the area of decomposition. Simplifying the problem by breaking the general problem into little tasks, completing each of them, and eventually the whole process culminates into the general solution of the entire process, [...]. This made me realize that problems are easily more handled when they are tackled by decomposing the entire situation and solving it gradually (*PGstud4 – overall reflective report*)

> Dividing the problem into smaller but related parts and analyse each part separately have been really a useful strategy for me to simplify the problems as it gave me some type of control and confident over the given tasks. Moreover, organising the divided parts and identifying the priority of each could help me understand and find the most proper solutions (*PGstud5 – overall reflective report*)

Cross-examining this observation with the comments made by lecturers via LoS, and one-to-one interviews, results showed that lecturers expect students to handle complex problems in such a manner that are broken down into their constituent solvable parts. For example, these are some of the comments which lecturers made:

> Students must be made to realise that examining a large, complex network is only possible by splitting the problem up. […] It should become apparent to the students that the larger scenario must be split into smaller pieces to be able to understand the whole (*LoS*)

> in networking, often students have a vague knowledge of an area, they are not experts, they don't have the theory behind, so they could not solve a complex network problem by themselves, but they could split the problem into smaller chunks that are manageable (*lec2 – one-to-one interview*)

Additionally, reading through the profile of lecturers (see Table 3.5), it showed that most of them have networking industrial experiences in designing, troubleshooting and managing network infrastructures. These lecturers are bringing their industrial expertise and experiences into their teaching that help students to approach problem solving in scaffolding fashion.

Based on students' reflective reports and their demonstrations via video clips, results have shown that simulation software facilitated students' ability to break down complex network problems to solvable level which led to the solution of their concrete simulated enterprise network infrastructures. There were clear evidences showing students' ability to apply the concepts of decomposition via simulation software. Unlike when they were applying the concepts of abstraction, students were very conscious and that they made deliberate strategy to break down problems into their respective subnets to solve them successfully.

4.4.2.3 Demonstration of generalisation

After abstracting their complex networks and decomposing problems into solvable tasks on simulation software as discussed in Sections 4.4.2.1 and 4.4.2.2 respectively, students were supposed to solve the problems which were deliberately embedded in all the three problem-solving tasks. The initial stages in which students applied the concepts of abstraction and decomposition, helped them to establish the simulated enterprise network infrastructure to a functional state. However, they needed to solve all the inherited and deliberate problems which were embedded within their problem-solving tasks. Therefore, using the concept of generalisation students were required to demonstrate their strategies in identifying patterns, similarities and commonalities based on their prior knowledge and experience to solve their

network problems. Reflecting on the findings from the first research question in an attempt to investigate their understanding of CT, it was noticed that students and lecturers were not clear about the concept of generalisation when they were asked about the *what* than the *how* of CT. Furthermore, there were evidences from the findings of the second research question showing *how* the concepts of generalisation could be applied when designing computer networks on simulation software. These findings consistently showed that these students were able to explain *how* they understand the concepts of CT by *doing*.

On other hand, based on the third research question focusing on problem-solving tasks, the findings were slightly different. Students demonstrated more concepts of generalisation than those of decomposition and abstraction. For instance, there were 41 coding occurrences for generalisation, 29 coding occurrences for decomposition while 21 coding occurrences for abstraction as shown on Table 4.9. Examining closely on the sources of coding occurrences, it was found out that predominant coding occurrences were sourced from PsTask1 than PsTask2 and PsTask3 (see Table 4.10). For instances, all six students (100%) commented about the concepts of generalisation when reflecting on PsTask1. However, only two of six students (33%) commented about the concept of generalisation when reflecting on PsTask2 and PsTask3 respectively. These results could be down to the nature of problem-solving task given to these students. For instance, PsTask1 was based on a reverse engineering problem-solving task which required students to trace the routes of subnets and identify their similarities and differences to build the final simulated enterprise network infrastructure. Additionally, the skill of identifying similarities or commonalities and differences helped students in troubleshooting embedded and inherited problems leading to solving them. Analysing the 41 codes shown in Table 4.9, it was found that 27 of 41 codes (66%) from all the problem-solving tasks focused on students' ability to identify patterns, similarities and connections on problem-solving task.

PsTask2, required students to build an enterprise network infrastructure from scratch, while PsTask3 required students to implement security features on a different network infrastructure. On both second and third problem-solving tasks students had no basis for observing patterns or connections other than using their prior knowledge and experiences to construct their simulated network enterprise infrastructure. Analysing students' reflective reports from PsTask2 and PsTask3, they emphasised more on prior knowledge and experiences in reference to the concept of generalisation. Therefore, the nature of the PsTask1 may have influenced these results. Additionally, based on the reflective reports, students explained on how they identified patterns of similarities and connections in solving problems. Subnets which were connected to one router were identified by the other router via routing protocols and that students were able to relate the networks with their associated routing protocol after working on PsTask1. For example, these were some of the comments which students made:

> From the routing output for the router R1: the networks 10.10.10.8, 10.10.10.4, 172.16.1.0, 172.16.3.0 and 192.168.1.0 are advertised as OSPF routing hence these networks are configured with OSPF routing and 10.10.10.4 is also advertised as EIGRP hence that network is configured with EIGRP routing protocol. Similarly routing process for all the networks are found out and configured with particular routing protocol (*PGstud4 – reflective report PsTask1*)

> "Firstly, I had to identify the connections and network addresses in which the main network infrastructure was built-in" (*PGstud2 – reflective report PsTask1*)

These results indicate that students were able to apply their prior knowledge and experiences in identify the key components provided in the problem-solving tasks which meant that they were able to apply the concept of generalisation as argued by many scholars (e.g., Barr & Stephenson, 2011; Grover & Pea, 2013). Furthermore, based on students' overall reflective report which covered all the three problem-solving tasks, apart from the concept of decomposition, their comments focused on how they identified patterns, connections and similarities to the given problems. These results were similar when observed on their video

clips too. Furthermore, in describing how they resolved some problems in PsTask2 this is how students commented via their overall reflective reports:

> 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 (*PGstud6 – overall reflective report*)

> LAN & WAN are bit similar to each other apart from their size, but the logic behind the WAN is LAN. When I was in WAN module I find that if I get expertise in LAN it is very easy to crack the WAN. Now I can say confidently the process that going to be processed in the router after configuration. And what extra features are needed for better network (*PGstud5 – overall reflective report*)

Students' reflective reports show that they were able to identify the patterns of similarities and commonalities from their previous knowledge and experiences in designing LAN, WAN and security problem-solving tasks. These findings clearly show that students applied the concepts of generalisation when designing their networks via simulation software. These results are consistent with scholars as discussed in Section 2.2.3 arguing that the concept of generalisation is the technique of identifying patterns of similarities, commonalities and connection in problem-solving.

During one-to-one interviews with undergraduate students, they emphasised that prior knowledge to a problem solving is as equally important as solving a problem itself. This became evident when postgraduate students were solving problems on the simulation software. Students used their prior knowledge and experiences in solving some of the problems. For example, via reflective reports students commented that they used lecture notes, research, general knowledge of networking and prior experiences in solving problems. The theories of computer networks which were provided during their lectures gave them essential background knowledge to apply the concepts learnt by *doing* in the lab. Students acknowledged that the

theories taught in lectures, underpinning their practical tasks in the labs, are essential. Here are some of their comments:

> through the lectures notes, I finally understood the importance of each metrics mentioned on the OSPF - EIGRP redistribution we needed bandwidth (10000) for fast Ethernet 0/0, delay(255), load(255), EIGRP As-number and the MTU(1500) to match each and every used links for the network that is going to forwards the data through redistribution especially on R1 (*PGstud2 – reflective report PsTask2*)

> For configuring basic device security, I used my previous knowledge to implement the basic security commands on the devices. However, for the new or complicated requirements (in my case configuring ASA firewall, zone-based firewall or VPN) I planned to search the similar cases to know more about the nature of the required task or how to do it (*PGstud6 – reflective report PsTask3*)

> "I have researched a lot and came with a solution that in order to communicate with each other I have to configure them with redistribution" (*PGstud5 – reflective report PsTask1*)

However, observing the video clips, similar to their demonstration of abstraction discussed in Section 4.4.2.1, it appeared that students were not conscious that they were applying the concepts of generalisation in troubleshooting and fine-tuning their simulated network designs. It was only apparent when students were explaining their strategies via the video clips and reflective reports, in identifying and solving problems, that it became obvious that they applied the concepts of generalisation. Some students used trial-and-error approach. For example, this is the comment one of the students made in his reflective report concurring with the comment made by one of the lecturers via LoS that students solve problems by trial and error if they do not have structured approach to thinking:

> 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] (*PGstud3 – reflective report PsTask1*)

Unlike what students commented in Section 4.3.5 about the challenges of simulation software, students' concerns were the difficult on solving the problems given – either because the students did not have enough background knowledge or experiences with networks or they

simply did not do an extensive research to help in solving the problem. At postgraduate level students are encouraged to do some extensive research in the lab to broaden up their knowledge and skills (see Section 3.5 of the Methodology chapter). For instance, the comments made by the student above (PGstud3) shows his lack of technical knowledge to configure the network and not necessarily due to the challenges on simulation software. Therefore, results seem to suggest that students had no problems in using simulation software to design and troubleshoot their network designs. However, when students do not have a structured approach to problem-solving the tendency is they use trial-and-error approach as this student (PGstud3) commented. Again, this concurs with the remarks made by some of the lecturers that neither themselves (lecturers) nor students are conscious that they are using CT skills; their interest is to make sure that the problems are solved.

It became clear that students were solving problems not based on their understanding and ability to apply the concepts of CT, but rather as their normal day-to-day routine of troubleshooting, configuring and fine-tuning their network design on simulation software. Nonetheless, the results have shown that these students, at university level, are able to apply the concepts of CT by *doing* rather than by being taught in class. On another hand, the studies of many scholars (e.g. Barr & Stephenson, 2011; Grover & Pea, 2013; Lu & Fletcher, 2009; Wing, 2008) argue that students at primary and secondary levels need to be taught the concepts of CT first to help them develop their problem-solving skills. Certainly, the students in this study were not taught the concepts of CT at their primary and secondary schools because in the UK, CT was not introduced in the school curriculum until in 2014 (Bocconi et al., 2016) when these students had left school education. However, despite not being taught these students were able to apply the concepts of CT though on some concepts, such as abstraction and generalisation were applied subconsciously.

These results appear to suggest that even if Computer Networks students are not taught the concepts of CT on building and troubleshooting network designs via simulation software, they are able to apply them in their practical problem-solving tasks. Additionally, results have shown that simulation software facilitates students' ability to apply the concepts of CT. The results could be because these students largely learn computer network designs by means of practical tasks. Computer networks are abstract by nature and the only best way to understand them is by using practical tasks (Dobrilovic & Odadzic, 2006; Ruiz-Martinez et al., 2013) which is the central delivery style of this course. The results may have been different if the focus of the course is based on only theories of designing computer networks. Additionally, it would be interesting to investigate if there will be any difference if students are intentionally taught the concepts of CT before applying them.

4.4.2.4 Problem-solving involving abstraction, decomposition and generalisation

Reading through students' reflective reports, observing their video clips, and solutions to the problem-solving tasks, there were clear evidences showing that students applied different techniques and concepts to problem solving. There were 45 coding occurrences from 17 sources in reference to problem solving which covered the areas of abstraction, decomposition and generalisation as shown in Table 4.9. For example, students demonstrated the concepts of abstraction, decomposition and generalisation in problem solving concurring with the findings of the first research question (see Section 4.2.4.2 and illustration in Figure 4.2). Additionally, students explained and demonstrated via video clips how they solved the problems and how they handled encountered challenges based on their experiences in problem-solving tasks. Students explained and demonstrated how simulation software helped in visualising the requirements of the complex network design to understand and identify important details to focus on when building simulated enterprise network infrastructures. Clearly this showed the

concept of abstraction as discussed in Section 2.2.1 of the Literature Review chapter – though students were not conscious of. Furthermore, students explained and demonstrated through video clips how simulation software helped them in breaking down their simulated network design to LAN, VLANS and WAN to the level that they could solve, develop and evaluate separately. This was a clear demonstration of the concept of decomposition as discussed in Section 2.2.2 of the Literature Review and that students, in this study, were conscious of. Students also reported and demonstrated via video clips how they were able to solve problems of their simulated network designs by identifying patterns, similarities, connections based on their prior knowledge and experiences. Again, that was students' evidence of applying the concepts of generalisation as discussed in Section 2.2.3 of the Literature Review chapter though students were not conscious of. Additionally, the entire process of problem-solving using the concept of CT on simulation software helped students to understand some networking theories which would not have been possible otherwise. For instance, students believe that they now understand the concept of project life cycle covering the concepts of design and implementation of LAN/WAN, application and mitigation of security threats leading to a successful project. This is an example of the comments made by one of the students via his overall reflective report:

> Developing my thoughts by understanding the concepts of computational thinking has helped me improve the way to tackle any computing problem, especially in understanding the concepts of a project lifecycle as in designing, and implementing LAN/WAN topology, also applying security policies to any enterprise network infrastructure, mitigating threats, finding compatible equipment, introducing new ideas to overcome limitations, to finally achieve a successful project (*PGstud2 overall reflective report*)

In some instances, students were using trial-and-error approach to solve problem. This was evident via their video clips which demonstrated how and when they reached the dead-end of their reasoning in the process of problem-solving. They started using trial-and-error approach to get to the solutions of their problems further demonstrating that naturally students when they

do not have a structured approach to thinking they resort to trial-and-error approach. Moreover, one of the lecturers during one-to-one interviews commented that he does not mind the problem-solving technique approach students take as long as they resolve the problem(s) given which cover all their learning outcomes.

Overall students demonstrated that simulation software was effective in building up their confidence and facilitating their ability to solve problems. This inference concur with the findings of the second research questions in which both students and lecturers believe that simulation software is simple, easy and effective in designing and troubleshooting simulated network leading to the application of their CT skills. Interestingly from students' reflective reports, none of students' recommendations pointed out about how simulation software could be used better to improve their challenging experiences they may have encountered when designing and troubleshooting networks. This might imply that the limitations of simulation pointed out in Section 4.3.5 were not significant enough to impede their learning experiences in problem solving. These findings show that students were satisfied to use simulation software in designing and troubleshooting their simulated network design. Clearly using problem-solving tasks via simulation software students were able to apply the concepts of CT investigated in this study though students themselves were not conscious when applying the concepts of abstraction and generalisation.

4.4.3 Summary of findings in relation to the third research question

Throughout the results shown in this study in answering all the three research questions, students have been consistent and clearer on the concepts of decomposition than the other two concepts of CT investigated. In the first research question, they were clear in their definition of CT based on the concept of decomposition (see Section 4.2.2.2.1); in the second research question they were clear on how simulation software facilitate the application of the concept

of decomposition (see Section 4.3.3.2) and finally in answering the final research question, students clearly demonstrated via video clips and reflective reports, how simulation software facilitated the application of decomposition (see Figure 4.5). When solving all the three problem tasks, students were applying the concepts of decomposition consciously by breaking down problems into their respective subnets and solved them separately before joining them together (see Appendix 4.8). Table 4.10 shows that four of six students (67%) commented on PsTask2 and PsTask3 respectively about how they applied the concepts of decomposition. There were two of the six students (33%) who commented on how they applied the concepts of abstraction and decomposition respectively on PsTask1. This could be because this was their very first problem-solving task they were working on and therefore all students had not yet developed the skills to apply these concepts. However, on PsTask3, all the students (100%) commented on how they applied the concepts of generalisation. This could be because of the reasons already discussed in Section 4.4.2.3 that the nature of the PsTask1 required students to apply their skills in identifying similarities and differences of subnets, IP addressing and other problems by applying their prior knowledge of networking to build a simulated enterprise network infrastructure.

Furthermore, on the demonstration of the concept of decomposition, students commented that the strategies of breaking problems into smaller tasks via simulation software made it clear to visualise problems at all points they needed to solve and build up their simulated enterprise network infrastructure. All students constantly commented on the ability to break complex problem to solvable level as their key to solving problems. It therefore came out clear that students were conscious of applying the concept of decomposition in solving problems.

On another hand, although students were able to apply the concepts of abstraction and generalisation, they were not conscious of these two concepts. For instance, it only became clear via their video clips that all students had applied the concepts of abstraction and

generalisation upon interpreting the strategies they adopted in problem solving. It did not come seemingly natural or clear from students via their own reflective reports or focus group interviews that they were solving problems based on the concepts of abstraction and generalisation. Even observing via their video clips, students did not come clear that they applied the concepts of abstraction and generalisation in solving problems. It was when they started explaining about the strategies which they adopted in problem solving that it showed that they were applying the concepts of abstraction and generalisation in solving problems. They often used trial and error approach and that they did not have a clear structured approach unlike when they were applying the concepts of decomposition. For examples, they were only two of six students (33%) who commented about how they applied the concepts of abstraction on PsTask1 and three of six students (50%) for PsTask2 and PsTask3 respectively. Similarly, there were only two of six students (33%) on PsTask2 and PsTask3 respectively who commented about how they applied the concept of generalisation (see Table 4.10 and Appendix 4.3).

The findings from the third research question have tallied with findings from the second research question that simulation software provides an effective platform for students to apply the concepts of CT. Students were able to apply the concepts of abstraction, decomposition and generalisation which were demonstrated via video clips involving problem-solving tasks on simulation software. Despite the challenges of simulation software which students and lecturers commented about as discussed in Section 4.3.5, none of the students reported about the problems the faced nor explained if simulation software prevented them from applying the concepts of CT. Therefore, these results suggest that the challenges of simulation software discussed in Section 4.3.5 do not prevent students from applying the concepts of CT.

4.4.4 Overall summary of the findings in this study

In answering the first research question, the focus was to find out students' and lecturers' understanding of CT in relation to the concepts of abstraction, decomposition and generalisation to remain consistent with literature discussed in Section 2.2. Therefore, results have shown that both students' and lecturers' main concept of CT is the ability to breakdown problems into manageable tasks at the level that students can solve and evaluate separately (*i.e. decomposition*). In answering the first research question students and lecturers dominantly referred to the concept of decomposition as they were explaining their understanding of CT. Furthermore, in answering the second research question, students and lecturers emphasised that simulation software facilitates students' skills in breaking down problems to the level that they can solve complex problems. Following up with the findings of the second research question, students were able to apply and demonstrate via video clips and reflective reports the concepts of abstraction, decomposition and generalisation in answering the third research question. Consciously, when designing and troubleshooting simulated network systems, students made intentional strategy to break down their problem-solving tasks via simulation software to the level that they solved problems successfully. Reading through their reflective reports, students constantly used words, such as "*decompose problems*" or "*break down problems*" in solving complex problems.

However, from the findings of the first research question when students and lecturers were asked to explain what their understanding of CT were, students were not clear on their understanding of CT in refence to the concepts of abstraction and generalisation. It was only evident when students were asked non-explicit questions, such as *how* CT may help in designing and troubleshooting their network systems that ideas on the concepts of abstraction and generalisation began to emerge. Furthermore, in response to the third research question, it became evident that students were able to apply the concepts of abstraction and generalisation

when they began to design their networks on simulation software. Therefore, the results indicated that unless students, in this study, did a practical demonstration via problem-solving tasks, there were no evidence to suggest that they understood the concepts of abstraction and generalisation. It only became apparent that students were applying the concepts of abstraction and generalisation upon interpreting the strategies they adopted in solving problems. However, this was different when they were applying the concepts of decomposition as previously discussed.

Students in this study, have demonstrated their practical skills in applying the concepts of CT skills though they were not conscious of the concepts of abstraction and generalisation. There were some occasions that students used trial and error in solving some of their problems. They did not have a structured approach to thinking through their problem-solving tasks. Their main interests were to get to the solutions of their problems. Therefore, after breaking down problems to the level that they could solve, they used any means to get the problem solved. This observation concurred with what some of the lecturers commented when asked the strategies they use when teaching and assessing students when designing networks. Lecturers explained that they are neither conscious nor it has ever been their focus to teach or measure students' computational thinking when designing computer networks. Their main interest is to find out if students can apply and demonstrate the concepts taught in class by designing functional network infrastructures. Lecturers do not seem to have a formal structured thinking approach that students can adopt in problem-solving. Additionally, all lecturers during one-to-one interviews commented that they are not conscious to teach and later on assess students on CT. Lecturers' focus is on the solutions that students produce and not on how they get into those solutions. They believe that every student has his or her own way of solving a problem.

However, from this study, both students and lecturers recommended the significance of teaching and assessing students on a structured thinking approach (*which in essence is CT*)

when students design and troubleshoot networks via simulation software. Additionally, results have shown that simulation software facilitates students' CT skills in designing and troubleshooting simulated network systems.

CHAPTER 5: CONCLUSION

5.1 Introduction

This chapter starts by reiterating the study's objective and adopted methodology, followed by a summary of its key findings. Three keys areas of original contribution to knowledge are then explained, followed by implications and recommendations. The study ends with an outline of some limitations and personal reflection of the study.

5.2 Objective of the study

The overarching objective of this study was based on the use of Computer Networking simulation software to facilitate the application of students' CT skills. Studies of many scholars (e.g., Dobrilovic & Odadzic, 2006; Hwang, et al., 2014; Ruiz-Martinez, et al., 2013) have shown that simulation software provides conducive platform for students to design, modify, redesign, test unlimited network systems without the restrictions of physical devices and/or geographical location.

Currently, scholars in Computer Sciences (e.g. Brennan & Resnick, 2012; Catlin & Woollard, 2014; Seiter & Foreman, 2013) have largely focused on how CT may be developed, taught, and assessed at primary and secondary school levels with an emphasis on programming-related subjects. For example, Brennan and Resnick (2012) developed their own CT framework on how to teach and assess elementary school students using Scratch, a programming platform that enables school students to create their own interactive stories, games and simulations to share with peers across the world. Other scholars (e.g., Berland & Lee, 2011; Kazimoglu et al., 2010; Yadav et al., 2011) have focused the studies of CT at university level, again with an emphasis on only programming-related subjects. For example, Berland and Lee (2011) used non-computational media called *Pandemic* (board games) as a way of engaging undergraduate

students in complex CT. Furthermore, many other studies (e.g., Barr & Stephenson, 2011; Grover & Pea, 2013; Lu & Fletcher, 2009; Wing, 2008) are advocating the development of CT at primary and secondary school levels with the hope that it will create significant foundation for the development of future computing skills (Seiter & Foreman, 2013).

Currently, none of the existing studies have considered how CT may be developed, taught and assessed for students studying for Computer Networks at university level; yet some studies (e.g. Wing 2006, 2011) argue that CT can be applied in all disciplines. However, the study of Computer Networks comprises designing and troubleshooting complex computing designs involving the knowledge and skills of abstraction, decomposition and generalisation which are the concepts of CT. In view of this research gap, this study sets out to investigate Computer Network students' and lecturers' understanding of CT; their perceptions of using simulation software and how may the use of simulation software facilitate the application of students' CT skills.

5.3 **The adopted methodology**

This study adopted the qualitative mixed-method model. The qualitative mixed-method approach fitted well because of the nature of the study which investigated students' and lecturers' understanding of CT; their perceptions and how students' CT could be applied by the use of simulation software. Students' online survey (SoS) and lecturers' online survey (LoS) were initially conducted followed up by one-to-one interviews with four students and three lecturers to primarily address the first and second research questions. Two focus groups consisting of seven undergraduate students and six postgraduate students were conducted separately. The focus group conducted for the undergraduate students explored their understanding of CT and experiences of using simulation software when designing computer networks in their day-to-day practical lab activities to address the first and second research

question. On the other hand, the focus group conducted for the postgraduate students was based on the three problem-solving tasks via Cisco Packet Tracer simulation software in addressing the third research question. Additionally, data were collected from postgraduate students' reflective reports based on each of the problem-solving tasks to triangulate the findings from their focus group and video demonstration in addressing the third research question. Using thematic data analysis (Bryman, 2015), data collected from SoS, LoS, one-to-one interviews, focus group interviews and reflective reports were initially analyzed by pen and paper to fully familiarize and understand it. During this stage, multiples of proliferation of codes were generated and refined into categories to form themes. On the second stage, data were re-coded using NVIVO software to revalidate the coding process and help in presenting data analyzed and results in graphical format.

5.4 **Key findings**

5.4.1 Key findings to the first research question

Students' and lecturers' main understanding of CT was based on problem solving by means of the concept of decomposition which involves breaking down complex computing problems to the level that students can solve.

When students and lecturers were asked to explain their understanding of CT, they went beyond the conceptualisation of the current study's predetermined themes (i.e. *abstraction, decomposition* and *generalisation*) to include *problem-solving approaches* and *algorithmic thinking approach*. Problem-solving approach became their dominant concept of CT which encapsulated their definition of abstraction, decomposition and generalisation. For example, they indicated that *abstraction* is a problem solving approach in which students are able to identify and represent significant components of a network design systems while hiding all unnecessary details in order to come up with concrete network infrastructure; *decomposition*

is a problem-solving approach by which students are able to break down problems to the level that they can work on; *generalisation* is a problem-solving approach in which students use their prior knowledge and experience to identify patterns, similarities and commonalties to solve network design problems.

However, students and lecturers dominantly emphasised on the concepts of decomposition in explaining their understanding of CT. Whichever way students and lecturers were explicitly or non-explicitly asked about their understanding of CT, their dominant word used was "*breaking down complex problem*". Students found it easier to define CT as a concept of breaking down complex problems because largely their learning style involve splitting big tasks into smaller one which they can solve. Additionally, it was noted that words such as '*breaking down problems', 'decompose problems'* were familiar and closely related to their daily approach to problem-solving techniques when designing and troubleshooting network problems either via simulation software or physical hardware devices. On another hand, it did not seemingly come clear, particularly students to understand and relate words such as abstraction and generalisation to their normal approaches to problem-solving skills until when they began to explain the strategies they thought CT would help in solving problems. Cross-examining lecturers' understanding of CT, they echoed that they expect students to break down problems into smaller bits to be able to solve them. Furthermore, examining lecturers' profile, it was noted that most of them have practical and industrial work experiences which implies that their teaching styles are practical-oriented approaches focusing on training students to solve practical and technical networking problems by means of breaking complex network designs into smaller solvable ones.

Furthermore, students and lecturers mentioned that *algorithmic thinking* is a problem-solving approach in which students use some "*if statement"* when working out programming-related

problems. However, there were few students and lecturers who mentioned about algorithmic thinking. This could be because their day-to-day practical tasks do not involve algorithms.

Overall, it was evident that students' understanding of CT was based on problem-solving concept which dominantly use the concepts of decomposition besides abstraction and generalisation. The findings from the first research question appear consistent with many scholars (e.g., Blikstein, 2018; Bocconi et al., 2016; Grover &Pea, 2013; Wing, 2011) particularly on the aspect that problem solving is involved in the application of the concepts of CT. However, the first research question has mainly shown that students and lecturers understand CT based on the concept of decomposition.

5.4.2 Key findings to the second research question

When students and lecturers were investigated on their perceptions of the use of simulation software to facilitate students' CT, majority of them indicated the simplistic and effective ways of abstracting, decomposing and generalising simulated network design systems when solving problems. Below is a synopsis explanation of their perceptions:

On the concept of abstraction, students and lecturers believe that students are able to abstract the complexity of network designs and understand problems which would otherwise be difficult to solve on physical hardware devices. For example, students and lecturers indicated that simulation software facilitates in the visual representation of complex network problems such that it becomes easier and simpler for students to have a wider insight to problems leading to multiple views of understanding them (Valanides & Angeli, 2009). Furthermore, during their focus group interviews, students mentioned that on simulation software they are able to visualise the movement of *packets* (data) from one end node to the other. This makes it easy for the students to conceptualise the operation of protocols in a network enforcing their understanding of network abstraction which is not possible on the physical devices.

Furthermore, students and lecturers reiterated that the visual representation of complex network problems stirs up students' imaginations leading to their innovation and creativity in designing and troubleshooting network systems concurring with many studies (e.g., Ruiz-Martinez et al., 2013; Wing, 2008; Zhang et al., 2012). However, similar to the first research question, students' and lecturers' perception of how simulation software facilitates the concepts of abstraction did not come out seemingly obvious, but it was made clear upon interpreting their own explanation of the use of simulation software. Otherwise, they were not conscious that their explanation about how simulation software facilitates the concepts of CT was about the concepts of abstraction. This could be because they were not explicitly taught the concepts of CT.

On the concept of decomposition, students and lecturers indicated that the use of simulation software makes it easier and simpler to decompose network design systems to the level that they can solve. For example, students mentioned that with the use of simulation software, they can build, break, modify, test parts of their simulated network designs much easier and simpler than using physical hardware devices. They believe that they can work on one component of a network design at a time. Basically, students indicated that they can solve and evaluate one part of a network design problem before joining them together. For instance, they mentioned that if they are working on designing a wide area network (WAN), they are able to break and work on each local area network (LAN) first before linking-up all LANs together to form a WAN. Similar to troubleshooting network design problems, students and lecturers indicated that students find it easier and simpler to isolate and solve problems via simulation software than on physical hardware devices. They commented that in contrast they cannot visually decompose network design systems on physical hardware devices. Therefore, students and lecturers believe that simulation software provides more flexibility than real physical device in designing complex networks which can help in applying their CT skills. Consistently with the

first research question, students and lecturers were very clear on how simulation software facilitates the concepts of decomposition. There were no extra prompts for the students and lecturers to explain about the application of the concepts of decomposition via simulation software. It appeared that the application of the concepts of decomposition via simulation software was obvious and natural to them. These findings indicated students' and lecturers' perceptions of how simulation software may facilitate the concept of decomposition.

On the concept of generalisation, students and lecturers indicated that the visual representation of network systems via simulation software, provides a better view for students to identify patterns of similarities, commonalities, differences in their network design which makes it easy for them to solve problems. Additionally, they indicated that students are able to use their prior knowledge and experiences in designing and solving problems thereby engaging their ability to apply the concept of generalisation. Similar to findings based on the first research question, students and lecturers were not conscious that when they were explaining how simulation software facilitates the application of solving problems by identifying similarities, differences and prior knowledge were the concepts of generalisation. They simply explained the strategies students use in troubleshooting network design problems via simulation software. Nonetheless, findings suggested how simulation software facilitates the application of generalisation in troubleshooting network design systems.

Additionally, students indicated that the flexibility of using simulation software enable them to continue designing and troubleshooting their network systems whenever and from wherever they wish without the restrictions of physical hardware devices and geographical locations thereby enhancing the continual experiences in problem-solving skills leading to CT. Furthermore, students and lecturers believe that the use of simulation software helps students to build up their confidence in handling complex networks on the physical network infrastructure.

Findings from SoS and LoS, indicated that students and lecturers believe that simulation software provides a good platform to troubleshoot network design thereby facilitates students' CT skills. Nonetheless, there were few students and lecturers who indicated the challenges of using simulation software, such as student missing out the industrial experiences of working with physical network devices and that some commands cannot be used on simulation software. However, none of these limitations indicated that students can not apply the concepts of CT via simulation software.

Overall, in answering to the second research question students' and lecturers' perceptions of problem solving via simulation software show the effectiveness of facilitating the application of the concepts of CT, such as abstraction, decomposition, generalisation. These findings resonate with the literature (e.g., Angeli et al., 2016; Csizmadia et al., 2015; Wannous & Nakano, 2010; Wing, 2008), demonstrating the effectiveness of facilitating the concept of abstraction, decomposition, and generalisation via simulation software.

5.4.3 Key findings to the third research question

Students were given three network design problem-solving tasks to solve via simulation software to follow-up from the second research question in investigating how simulation software may be used to facilitate the application of students CT skills. The results have shown that students were able to apply the concepts of abstraction, decomposition and generalisation via simulation software concurring with the findings obtained from the second research question.

Although students were able to apply the concepts of abstraction and generalisation via simulation software, it did not come obvious to students neither were they conscious that the strategies they adopted in solving problems were based on the concepts of abstraction and generalisation. In some instances (e.g. when troubleshooting their network designs), they used

trial and error approach to solve their problems. They did not have a structured thinking approach to solving problems. It only came out clear that students had applied the concepts of abstraction and generalisation upon viewing their video clips and interpreting their reflecting reports on how they solved the problems.

However, as observed from the first and second research questions, students clearly demonstrated how they applied the concepts of decomposition in designing network systems via simulation software. Students consciously adopted the strategies to decompose complex problems into smaller solvable tasks. It became obvious via the use of the language in their reflective reports, facial expressing and video demonstration that all students were familiar with the concepts of decomposition. They used words such as "*breaking down problems"*, "*decompose problems",* via their reflective report and video demonstration. Their video clips clearly demonstrated how they applied the concepts of decomposition in designing their simulated enterprise network infrastructure. These results suggested that students were very familiar with the concepts of decomposition – and this could be largely because their learning style is in such a way that they spent more hours focusing on developing practical skills in designing and troubleshooting network systems in the lab by solving one part of a problem at a time.

Although it was only obvious when they were applying the concepts of decomposition, this study has shown that students are able to apply the concepts of CT via simulation software. The simplistic and flexible usage of simulation software led students to easily and quickly abstract complex network to the level they could understand; decomposed problems to a solvable and manageable level; identified trends, patterns, similarities and commonalities in problem-solving. Students could easily apply their prior knowledge and experiences in solving problems.

The findings for the third research question resonated with results obtained from the first and second research questions. Finally, both students and lecturers in this study, acknowledged and recommended the significance of teaching and assessing CT when designing and troubleshooting networks on simulation software. They believe that it will help students to have a thinking structured approach to problem solving which will enhance not only in problem solving but enable them to develop and produce network systems.

5.5 **Original contribution to knowledge**

There are three main significant original contributions to knowledge as explained below. These contributions are not necessarily in any significant order:

Firstly, the study has revealed that both lecturers and students of Computer Networks course were not aware of the term, "*computational thinking*", in their teaching and learning experiences respectively. For instance, lecturers consistently commented that they were not conscious when teaching and assessing students about the skills of CT. They believe that students solve problems by means of trial and error. They also believe that each student has his or her own best way of solving problems, therefore lecturers' interests are to only check that students have produced the desired goal that satisfies the learning outcome. Lecturers' concerns were not the thought processes that students go through in solving problems but rather the solutions they give. Additionally, it was evident when all the three lecturers, who participated on the one-to-one interviews, revealed that they did not know the term "*computational thinking*". They had to search on Google before the interviews. Therefore, if lecturers did not know the term "*computational thinking*" it was unlikely for them to consciously teach and assess their students. Additionally, students during focus group interviews confirmed that they are not taught the concepts of CT. The focus on this course is to develop students' practical skills to design and troubleshoot network systems leading to employment. Therefore, the study

reveals the need for the lecturers in Computer Networks course to be adept in the concepts of CT to consciously teach and assess students. This will also help students to be conscious of the application of CT skills when designing and troubleshooting network systems. Although students managed to apply all the three concepts of CT investigated in this study, it was only the concept of decomposition that was clearly evident and consistent throughout the three research questions. It was because students were mainly familiar with the concepts of decomposition which they could easily relate to when designing and troubleshooting network systems. Additionally, in this study students simply answered all the key areas on the problem-solving tasks, but they did not demonstrate their innovation or creativity into their simulated network infrastructure considering that they were postgraduate students who should have been more imaginative or thoughtful to create and develop systems. A structured-thinking approach, such as CT to problem-solving will be helpful not only in developing their problem-solving skills (e.g Wing, 2011) but also to become producers of network systems in the 21st century as argued by many scholars (e.g. Angeli et al 2016; Bocconi et al., 2016).

Secondly, extending on the studies of many scholars (e.g., Barr & Stephenson, 2011; Berland & Lee, 2011; Seiter & Foreman, 2013) which have focused on the development of CT skills in schools and universities in the areas of only Programming and Games courses, this study has provided evidenced-based data on how CT can be applied and developed for students studying on Computer Networks course via simulation software. In the discipline of Computer Sciences studies have mainly shown how CT is developed and applied in Programming and Games courses and yet other studies (e.g., Wing, 2006, 2011) argue that CT can be taught and applied across any other discipline, such as Sports, Business, Law, Biology, Sciences, and Engineering, just to mention a few examples. Currently there are no sufficient studies showing how CT can be applied on students studying for Computer Networks course. However, students studying for Computer Networks in this study have shown how the concepts of abstraction,

decomposition and generalisation can be achieved via simulation software while designing and troubleshooting network systems. Therefore, it would be interesting that further studies investigate how other core concepts of CT, such as algorithms, debugging, automation among others as shown in Table 2.1 may be achieved for students studying for Computer Networks course.

These study skills (i.e. the concepts of CT) need to be re-enforced not only in practical-oriented subjects but across other subjects too which may not be practical oriented. These study skills are essential in developing students thinking approach to problem solving which will foster their imaginations and hopefully trigger off their creativity and innovation to design and produce systems in the 21st century as argued by many scholars discussed in Section 1.1 of the Introduction chapter.

Thirdly, concurring with other studies (e.g., Dobrilovic & Odadzic, 2006; Hwang et al., 2014; Ruiz-Martinez et al., 2013), the study has revealed that the use of simulation software is conducive in facilitating the application of CT skills. In this study, students found the use of simulation software simple, easy and effective in their problem-solving tasks which enabled them to abstract computer network systems to the level that they were able to understand the complexity of network design systems. Students were able to decompose problems to the level that they could solve and manage them. Finally, the visual representation of problems via simulation software enables students to identify patterns of similarities, commonalities, differences besides the use of their prior knowledge in designing and troubleshooting network systems. Additionally, simulation software provided an environment which fostered students' confidence, creativity and imagination, which then facilitated the application of CT. The study has therefore shown that simulation software provides a conducive platform for students to develop and apply their CT skills.

5.6 **Implications and recommendations**

Scholars are advocating the significant requirement of developing CT at primary and secondary levels with little evidence on how these skills have been developed for students studying for Computer Networks at higher education level. As alluded to from previous sections in this thesis, scholars have mainly advocated the application of CT on programming-related subjects. However, this study has shown that although students are able to apply the concepts of abstraction, decomposition and generalisation, it was only the concept of decomposition that was obvious and familiar with them because they were able to relate to their daily practical problem-solving approaches. The results from this study suggest that it is unlikely that these students would be able to apply the same concepts across other subjects. Computer Networks students need to develop computer network systems rather than mere repairing the existing network systems faults. Therefore, the developing and application of thinking-structured approach, such as computational thinking to problem-solving would make them stand out not only in developing their problem-solving skills but becoming producers of computer network systems (Bocconi et al., 2016).

Although throughout this study students were not taught the concepts of CT, their perceptions in explaining how simulation software facilitates the development of their problem-solving skills suggest that they are able to apply the concepts of CT. Furthermore, in answering the third research questions, results clearly demonstrated that students were able to apply the concepts of CT. Therefore, results suggest that although students at university level may not know the academic term "computational thinking", it does not necessarily imply they do not know how to apply its concepts. It would therefore be recommendable that some of these terms introduced in education, such as "computational thinking" are carefully considered since they may simply puzzle students yet in their simplistic term, such as breaking down of problems, use of pattern identification to solving problems and using prior knowledge and experiences,

are familiar to students and applicable in their normal approaches to problem-solving tasks. For example, one student during focus group commented that: "*Computational thinking is kind of like an academic buzz word and not real while in deploying enterprise architects*" (*UGstud1 – UGFG).* This may imply that if CT is known to students in its simplistic form it would not sound as an academic buzz word. Additionally, students would easily relate to its concepts. However, from this study, results have indicated that the concepts of CT skills need to be re-enforced in their teaching and learning so that students not only improve in their problem-solving skills but also develop and produce network systems in the 21st century.

Therefore, the curriculum for Computer Networks course at university level needs to integrate the concepts of CT in a simplified form where students and lecturers can understand the terms used. Lecturers, as agents of change, need to familiarise themselves with the concepts of CT via staff development training to apply purposefully the concepts in their professional practice. This will help to re-enforce these problem-solving skills in their teaching and learning styles particular as they develop a thinking-structured approach which can help students to be the producers of network systems than consumers. Empowering lecturers with the understanding of these concepts will mean that they will be able to transfer and transform students' skills in designing and developing network systems. Currently, this study has revealed that Computer Network students are not explicitly taught the concepts of CT.

The study has also shown that simulation software facilitates students to visually understand complex problems, thereby helping them to work-out solutions at their level of comprehension leading to the application of CT skills. Furthermore, the study has shown that students are able to build their confidence when designing networks via simulation software. It is therefore recommendable from this study that students studying on Computer Networks course are introduced to the simplest and familiar terms of the concepts of CT when designing and troubleshooting complex network systems via simulation software. This practice will also help

students master skills and build their confidence to handle complex network design on real physical devices.

Further work is needed to investigate whether, and how, other core concepts of CT such as the ones listed on Table 2.1 may be developed implicitly via the use of simulation software or other aspects of education in computer networking. The results from this study were conducted on a smaller scale and that the focus was initially on the three core concepts of CT (i.e. abstraction, decomposition and generalisation).

5.7 **Limitations**

The major limitation of this study was the breadth and content of the study since it only focused on students studying for Computer Networks course, particularly those enrolled on the postgraduate level. Initially, for a broader view of investigating students' and lecturers' understanding of CT, surveys were conducted on all undergraduate and postgraduate students studying for Computer Networks course. Although there was a bigger number of students (69) who participated on surveys, there were only six postgraduate students who participated fully in the entire study. The small sample size of students who fully participated to the end of this study was because I did not have direct contact with undergraduate students other than those I was supervising for their projects. I ruled out to depend on fellow lecturers who were teaching undergraduates students to collect data for me because I did not want to miss out students' body language, voice used, their enacted values besides the richness and authenticity of the original data (Bearman & Dawson, 2012; Muller & Damico, 2002). I had a small cohort of postgraduate students (six) who I was teaching therefore they became the main source of data collection.

Additionally, the original plan was to give problem-solving tasks to both undergraduate and postgraduate students to help in comparing how both cohort of students apply their CT skills on simulation software. However, it was not possible because of the same reason that I was not

teaching undergraduate students therefore I did not consider it feasible and legitimate to ask someone to conduct the experiment for me. Following these constraints, the focus of the study was predominately on the six postgraduate students who I was teaching.

The other limitation to this study could be the two survey questions in which students and lecturers were asked to identify their main concepts of CT (see Appendices 3.01 & 3.02) provided via SoS (questions 5 and 6) and LoS (questions 6 and 7) respectively. These questions might be biased to only the three core concepts of CT (abstraction, decomposition and generalisation). The rationale behind these questions was to remain consistent with the investigation of the study which focused on students' and lecturers' understanding of CT based on the three core concepts as explained in Section 2.2 and 4.2.2. However, reflecting on the findings of this study, these questions may not have provided enough choices for participants to give their own views about their understanding of CT. For example, the questions should have perhaps provided either more options of the core concepts of CT as provided in Table 2.1 or should have been open-ended questions to eliminate any form of bias. Otherwise, the questions seem to have been ring-fenced within the three core concepts of CT. Nonetheless, via the use of one-to-one and focus group interviews, participants had open-ended discussions which helped them to explain more ideas which may have been ring-fenced via surveys.

In view of the outlined limitations, the results in this study may not be generalisable, further studies exploring more students across different levels of their studies (i.e. undergraduate and postgraduates) and different sets of the core concept of CT as outlined in Table 2.1 are recommendable.

Nonetheless, there were a considerable number of students who participated on online survey (SoS). For example, there was a total of 69 students of which 58 were undergraduate students and 11 were postgraduate students. Furthermore, there were 14 lecturers who participated on

online survey (LoS); seven undergraduate students and six postgraduate students participated in the focus groups respectively.

5.8 **Reflections**

As a lecturer teaching on Computer Networks course, this study has given me a more evidenced-based foundation on how to improve my teaching practice particularly on how I can help my students to apply and develop their CT skills. Furthermore, the study has helped to fully understand my colleagues' approaches to teaching and assessment. During one-to-one interviews with my fellow lecturers there were many issues which became evident and obvious that are taken for granted yet are fundamental to students learning experiences. For instance, as alluded to in the previous sections of this study, the focus of the teaching style has been to produce students who are ready for employment and not necessarily to develop their thought process that the can develop and produce network systems. The problem-solving tasks which the students in this study were given were fictitious but were designed in such a manner that they were thought provocative and imaginative fitting in the requirements of the curriculum to cause students to design and develop their own network systems. However, students only designed simulated network systems but did not manage to design and produce their own systems because students were not explicitly taught the concepts of CT and that there was limited time for data collection. On the other hand, majority of lecturers only focus on the solutions which students produce without following up the process students go through in coming up with their solutions. The focus has been to let student work on their own problem in whatever way suitable for them. The results of this study have provided a basis for sharing good professional practices in teaching students on how to apply the concepts of CT skills via simulation software.

REFERENCES

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference? *Future of learning group publication*, *5*(3), 438-449.

Aho, A.V. (2012). Computation and computational thinking. *The Computer Journal*, *55*(7), 832-835.

Angeli, C., & Valanides, N. (2009). Epistemological and methodological issues for the conceptualisation, development, and assessment of ICT–TPCK: Advances in technological pedagogical content knowledge (TPCK). *Computers & Education*, *52*(1), 154-168.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology & Society*, *19*(3), 47–57.

Barbour, R. S. (2005). Making sense of focus groups. *Medical Education*, *39*(7), 742-750.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*, 48–54.

Bearman, M., & Dawson, P. (2013). Qualitative synthesis and systematic review in health professions education. *Medical Education*, *47*(3), 252-260.

Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning*, *1*(2), 65-81. doi: 10.4018/ijgbl.2011040105.

Blikstein, P. (2018). Pre-College Computer Science Education: A Survey of the Field. *Mountain View, CA: Google LLC*. Retrieved from https://goo.gl/gmS1Vm

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.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada (1-25).

Bryman, A. (2015). *Social research methods* (5th ed.). New York, NY: Oxford University Press.

Burke, L.A., & Williams, J.M. (2008). Developing young thinkers: An intervention aimed to enhance children's thinking skills. *Thinking Skills and Creativity, 3*(2), 104 – 124.

Catlin, D., & Woollard, J. (2014). *Educational robots and CT*. Paper presented at Teaching Robotics & Teaching with Robotics (TRTWR) - *Robotics in Education (RIE) 2014 Conference,* Padova, Italy.

Chance, P. (1986). *Thinking in the classroom: A survey of programs*. New York, NY: Teachers College Press.

Cochran, K. F., DeRuiter, J. A., & King, R. A. (1993). Pedagogical content knowing: An integrative model for teacher preparation. *Journal of Teacher Education, 44*(4), 263-272.

Cohen, L., Manion, L., & Morrision, K. (2009). *Research methods in education* (6th ed.). London, UK: Routledge.

Cole M. (2009) Using Wiki technology to support learners engagement: lessons from the trenches. Computers & Education Elsevier journal. (52), 141-146

Crang, M. & Cook, I. (2007). *Doing ethnographies*. Thousand Oaks, California: Sage.

Creswell, J.W. (2014). *Research design: qualitative, quantitative and mixed methods approaches* (4rd ed.). Thousand Oaks, CA: Sage.

Creswell, J. W. & Miller, D. L. (2000). Determining validity in qualitative inquiry. *Theory into Practice*, 39(3), 124-131.

Creswell, J. W., Shope, R., Plano Clark, V. L., & Green, D. O. (2006). How interpretive qualitative research extends mixed methods research. *Research in the Schools, 13*(1), 1-11.

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the research process*. London, UK: Sage.

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*. Retrieved from http://computingatschool.org.uk/computationalthinking

Curzon, P., Dorling, M., Ng, T., Selby, C., & Woollard, J. (2014). *Developing computational thinking in the classroom: a framework*. Swindon, GB: Computing At School.

Czerkawski, B.C. (2015). Computational thinking in virtual learning environments. In *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2015 (993-997).* Chesapeake, VA: Association for the Advancement of Computing in Education (AACE).

Czerkawski, B.C., & Lyman III, E.W. (2015). Exploring issues about computational thinking in higher education. *TechTrends*, 59(2), 57-65.

Decrop, A. (1999). Triangulation in qualitative tourism research. *Tourism Management*, *20*(1), 157-161.

Denzin, N.K., & Lincoln, Y.S. (2005). Introduction: The discipline and practice of qualitative research. In N.K. Denzin & Y.S. Lincoln (Eds.), *The Sage handbook of qualitative research* (1-32). Thousand Oaks, CA: Sage.

Dobrilovic, D., & Odadžic, B. (2006). Virtualization technology as a tool for teaching computer networks. *Learning*, *133*(231), 712-716.

Elliot, G. (1996). Why is research invisible in Further Education? *British Educational Research Journal*. *22*(1), 101-111.

Empson, S., (2009). *CCNA portable command guide*, (2nd ed.) USA, Cisco press.

Ennis, R. H. (1989). Critical thinking and subject specificity. Clarification and needed research. *Educational researcher*, *18*(3), 4-10.

Ertmer, P. A., & Ottenbreit-Leftwich, A. T. (2010). Teacher technology change: How knowledge, confidence, beliefs, and culture intersect. *Journal of Research on Technology in Education*, *42*(3), 255-284.

Ertmer P.A., Ottenbreit-Lftwich A.T., Sadik O., Sendurur E., & Sendurur P. (2012). Teacher beliefs and technology integration practices: A critical relationship. *Computer & Education 59*(2), 423-435.

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 (1).

Ferdig, R. E., Mishra, P., & Zhao, Y. (2004). Component architectures and Web-based learning environments. *Journal of Interactive Learning Research, 15*(1), 75–90.

Fosnot, C. T. (2013). *Constructivism: Theory, perspectives, and practice*. Teachers College Press

Fowler, F.J. (2002). S*urvey research methods* (3rd ed.). Thousand Oaks, CA: Sage

Floyd, A., & Arthur, L. (2012). Researching from within: External and internal ethical engagement. *International Journal of Research & Method in Education, 35*(2), 171-180.

Fluck, A., Webb, M., Cox, M., Angeli, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for computer science in the school curriculum. *Educational Technology and Society, 19*(3), 38-46.

Fuertes, W., de Vergara, J.L.. & Meneses, F. (2009). Educational platform using virtualization technologies: Teaching-learning applications and research uses cases. *In Proc. II ACE Seminar: Knowledge Construction in Online Collaborative Communities*, 16.

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.

Galán, F., Fernández, D., Ruiz, J., Walid, O., & De Miguel, T. (2004). Use of virtualization tools in computer network laboratories. *In Proc. 5th International Conference on Information Technology Based Higher Education and Training*, 209-214.

Gilbert N (2001). *Researching social life* (2nd ed.). London, UK: Sage.

Gill, J. & Johnson, P. (2002). *Research methods for managers* (3rd ed.). London, UK: Sage.

Golafshani, N. (2003). Understanding reliability and validity in qualitative research. *The Qualitative Report, 8*(4), 597-606.

Goode, J., Chapman, G., & Margolis, J. (2012). Beyond curriculum: The Exploring computer science program. *ACM Inroads*, 3(2), 47-53.

Graneheim, U. H., & Lundman, B. (2004). Qualitative content analysis in nursing research: concepts, procedures and measures to achieve trustworthiness. *Nurse Education Today*, *24*(2), 105-112.

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

Guba, E.G. & Lincoln Y.S. (1994). Competing paradigms in qualitative research. In: N.K. Denzin & Y.S. Lincoln (Eds.). *Handbook of Qualitative Research* (105-117). Thousand Oaks, CA: Sage.

Halpern, D. F. (1998). Teaching critical thinking for transfer across domains. Dispositions, skills, structure training, and metacognitive monitoring. *American Psychologist*, 53, 449–455.

Halpern, D. F. (1996). *Thought and knowledge: An introduction to critical thinking* (4th ed.). New Jersey: Lawrence Erlbaum.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *SIGCSE Bulletin*, *41*(1), 183-7.

Harris A., Jones M., & Baba S. (2013). Distributed leadership and digital collaborative learning: A synergistic relationship? *British Journal of Education Technology*, *44*(6), 926-939.

Hemmendinger, D. (2010). A plea for modesty. *Acm Inroads*, *1*(2), 4-7.

Hitchcock, G., & Hughes, D. (1995). *Research and the teacher* (2nd ed.). London, UK: Routledge.

Howe, K. R. (2004). A critique of experimentalism. *Qualitative Inquiry*, *10*(1), 42-61.

Hughes, J. (2005). The role of teacher knowledge and learning experiences in forming technology-integrated pedagogy. *Journal of Technology and Teacher Education*, *13*(2), 277-302.

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.

Johnson, R. B., & Onwuegbuzie, A. J. (2004). Mixed methods research: A research paradigm whose time has come. *Educational Researcher, 33*(7), 14-26.

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.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2010). Developing a game model for computational thinking and learning traditional programming through game-play. *In J. Sanchez & K. Zhang (Eds.), Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*. 1378-1386.

Kennedy T.J.T., & Lingard L. A. (2006). Making sense of grounded theory in medical education. *Medical Education, 40,* 101-108.

Knowles, M.S., (1970). *The modern practice of adult education* (41). New York: New York Association Press.

Koehler, M. J., Mishra, P., & Cain, W. (2013). What is technological pedagogical content knowledge (TPACK)? *Journal of Education, 193*(3), 13-19.

Koehler, M. J., & Mishra, P. (2005). Teachers learning technology by design. *Journal of Computing in Teacher Education, 21*(3), 94–102.

Koehler, M. J., & Mishra, P. (2002). Art from randomness. How Inverso uses chance to create haiku [Electronic version]. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning, 4*(1).

Koehler, M. J., Mishra, P., & Yahya, K. (2007). Tracing the development of teacher knowledge in a design seminar: Integrating content, pedagogy and technology. *Computers & Education, 49*(3), 740-762.

Kuhn, D. (1999). A developmental model of critical thinking. *Educational Researcher, 28,* 16–25.

Kules, B. (2016) Computational thinking is critical thinking: Connecting to university discourse, goals, and learning outcomes. *Proceedings of the association for information science and technology*, *53*(1), 1-6.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, *2*(1), 32-37.

Leung, L. (2015). Validity, reliability, and generalizability in qualitative research. *Journal of family medicine and primary care*, *4*(3), 324.

Lewins, A., & Silver, C. (2009). Choosing a CAQDAS package. (6th ed.) CAQDAS Networking Project Working Paper. Retrieved from http://eprints.ncrm.ac.uk/791/1/2009ChoosingaCAQDASPackage.pdf

Lincoln, Y.S., & Guba, E.G., (1985). *Naturalistic Inquiry*. London, UK: Sage.

Liu, C., Cheng, Y., & Huang, C. (2011). The effect of simulation games on the learning of computational problem solving, *Computers & Education*. *57*(3), 1907- 1918.

Lu, J.J., & Fletcher, G.H. (2009). Thinking about computational thinking. In *ACM SIGCSE Bulletin 41*(1), 260-264.

Lund T (2012). Combining qualitative and quantitative approaches: Some arguments for mixed methods research. *Scandinavian Journal of Educational Research, 56*(2), 155-165.

Mack, L. (2010). The philosophical underpinnings of educational research. *Polyglossia*, *19*, 1-11.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference* (1-29). ACM.

Mason J. (2006). Mixing methods in a qualitatively driven way. *Qualitative Research*, *6*(1), 9-25.

Mathison, S. (1988). Why triangulate? *Educational Researcher*, *17*(2), 13-17.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record*, *108*(6), 1017.

McGuinness, C. (1993). Teaching thinking: new signs for theories of cognition. *Educational Psychology*, *13*(3-4), 305-316.

Müller, N., & Damico, J. S. (2002). A transcription toolkit: Theoretical and clinical considerations. *clinical linguistics & phonetics*, *16*(5), 299-316.

Musheer, A., Sotnikov, O., & Heydari, S. S. (2012). Multiuser Simulation-Based Virtual Environment for Teaching Computer Networking Concepts. *International Journal on Advances in Intelligent Systems*, *5*(1).

Myers, M. D. (1997). Qualitative research in information systems. *Management Information Systems Quarterly*, *21*(2), 241-242.

National Research Council. (2010). Committee for the workshops on computational thinking: *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Academy Press. doi:10.17226/12840.

Niess, M. L. (2005). Preparing teachers to teach science and mathematics with technology: Developing a technology pedagogical content knowledge. *Teaching and Teacher Education*, *21*(5), 509-523.

Onwuegbuzie, A. J., & Johnson, R. B. (2006). The validity issue in mixed research. *Research in the Schools*, *13*(1), 48-63.

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York, NY: BasicBooks.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, *36*, 1-11.

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

Patton, M. Q. (2002). *Qualitative evaluation and research methods* (3rd ed.). Thousand Oaks, CA: Sage.

Phillips, D. C., & Burbules, N. C. (2000). *Postpositivism and educational research*. New York, NY: Maryland.

Pole, K. (2007). Mixed method designs: A review of strategies for blending quantitative and qualitative methodologies. *Mid-Western Educational Researcher*, *20*(4), 35-38.

Polit, D.F., & Hungler, B.P., (1999). *Nursing research: Principles and methods* (6th ed.). New York, NY: Lippincott Williams & Wilkins.

Pulimood, S. M., Pearson, K., & Bates, D. C. (2016). A study on the impact of multidisciplinary collaboration on computational thinking. In *Proceedings of the 47th ACM technical symposium on computing science education* (30-35). ACM.

Repenning, A., Basawapatna, A. R., & Escherle, N. (2016). 'Computational thinking tools', *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–5.

Pulimood, S. M., Pearson, K., & Bates, D. C. (2016). A study on the impact of multidisciplinary collaboration on computational thinking. In *Proceedings of the 47th ACM technical symposium on computing science education* (30-35). ACM.

Ristov, S., Spasov, D., & Gusev, M. (2015, March). Successful integration of practical Cisco CCNA in the Computer Networks Design course. In *Global Engineering Education Conference (EDUCON), 2015 IEEE* (694-703).

Rolfe, G. (2006). Validity, trustworthiness and rigour: Quality and the idea of qualitative research. *Journal of Advanced Nursing*, *53*(3), 304-310.

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.

Sale J.E.M., Lohfeld L., & Brazil K. (2002). Revisiting the quantitative-qualitative debate. *Implications for Mixed-Methods Research*, *36*, 43-53.

Sandelowski M. (1993). Rigor or rigor mortis: The problem of rigor in qualitative research revisited. *Advances in Nursing Science 16*(2), 1–8.

Saunders M., Lewis P., & Thornhill, A. (2009). *Research methods for business students* (4th ed.). Essex, UK: Pearson Education Limited.

Schneckenberg D. (2014). Easy, collaborative and engaging – the use of cloud computing in the design of management classrooms. Educational Research, published by Routledge Taylor & Francis Group *4*(56), 412-435.

Seale, C. (1999). Quality in qualitative research. *Qualitative Inquiry*, *5*(4), 465-478.

Seiter, L., & Foreman, B. (2013) 'Modeling the learning progressions of computational thinking of primary grade students', in *Proceedings of the ninth Annual International ACM Conference on International Computing Education Research 13*, 59–66. doi: 10.1145/2493394.2493403.

Selby, C. C. (2015) Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (80-87). ACM.

Selby, C. C. (2012). Promoting computational thinking with programming. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (74-77). ACM.

Selby, C., & Woollard, J. (2013). *Computational thinking: the developing definition* University of Southampton (E-print). Retrieved from https://eprints.soton.ac.uk/356481/.

Selby, C., & Woollard, J. (2014). Refining an understanding of computational thinking. *Author's Original*, 1-23.

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, *15*(2), 4-14.

Shulman, L. (1987). Knowledge and teaching: Foundations of the new reform. *Harvard Educational Review*, *57*(1), 1-23.

Sllame, A. M., & Jafaray, M. (2013). Using simulation and modeling tools in teaching computer network courses. *In IT Convergence and Security (ICITCS), International Conference on IEEE*, 1-4.

Stenbacka, C. (2001). Qualitative research requires quality concepts of its own. *Management Decision*, 39(7), 551-555

Strauss, A. and Corbin, J. (2008) *Basics of qualitative research* (3rd ed.). Thousand Oaks, CA: Sage.

Sun, L., Wu, J., Zhang, Y., & Yin, H. (2013). Comparison between physical devices and simulator software for Cisco network technology teaching. In *Computer Science & Education (ICCSE), 2013 8th International Conference on* (1357-1360).

Tashakkori A., & Teddlie C. (Eds.) (2003). *Handbook of mixed methods in social and behavioural research*. Thousand Oaks, CA: Sage.

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (pp. 120-129). ACM.

Ten Dam, G., & Volman, M. (2004). Critical thinking as a citizenship competence: Teaching strategies. *Learning and Instruction*, *14*(4), 359-379.

Ulger, K. (2016). The relationship between creative thinking and critical thinking skills of students. *Hacettepe Universitesi Egitim Fakultesi Dergisi-Hacettepe University Journal of Education*, *31*(4), 695-710.

Van Driel, J. H., Verloop, N., & De Vos, W. (1998). Developing science teachers' pedagogical content knowledge. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, *35*(6), 673-695.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4), 715-728.

Voskoglou, M. G., & Buckley, S. (2012). Problem solving and computational thinking in a learning environment. *Egyptian Computer Science Journal, ECS*, *36*(4), 28-46.

Vyas, S., & Mishra, P. (2002). The re-design of an after-school reading club. *Hanging out: After-school community-based programs for children*, 75-93.

Vygotsky, L.S. (1978). *Mind in society*. Harvard University Press. Cambridge, MA.

Wannous, M., & Nakano, H., (2010). NVLab, a networking virtual web-based laboratory that implements virtualization and virtual network computing technologies. *Learning Technologies, IEEE Transactions*, *3*(2), 129-138.

Weitzman, E.A., and Miles, M.B. (1995). *Computer programs for qualitative data analysis*. Thousand Oaks, CA: Sage.

Willis, C. L., & Miertschin, S. L. (2005). Mind tools for enhancing thinking and learning skills. *In Proceedings of the 6th Conference on information Technology Education* 249-254. ACM.

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

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. (2010). Computational thinking: What and why? Retrieved from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Wing, J.M. (2011). Computational thinking. In *VL/HCC* (p.3). Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/WingCTPrez.pdf

Wing, J.M. (2014). Computational thinking benefits society. *Social Issues in Computing*. New York: Academic Press. *Artigo disponível e consultado em: Socialissues. cs. toronto. edu*.

Xu, L., Huang, D., & Tsai, W.T. (2014). Cloud-based virtual laboratory for network security education. *Education, IEEE Transactions*, *57*(3), 145-150.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J.T. (2011). Introducing computational thinking in education courses. *In Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, 465-470.

Yalcin, N., Altun, Y., & Kose, U. (2015). Educational material development model for teaching computer network and system management. *Computer Applications in Engineering Education*, *23*(4), 621-629.

Yeganeh, H., Su, Z., & Chrysostome, E. V. M. (2004). A critical review of epistemological and methodological issues in cross-cultural research. *Journal of Comparative International Management*, *7*(2), 66-86.

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

Zhao, Y. (Ed.). (2003). *What teachers should know about technology: Perspectives and practices*. Greenwich, CT: Information Age.

Appendix 3.01 Students' Online Survey (SoS)

Students' Understanding on Computational Thinking

1. Could you please specify your gender:
   - Male
   - Female
   - Prefer not to say

2. Could you please indicate the level of your course?
   - Undergraduate
   - Postgraduate

3. Could you please indicate your route to coming to study at university?
   - College studying for BTEC
   - College studying for A-levels
   - Sixth form studying for A-level
   - Others, Specify please

4. In your own view, what is computational thinking?

5. What are the main ideas of computational thinking?
   - Abstraction (*making an artefact more understandable by hiding complex details*)
   - Decomposition (*breakdown artefact into solvable parts*)
   - Generalisation (*identifying patterns & commonalities in solving problems of artefact*)
   - All the above

6. When attempting problem-solving tasks in computer networking design, you first need to have developed:
   - Critical thinking skills
   - Computational thinking skills
   - General knowledge about computing
   - All the above

7. What is your first strategy when solving a complex computer network design problem?
   - Trial and error
   - Research and understand the problem first
   - Breakdown the main problem into smaller manageable problems
   - Others, specify:

For questions 8 – 12, please answer by selecting one of the following options:

(a) strongly agree (**b**) **agree (c) disagree** (**d**) **strongly disagree**

8. Computational thinking helps me to solve complex network design problems

   a b c d

9. There is no difference between computational thinking and critical thinking

   a b c d

10. Computational thinking can be integrated in learning computer network systems designs

    a b c d

11. Having good background knowledge in computer science helps to apply computational thinking when designing complex computer network systems

    a b c d

12. I really do not need computational thinking skills to design and troubleshoot network systems

    a b c d

13. How can computational thinking help in solving computer network systems?

14. Any other comment about computational thinking?

Students' perception of simulation software to facilitate students' CT

15. Which of the following simulation softwares are you familiar with?
    - GNS3
    - Packet Tracer
    - VIRL
    - Others, specify below please:

16. Which of the following elements of computational thinking can be demonstrated by the use of simulation software you have selected on question 15
    - Abstraction (*making an artefact more understandable by hiding complex details*)
    - Decomposition (*breakdown artefact into solvable parts*)
    - Generalisation (*identifying patterns & commonalities in solving problems of artefact*)
    - All the above

17. In your view, what is your experience and opinion of using the simulation software you have selected in question 15 to facilitate your computational thinking?

For questions 18 – 21, please answer by selecting one of the following options based on your opinion on using simulation software in facilitating your computational thinking:

(**a**) **strongly agree** (**b**) **agree (c) disagree** (**d**) **strongly disagree**

18. Simulation software provides a good platform in facilitating my computational thinking when designing computer networks

a b c d

19. Simulation software provides more flexibility than real physical device in designing complex networks which can help in developing my computational thinking.

a b c d

20. Simulation software provides a good platform to troubleshoot network design thereby develops my computational thinking skills

a b c d

21. Simulation software does not provide real practice and experiences as compared to physical devices therefore it is not ideal to develop my computational thinking

a b c d

Please indicate by typing your email address below if you would like to be contacted for an in-depth interview

***Your contact email address**:*

The End!

Thank You

Appendix 3.02 Lecturers' Online Survey (LoS)

Lecturers' understanding on Computational Thinking

1. How long have you been lecturing in computing department?
   - ☐ Less than 2 years
   - ☐ 2 to 4 years
   - ☐ 5 to 7 years
   - ☐ 8 + years

2. If you worked in the computing industry before becoming a lecturer, please specify your industrial work experience?
   - ☐ Less than 2 years
   - ☐ 3 to 5 years
   - ☐ 5 to 8 years
   - ☐ 9 + years

3. Specify your highest academic qualification
   - ☐ Doctorate
   - ☐ Masters
   - ☐ Others, specify

4. Which course(s) do you mainly teach on?
   - ☐ Computer network course
   - ☐ Computing with information security systems course
   - ☐ Computing with Forensic course
   - ☐ None of the above

5. In your own view, what is computational thinking?

6. What are the main ideas of computational thinking?
   - ☐ Abstraction (*making an artefact more understandable by hiding complex details*)
   - ☐ Decomposition (*breakdown artefact into solvable parts*)
   - ☐ Generalisation (*identifying patterns & commonalities in solving problems of artefact*)
   - ☐ All the above

7. When attempting problem-solving tasks in computer networking design, you first need to have developed:
   - Critical thinking skills
   - Computational thinking skills
   - General knowledge
   - All the above

8. What do you expect to be students' first strategy when solving a complex computer network design problem?
   - Trial and error
   - Research and understand the problem first
   - Breakdown the main problem into smaller manageable problems
   - Others, specify:

For questions 9 – 13, please answer by selecting one of the following options:

(b) strongly agree (**b**) **agree (c) disagree** (**d**) **strongly disagree**

9. Computational thinking helps students solve complex network design problems

   a b c d

10. There is no difference between computational thinking and critical thinking

    a b c d

11. Computational thinking can be integrated in teaching computer network systems designs

    a b c d

12. Having good background knowledge in computer science helps to apply computational thinking when designing complex computer network systems

    a b c d

13. Students do not need computational thinking skills to design and troubleshoot network systems

    a b c d

14. How can computational thinking skills be integrated in understanding how networks are designed?

15. How can computational thinking help in solving computer network systems?

16. Any other comment about computational thinking?

Lecturers' perception of simulation software to facilitate students' Computational Thinking

17. Which of the following simulation softwares are you familiar with?
    - ☐ GNS3
    - ☐ Packet Tracer
    - ☐ VIRL
    - ☐ Others, specify below please:

18. Which of the following elements can be demonstrated by the use of simulation software you have selected on question **17**?
    - ☐ Abstraction (*making an artefact more understandable by hiding complex details*)
    - ☐ Decomposition (*breakdown artefact into solvable parts*)
    - ☐ Generalisation (*identifying patterns & commonalities in solving problems of artefact*)
    - ☐ All the above

19. In your view, how does the use of simulation software you have selected in question **17** facilitate students' computational thinking?

20. When designing a piece of assessment, which elements do you like your students to demonstrate when using simulation software?
    - ☐ Abstraction (*making an artefact more understandable by hiding complex details*)
    - ☐ Decomposition (*breakdown artefact into solvable parts*)
    - ☐ Generalisation (*identifying patterns & commonalities in solving problems of artefact*)
    - ☐ All the above

For questions 21 – 23, please answer by selecting one of the following options based on your opinion on using simulation software in facilitating students' computational thinking:

(**a**) **strongly agree** (**b**) **agree (c) disagree** (**d**) **strongly disagree**

21. Simulation software helps in facilitating students' computational thinking when designing computer networks

a b c d

22. I don't think simulation software provides a platform to develop students' computational thinking in designing computer networks

a b c d

23. Simulation software provides a good platform to troubleshoot network design and therefore develops students' computational thinking

a b c d

Please indicate by typing your email address below if you would like to be contacted for an in-depth interview

***Your contact email address**:*

The End!

Thank You

Appendix 3.03 Focus Group Sample Questions for Students

These are only the key questions. More questions were generated within the focus group. The discussion was intended to be fluid so as to capture data which was rich of students' understanding, perceptions and experiences in using simulation software to facilitate their computational thinking:

1. Can you describe or explain the strategies you normally follow when solving complex network designs on a simulation software platform?
   (This question is aimed at understanding students' application on: abstraction, decomposition and generalisation when they design and solve complex problems on the network)

2. Can you explain your opinion on using simulation software to solve network design?
   (the idea is to capture some of their perceptions in the use of simulation software in facilitating their computational thinking)

3. From the given scenario what were your challenges in solving network design problems and how did you handle them?
   (the idea of this question is to find out if students can demonstrate how they identified problems, how they solved them and how they provided their solution(s) – the whole process is to find out if students can demonstrate their skills in abstraction, decomposition and generalisation)

4. Have you applied your computational thinking skills in other subjects which are not computer related module? If yes, can you explain how
   (This is to find out how students have applied their computational thinking skills outside their subject speciality)

5. What are your recommendation in the use of simulation software in facilitating your computational thinking skills
   (the idea of this question is to elicit if students can provide some new ideas which could possibly help, if possible, how simulation software could facilitate students computational thinking)

Appendix 3.04 Interview Sample Questions for Students

Following on the online survey conducted, there are a few areas I would like to investigate further. Feel very free to ask any question if you are not clear of anything I will be asking you. Also feel free to bring in your own opinion and ideas to the questions I will be asking you. This study focuses on the application of Computer Networks students' computational thinking via the use of simulation software.

1. How many years have you been learning computer networks?

2. What is your own understanding by the term computational thinking?

3. When designing computer network systems:
    a. How do you remove unnecessary details in order to make the design or problem simple to understand? (*abstraction*)
    b. How do you solve complex problems in order to make them easily solvable? (*decomposition*)
    c. What strategies do you normally take when troubleshooting a network design system? (*generalisation*)

4. What type of simulation software are you familiar with?

5. How would you develop computational thinking skills when designing computer networks on simulation software?

6. What are some of the challenges, if any, in developing computational thinking skills when designing computer systems on simulation software?

7. Any particular comments, remarks or opinion about the use of simulation software in facilitating your computational thinking?

Appendix 3.05 Interviews Sample Questions for Lecturers

Following on the online survey conducted, there are a few areas I would like to investigate further. Feel very free to ask any question if you are not clear of anything I will be asking you. Also feel free to bring in your own opinion and ideas to the questions I will be asking you. This study focuses on the application of Computer Networks students' computational thinking via the use of simulation software.

1. What is your subject speciality?

2. How long have you been teaching this subject?

3. What is your own understanding by the term computational thinking?

4. How do you design your assessments or lessons that facilitate students' computational thinking?

5. What type of simulation software are you familiar with?

6. Can the use of simulation software help to facilitate students' computational thinking? If so explain, how?

7. What are some of the challenges you may have come across when developing students' computational thinking through the use of simulation software?

8. Some elements of computational thinking are abstraction, decomposition and generalisation, can you please explain how you apply these elements in your teaching?

9. Any particular comments, remarks or opinion about the use of simulation software in facilitating students' computational thinking?

Appendix 3.06 Students' Reflective Sample Report Form

Problem-Solving Tasks based on the scenario provided	Strategies applied	Reflective Remarks/Report
Task 1. Planning a. Work out IPv4 & IPv6 addresses (*as appropriate*) for the following areas: i. LAN ii. VLANs iii. WAN		**Remarks**
		Remarks
b. show your plan for: i. DHCP ii. DHCP Relay agent iii. ACLs & NAT policies iv. VLANs for each LAN v. Inter-VLANs as appropriate		
		Remarks
Task 2. design & demonstrate i. Network infrastructure (topology) ii. appropriate assignment of IP addresses per each LAN, VLAN and WAN iii. configuration and demonstration of all what is included in your planning (from Task 2)		
	Reflective Report	
Task 3 Reflective Report with recommendations. This should include: i. challenges encountered ii. how your computational thinking skills may have helped in tackling tasks e.g. troubleshooting techniques applied iii. areas to improve and add in the network infrastructure with justification iv. recommendation (i.e. security, training)		

Appendix 3.07 Problem-Solving Task 1 (*only routing table*)

OUTPUT ROUTING TABLES:

R1

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

    10.0.0.0/30 is subnetted, 5 subnets
O       10.10.10.0 [110/128] via 10.10.10.5, 01:56:34, Serial0/0
C       10.10.10.4 is directly connected, Serial0/0
C       10.10.10.8 is directly connected, Serial0/1
D       10.10.10.12 [90/2681856] via 10.10.10.9, 01:56:39, Serial0/1
O       10.10.10.16 [110/128] via 10.10.10.5, 01:56:34, Serial0/0
        172.16.0.0/16 is variably subnetted, 13 subnets, 4 masks
C       172.16.1.0/27 is directly connected, FastEthernet0/0
D       172.16.1.32/28 [90/2684416] via 10.10.10.9, 01:56:37, Serial0/1
O       172.16.2.0/29 [110/129] via 10.10.10.5, 01:56:34, Serial0/0
O       172.16.2.8/29 [110/129] via 10.10.10.5, 01:56:34, Serial0/0
O       172.16.2.16/29 [110/129] via 10.10.10.5, 01:56:34, Serial0/0
D       172.16.2.64/27 [90/2172416] via 10.10.10.9, 01:56:39, Serial0/1
C       172.16.3.0/25 is directly connected, FastEthernet0/1
O       172.16.3.128/29 [110/65] via 10.10.10.5, 01:56:34, Serial0/0
O       172.16.3.136/29 [110/65] via 10.10.10.5, 01:56:34, Serial0/0
O       172.16.3.144/29 [110/65] via 10.10.10.5, 01:56:34, Serial0/0
O       172.16.3.192/29 [110/129] via 10.10.10.5, 01:56:34, Serial0/0
D       172.16.4.0/27 [90/2172416] via 10.10.10.9, 01:56:39, Serial0/1
D       172.16.4.128/25 [90/2684416] via 10.10.10.9, 01:56:37, Serial0/1
C       192.168.1.0/24 is directly connected, Loopback0
S*      0.0.0.0/0 is directly connected, Loopback0
```

R2

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route
```

```
Gateway of last resort is 10.10.10.1 to network 0.0.0.0

     10.0.0.0/30 is subnetted, 5 subnets
C        10.10.10.0 is directly connected, Serial0/0
O        10.10.10.4 [110/128] via 10.10.10.1, 02:08:36, Serial0/0
O  E2  10.10.10.8 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O  E2  10.10.10.12 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O        10.10.10.16 [110/128] via 10.10.10.1, 02:08:36, Serial0/0
         172.16.0.0/16 is variably subnetted, 13 subnets, 4 masks
O E2  172.16.1.0/27 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O E2  172.16.1.32/28 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
C       172.16.2.0/29 is directly connected, FastEthernet0/0.15
C       172.16.2.8/29 is directly connected, FastEthernet0/0.25
C       172.16.2.16/29 is directly connected, FastEthernet0/0.30
O E2  172.16.2.64/27 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O E2 172.16.3.0/25 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O       172.16.3.128/29 [110/65] via 10.10.10.1, 02:08:36, Serial0/0
O       172.16.3.136/29 [110/65] via 10.10.10.1, 02:08:36, Serial0/0
O       172.16.3.144/29 [110/65] via 10.10.10.1, 02:08:36, Serial0/0
C       172.16.3.192/29 is directly connected, FastEthernet0/1
O E2 172.16.4.0/27 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O E2 172.16.4.128/25 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O E2 192.168.1.0/24 [110/20] via 10.10.10.1, 02:08:36, Serial0/0
O*E2 0.0.0.0/0 [110/1] via 10.10.10.1, 01:58:35, Serial0/0
```

R3

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is 10.10.10.6 to network 0.0.0.0

    10.0.0.0/30 is subnetted, 5 subnets
C       10.10.10.0 is directly connected, Serial0/1
C       10.10.10.4 is directly connected, Serial0/0
O E2 10.10.10.8 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                         [110/20] via 10.10.10.6, 00:11:00, Serial0/0
O E2 10.10.10.12 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                          [110/20] via 10.10.10.6, 00:11:00, Serial0/0
C       10.10.10.16 is directly connected, Serial0/2
        172.16.0.0/16 is variably subnetted, 14 subnets, 5 masks
O E2  172.16.1.0/27 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                             [110/20] via 10.10.10.6, 00:11:00, Serial0/0
O E2 172.16.1.32/28 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
```

```
                    [110/20] via 10.10.10.6, 00:11:00, Serial0/0
C       172.16.1.192/26 is directly connected, FastEthernet0/1
O       172.16.2.0/29 [110/65] via 10.10.10.2, 00:11:00, Serial0/1
O       172.16.2.8/29 [110/65] via 10.10.10.2, 00:11:00, Serial0/1
O       172.16.2.16/29 [110/65] via 10.10.10.2, 00:11:00, Serial0/1
O  E2 172.16.2.64/27 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                    [110/20] via 10.10.10.6, 00:11:00, Serial0/0
O  E2 172.16.3.0/25 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                   [110/20] via 10.10.10.6, 00:11:00, Serial0/0
C       172.16.3.128/29 is directly connected, FastEthernet0/0.15
C       172.16.3.136/29 is directly connected, FastEthernet0/0.25
C       172.16.3.144/29 is directly connected, FastEthernet0/0.30
O       172.16.3.192/29 [110/65] via 10.10.10.2, 00:11:00, Serial0/1
O  E2  172.16.4.0/27 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                    [110/20] via 10.10.10.6, 00:11:00, Serial0/0
O E2   172.16.4.128/25 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                     [110/20] via 10.10.10.6, 00:11:00, Serial0/0
O E2   192.168.1.0/24 [110/20] via 10.10.10.18, 00:11:00, Serial0/2
                     [110/20] via 10.10.10.6, 00:11:00, Serial0/0
O*E2 0.0.0.0/0 [110/1] via 10.10.10.6, 00:00:44, Serial0/0
```

R4

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is 10.10.10.10 to network 0.0.0.0

10.0.0.0/30 is subnetted, 5 subnets
D  EX  10.10.10.0 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
D  EX  10.10.10.4 [109/2560512256] via 10.10.10.10, 01:39:42, Serial0/0
C       10.10.10.8 is directly connected, Serial0/0
C       10.10.10.12 is directly connected, Serial0/1
C       10.10.10.16 is directly connected, Serial0/2
        172.16.0.0/16 is variably subnetted, 13 subnets, 4 masks
D       172.16.1.0/27 [90/2172416] via 10.10.10.10, 01:39:42, Serial0/0
D       172.16.1.32/28 [90/2172416] via 10.10.10.13, 01:39:40, Serial0/1
D  EX  172.16.2.0/29 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
D  EX  172.16.2.8/29 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
D  EX  172.16.2.16/29 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
C       172.16.2.64/27 is directly connected, FastEthernet0/1
D       172.16.3.0/25 [90/2172416] via 10.10.10.10, 01:39:42, Serial0/0
D  EX  172.16.3.128/29 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
D  EX  172.16.3.136/29 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
```

```
D  EX  172.16.3.144/29 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
D  EX  172.16.3.192/29 [109/2560512256] via 10.10.10.10, 01:39:37, Serial0/0
C        172.16.4.0/27 is directly connected, FastEthernet0/0
D        172.16.4.128/25 [90/2172416] via 10.10.10.13, 01:39:40, Serial0/1
D        192.168.1.0/24 [90/2297856] via 10.10.10.10, 01:39:42, Serial0/0
D*EX   0.0.0.0/0 [109/3449856] via 10.10.10.10, 01:39:42, Serial0/0
```

R5

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
* - candidate default, U - per-user static route, o - ODR
P - periodic downloaded static route

Gateway of last resort is 10.10.10.14 to network 0.0.0.0

10.0.0.0/30 is subnetted, 5 subnets
D   EX   10.10.10.0 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D   EX   10.10.10.4 [170/2561024256] via 10.10.10.14, 01:33:37, Serial0/0
D         10.10.10.8 [90/2681856] via 10.10.10.14, 01:33:37, Serial0/0
C         10.10.10.12 is directly connected, Serial0/0
D   EX  10.10.10.16 [170/2560512256] via 10.10.10.14, 01:33:37, Serial0/0
          172.16.0.0/16 is variably subnetted, 15 subnets, 4 masks
D         172.16.1.0/27 [90/2684416] via 10.10.10.14, 01:33:37, Serial0/0
C         172.16.1.32/28 is directly connected, FastEthernet0/1.15
C         172.16.1.48/28 is directly connected, FastEthernet0/1.25
C         172.16.1.64/28 is directly connected, FastEthernet0/1.30
D  EX   172.16.2.0/29 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D  EX   172.16.2.8/29 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D  EX   172.16.2.16/29 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D         172.16.2.64/27 [90/2172416] via 10.10.10.14, 01:33:37, Serial0/0
D         172.16.3.0/25 [90/2684416] via 10.10.10.14, 01:33:37, Serial0/0
D  EX   172.16.3.128/29 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D  EX   172.16.3.136/29 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D  EX   172.16.3.144/29 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D  EX   172.16.3.192/29 [170/2561024256] via 10.10.10.14, 01:33:35, Serial0/0
D         172.16.4.0/27 [90/2172416] via 10.10.10.14, 01:33:37, Serial0/0
C         172.16.4.128/25 is directly connected, FastEthernet0/0
D         192.168.1.0/24 [90/2809856] via 10.10.10.14, 01:33:37, Serial0/0
D*EX    0.0.0.0/0 [170/3961856] via 10.10.10.14, 01:33:37, Serial0/0
```

Appendix 3.08 Problem-Solving Task 2

Case Scenario

You have just been recruited in one of the outstanding companies in the United Kingdom specialising in design and implementation of computer network infrastructures for medium and large organisations. You have just graduated from X University and you have been awarded a Master's Degree in Network Professional.

The company has just won a tender to design and implement a network infrastructure for an organisation which has branches in Sheffield, Derby, Leicester and Manchester. You have been requested to design this network to an efficient working condition. You have also been requested to provide your professional recommendations. The company is also expecting your expertise in creativity and innovation of your network infrastructure design.

Here are the specifications which have been outlined on what is expected.

Manchester (Headquarters):

- This is the Headquarters for this organisation. Therefore, all other branches will need to access the Internet via Manchester. The ISP should therefore directly be connected to Manchester.
- Manchester has two main LANs. One should be running on IPv6 with over 5000 users while the other LAN with only 779 users should be running on IPv4.
- Sheffield is envisaged to be an experimental base running on different protocols - therefore you need to consider an appropriate WAN protocol to connect between Sheffield and Manchester.
- The company would like to have a secure link between Sheffield and Manchester.

- Derby and Leicester are running on a limited budget, therefore you need to choose an appropriate WAN technology to connect with Manchester. It is recommended that Derby and Leicester should use a common WAN link to Manchester which is cheaper and easy to manage.

Sheffield

- Sheffield also has two LANs. One LAN is used by a Research Centre with about 100 workstations, which needs to be running on IPv6 only. The other LAN is subdivided into2 branches. Branch1 has approximately 2700 people, while Branch2 has approximately 987 people. These two branches should both run on IPv4.

  The Plan for Sheffield consists of two different distinct Networks as follows:
- There will be very few technical members of staff to support, maintain and manage the network at Sheffield. Most of the technical support, maintenance, network administration and management should therefore be considered to be carried out by the staff at Manchester.
- Also, consider minimising the traffic between Manchester and Sheffield
- All computing devices in Sheffield should have access to the Internet and also be able to access other branches.

Derby

- Derby has two main buildings (i.e LANs). Building X, has Sales, Production and Marketing departments with a capacity of 150, 200 and 500 people respectively. Building Y, has Administration and Support Staff with a capacity of 450 and 300 respectively.
- Derby requires both buildings to be running IPv4 only.

- Derby has enough trained and qualified technical staff in order to maintain and manage their network.
- All computing devices at Derby should have access to the Internet and to be able to access other branches too

Leicester

- Leicester is a branch that requires a considerable of amount of security
- Currently there is only one LAN (but plans are underway to expand the branch). This branch has about 6 people who often use wireless connectivity to the LAN. The 6 people belong to the management team of the Leicester branch. However, it has a small sub-branch for the Sales, Marketing and Accounts departments. There are approximately 110 people for sales, 52 people for marketing and 20 for the accounts department.
- For security reasons most of their workstations will have restrictions to Internet Access only. Most of the employees will only require internet access, but are prohibited to download or upload files. There are 5 senior managers in the Sales and Marketing departments who require full access to the Internet and access to other branches.

Recommendations, innovation and creativity

You will be allocated about a maximum of 45 minutes to demonstrate and present your solution(s) to the requirements on the final week of the course. Your innovation and creativity on the network design and implementation will need to be clearly presented during your demonstration. Your recommendation should cover approximately a page of A4, clearly demonstrating your understanding of the requirements and any areas to

consider for future improvements. You should also consider issues of network utilization, performance, and security within the wide area network infrastructure.

Schedule for demonstration and presentation will be provided later

Good Luck!

Appendix 3.09 Overall Reflective Sample Report Form

	Area of focus	**Meanings (themes)**	**Your experience/reflection/feedback based on problem solving tasks on three experiments you did on LAN, WAN and Security modules**
Core of CT understudy	**Abstraction** (thought process)	• Artefact more understandable • Good representation of a system	
	Decomposition (thought process)	• Problem simplification • Providing easy solutions (*understood, solved, developed & evaluated separately*)	
	Generalisation (thought process)	• Pattern identification (similarities & connections) • Solved based on previous solutions & prior knowledge	

Appendix 3.10 Sample for First Stage of Data Coding without NVivo

What is computational thinking

The definitions of **Themes** have been formed based on the core concepts of computational thinking - **Abstraction (AB)**, **Decomposition (DE)**, & **Generalisation (GE)**. These core concepts have been adopted and further explained based on the literature review (Csizmadia et al. 2015; Grover & Pea, 2013; Selby & Woollard, 2010; Wing, 2006, 2008). Other Themes have also been formed based on the emerging ideas and both themes have been colour coded as follows:

Prescribed codes based on the concept of CT

Abstraction (AB),
Decomposition (DE), &
Generalisation (GE)

Emerged Themes

Problem-solving=26
Logical thinking=7
Think like computers=6
Technical way of thinking=3
Computer solution = 6

Primary Theme	Sub-themes	The exact quote from students' responses (if a student response has more than one different idea then the response is copied to other sub-themes respectively)

Abstraction – AB (*complexity of a concept that needs to be made concrete***)**
 	
• Artefact more understandable
  ○ reducing unnecessary details
  ○ choosing the right details to hide
  ○ making problem become easier – without losing important details
 	
• Good representation of a system
	AB (clear understanding to be used by humans and computers)
	
	AB (abstraction)
		"the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out"
		
		"ability to think abstract"

Decomposition – DE (*the process of breaking down an abstract to the level that it can be understood and solved***)**
 	
• Problem simplification
  ○ makes complex problem easy to solve (solvable problems)
  ○ novel situations better understood
  ○ large system easy to design
 	
• Providing easy solutions (i.e. *understood, solved, developed & evaluated separately*)
	DE (breakdown-of-problems)	"breaking down a larger issue into smaller chunks to make it easier to solve"
		
		"It is a way of thinking, similar to a machine i.e. mathematical but could also be done in working out a way to break down a large problem into smaller problems, to solve the overall bigger problem"
		
		"Thinking in a logical matter in order to **solve a problem** (possibly by breaking it down into smaller parts)"
		
		"It is the process of breaking down a problem into parts, analysing the inputs for patterns, identifying the key bits and developing a solution"
		
		"the idea of being able to break the information or detail into bitesize features to solve towards the main objective"
		
		"I think it's a thinking method in which a problem is divided into several part to find a solution for each part more easily"
		
		"Abstracting a big problem into smaller ones which can then be managed and solved easier. Constructing your thinking and ideas in a step.by step manner"
		
		"I am unaware what computational thinking actually is, I assume it is something to do with the breaking down of computer related tasks and simplifying problems in regards to the chosen topic"
		
		"Breaking a problem down into parts to solve them individually rather than taking on one problem as a whole"
		
		"It is a breakdown of the problem which is easier to be solved"
		
		"breaking down a problem in such a way that makes it simpler and solvable by a person or machine"

		"Taking a very general, or complex, idea and solving it in manageable pieces" "breaking down a problem into a series of tasks" "Techniques and decomposition"
Generalisation – GE (*the process of identifying patterns, similarities, through prior knowledge in solving an abstract***)** • Pattern identification (similarities, connections, commonalities) • Solved based on previous solutions & prior knowledge	GE (pattern identification) GE (Problems identification) GE (problem-identification) GE (previous knowledge/skills/etc) GE (connections) GE (chronological order)	"It is the process of breaking down a problem into parts, analysing the inputs for patterns, identifying the key bits and developing a solution" "identifying what you have to do and find reasonable ways to **solve the problem"** "Computational thinking is identifying a problem and find ways to solve it based on your general computing knowledge" "coming up with **a solution to a problem** and expressing it in a manner which others can follow to solve the given problem" "Abstracting a big problem into smaller ones which can then be managed and solved easier. Constructing your thinking and ideas in a step by step manner"
Emerged Themes	Sub-emerged themes	
Problem-solving	**Applying thought process for logical analysis** **Steps for problem-solving** **Thought process that a computer understands** **Problem-solving**	**"Applying a thought process similar to that of a computer by taking away emotion, prejudice and external factors to allow a purely logical and relevant analysis"** **"A way to solve problems while expressing the steps of solution"** **"thought process/solution** delivered in a way that a computer will understand" "Thinking in a logical matter in order to **solve a problem** (possibly by breaking it down into smaller parts)"

	Problem-solving technique	"It assists people in **tackling challenging problems and understanding them on the way to devising their solutions** and *these solutions can then be effectively performed by people or machines"*
	Problem-solving	"coming up with **a solution to a problem** and expressing it in a manner which others can follow to solve the given problem"
	Thought process involved	**"the thought processes involved in formulating a problem** and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out"
	Problem-solving	**"Formulating a problems and finding a solution"**
	Problem-solving by a computer	"The way a ***computer decides how to solve a problem"***
	Problem solving	"identifying what you have to do and find reasonable ways to **solve the problem"**
	Problem-solving process	"computational thinking is **a problem solving** in **a process that machine can carry out solution"**
	Thought process to automated solution	**"A process of thinking that involves providing an automated solution to a problem"**
	Thought process	"understanding and **analysing through deep thinking"**
	Computer problem solving	"Applying a logical reasoning in ***solving computing operations"***
	problem solving	**"analysing the solutions to the problems such a way humans and computer's do"**
	Problem solving	**"solving problems and expressing its solution"**
	Thought process	**"A way of thinking that means your brain understands computers"**

	Thought process ***Problem solving like computers*** **Steps for problem-solving** **Thought problems in problem-solving** **Logical problem-solving** Logical thinking**/thought process** **Problem-solving** **Complex-thinking (thought process)**	**“is the way of thinking about the problem that a computer and a human can understand”** **“Providing a answer or solution, in a manor similar to a computer system”** “A way to **solve problems while expressing the steps of solution”** **“Thought involved in problem solving”** **“Systematic and logical problem solving”** **“logical thought process”** **“Solving and understanding problems”** **“complex thinking”**
Logical thinking	Logical thinking Logical reasoning/thinking Logical thinking Logical thinking Logical thinking Logical thinking Logical thinking**/thought process** Logical thinking Logical thinking	“Thinking in a logical matter in order to **solve a problem** (possibly by breaking it down into smaller parts)” “Applying a logical reasoning in **solving computing operations”** “Thinking logically, exploring ideas and possibilities” “thinking in a logical manor” “Thinking Logically” “thinking logically” **“logical thought process”** **“**logical **thought process”** **“Systematic and** logical **problem solving”**

Think like computers	Thinking like a machine	"It is a way of thinking, similar to a machine i.e. mathematical but could also be done in working out a way to break down a large problem into smaller problems, to solve the overall bigger problem"
	Computers thinking	"How a computer thinks and how humans can make a computer to think like them"
	Thinking like a computer	"Thinking like a PC, following commands yes or no"
	Think like a computer	"thinking in a way a computer processes actions"
	Thinking like a computer	"Thinking like a computer"
Technical way of thinking	technical way of thinking	"In my view computational thinking is a technical way of thinking"
	technical way of thinking understand technical terms	"Thinking in a technical way, able to easily understand technical terms"
Computer solution	**Computers calculating answers**	**"AI, allowing a computer to calculate answers,** where we would generally 'think' about them"
	Machine carries out solutions	"computational thinking is **a problem solving** in **a process that machine can carry out solution"**
	Use of computer for simulation	**"The use of computers to do simulations and other work"**
	computer analysis to problem solution	**"analysing the solutions to the problems such a way humans and computer's do"**
	Efficiency use of computers	**"Making real world tasks more efficient with the use of computers"**
	Computer solutions to various tasks	**"Ability to use computer systems to perform various functions using softwares"**
	Computer-solving problem	**"The way a computer decides how to solve a problem"**

Appendix 3.11 Second Stage of Data Coding with NVivo

16/09/2018 20:15

Coding Summary By Node

Computational Thinking

16/09/2018 20:15

Aggregate	Classification	Coverage	Number Of Coding References	Reference Number	Coded By Initials	Modified On

Node

Nodes\\RQ1) What is students' & lecturers' understanding of CT

Document

Internals\\LoS\\LoS - Lecturers' opinion on how CT may be integrated in network design

Yes	Survey	0.7010	9			
				1	SM	16/03/2018 10:12

Breaking down a complex network into smaller manageable tasks, which is computational thinking.

2	SM	15/08/2018 23:01

Understanding Logic, the fundamentals of programming and at the base level how protocols are designed.

3	SM	15/08/2018 23:02

Fundamental understanding of underlying network operations

4	SM	16/03/2018 11:21

I'm not sure that Computational Thinking is helpful

Aggregate	Classification	Coverage	Number Of Coding References	Reference Number	Coded By Initials	Modified On

Internals\\LoS\\LoS - Lecturers' opinion on how CT may help in solving network systems

Yes	Survey	0.6654	10			
				1	SM	31/03/2018 14:44

It is much simpler to both understand the system and to troubleshoot if the system is divided into smaller, more manageable chunks - these chunks can also be examined by other individuals who specialise in that particular field"

2	SM	13/08/2018 13:40

A network is really a set of abstract components that are connected. As the abstraction becomes more concrete the detail of the hots, connections, routes etc becomes clear.

3	SM	15/08/2018 22:33

Students need to be able to decompose and to abstract so that they are always thinking at a suitable level and not trying to understand the entire system at once

4	SM	26/03/2018 13:58

problems can be decomposed into discrete soluble portions

5	SM	16/03/2018 11:19

However, I don't think all problems can be tackled by computational thinking

6	SM	16/03/2018 11:04

It will make the pupil more innovative and can even help in foreseeing and designing networks

7	SM	26/03/2018 13:59

Breaking the networking down into computational components

8	SM	15/08/2018 23:02

Students need to understand what is going on from the lowest level, understanding the core components and how they work to appreciate what is going on

9	SM	26/03/2018 13:59

Internals\\PGFocus-Grp\\PGFG

Yes	Focus Group	0.1724	15

1	SM	15/08/2018 22:34

Its much harder to find what is the problem than solving each task as you go along.

2	SM	31/03/2018 17:52

Yes, I consider doing step by step procedure in solving problems, I use the road map to identify the problems. This

3	SM	31/03/2018 14:56

This is the process of breaking down complex problem into small manageable problems which can easily be solved and then bring them together to get the final holistic picture of the network

4	SM	26/03/2018 14:58

You need some knowledge as well.

5	SM	31/03/2018 16:12

You need to know how to configure a network, some basic information about networking, IP addresses and devices connected to it, you also need to know commands, you need to know all that….and have basic understanding of network itself. Otherwise if you do not have that basic understanding you can not solve any networking problem, no matter how you breakdown that problem; because evenif you see a fault, you wont be able to solve it

6	SM	31/03/2018 16:13

I also wanted to say, under the assumption that you have, basic knowledge that you need to fix a problem, for me its easy to look at the problem than looking at understanding what the actual finished product should look like

7	SM	15/08/2018 22:35

A network design is in such a way that it comes from a bit bits to be a holistic giant topology so in solving issues in a network you have to go from little bit. You have to look at the smallest bits

Aggregate	Classification	Coverage	Number Of Coding References	Reference Number	Coded By Initials	Modified On
				8	SM	31/03/2018 14:58
Therefore you must break every problem to the smallest bit look at it in detail and figure out what is the problem						
				9	SM	30/08/2018 16:13
its better to run show run command on the device that has a problem and compare the results to a similar device with the same configuration and go and see, because it is easy to overlook the simple mistakes because you want to look at complex mistakes						
				10	SM	31/03/2018 16:16
my previous learning has a significant role to play in solving a problem						
				11	SM	31/03/2018 14:57
I had so many problems, however, when I broke down the problems into small ones, It became so clear how to solve the problems						
				12	SM	30/08/2018 16:13
I then looked at the similarities of serial interfaces against other routers and then I was able to connect them together. I saw that there were networks which were sharing the same network hence I connected them together.						
				13	SM	31/03/2018 14:57
computational problems are helping me how I can solve any other problems in breaking them into small manageable solutions.						
				14	SM	31/03/2018 14:58
It [CT] has helped me in broaden my mind in handling problem, for instance in taking a problem and breaking it down into small bits						
				15	SM	22/08/2018 19:42
In my case, the entire process has given me more confidence in tackling problems. In all the assignments except this one gave me step by step of what I should do to get the exact results. While as this one, I had to figure out myself how to solve a problem and hence gave me much chance for thought processing						

Aggregate	Classification	Coverage	Number Of Coding References	Reference Number	Coded By Initials	Modified On
Internals\\PS-Task 1 - Reflective Report\\PGstud5 - PsTask 1						
Yes		0.0314	1			
				1	SM	29/03/2018 15:02

This a brilliant exercise where I have learnt many thing and am very confident about what am doing and what am configuring.

Internals\\PS-Task 2 - Reflective Report\\PGstud1 - PsTask 2

Yes		0.0274	2			
				1	SM	29/03/2018 18:06

Making use of my computational thinking skills helped to create the network as a whole, especially in troubleshooting non-convergence and errors, and to produce convergence while satisfying the necessary requirements

				2	SM	29/03/2018 18:10

The application of routing protocols round the network was for the purpose of connectivity in general but utilizing two different routing protocols in the Sheffield branch and redistributing them for connectivity with multiple protocols and different networks over different WAN links, and for it to work, was a result of utilizing computational thinking

Internals\\PS-Task 2 - Reflective Report\\PGstud2 - PsTask 2

Yes		0.0945	2			
				1	SM	29/03/2018 18:46

The way of using computational thinking was accurate because it gives you an ability to evaluate each step as long as you progress with your work from assigning IP address on Interfaces, creating virtual circuit Frame-relay on sub interfaces, Assigning VLANs, security routine on specified branch such as LEICESTER

				2	SM	29/03/2018 18:59

the Computational thinking which is probably the main point giving me intuition and stability in solving issues like this I gained ability in looking thoroughly into the routing tables of every devices, collect information and troubleshoot issues as quickly as possible.

Aggregate	Classification	Coverage	Number Of Coding References	Reference Number	Coded By Initials	Modified On

Internals\\Selected-Lecs-1-2-1\\Lec2 on 1-2-1 interviews

Yes	Interviews	0.0668	6			
				1	SM	20/03/2018 17:54

I understand the term computational thinking as sort of the term that taking the large problem that can not be solved on its own dividing into smaller pieces each of which can be solved on its own

				2	SM	31/03/2018 15:01

its just a method of splitting a large piece of job into smaller manageable chunks, I Think

				3	SM	31/03/2018 15:02

you try to split into chucks that you can look at everything

				4	SM	31/03/2018 15:02

experts, they don't have the theory behind so they could not solve a complex network problem by themselves but they could split the problem into smaller chunks that are manageable

				5	SM	31/03/2018 15:03

You are splitting problem into small manageable chunks but you are simply doing it into small manageable chunks in terms of time and effort instead of each individual….

				6	SM	31/03/2018 16:17

I think this goes back to the previous knowledge they have so for example in routing – all routing protocols are basically the same things and so you can talk about how do they work and how do they compare

Internals\\Selected-Lecs-1-2-1\\Lec3 on 1-2-1 interviews

Yes	Interviews	0.2115	16			
				1	SM	20/03/2018 16:47

is the ideas is trying to determine computer could perform certain functions you've got to break down an operation into a number of different steps that you can then turn into instructions for how to perform that operation. So that is the basis of my understanding – so it's the breaking down of a

Appendix 3.12 Ethical Approval from the University of Reading

University of Reading

Institute of Education

Ethical Approval Form A (version May 2015)

Tick one:

Staff project: _____ PhD ____ EdD __√_

Name of applicant (s): Mr Steve Mvalo

Title of project: Developing computer science students' computational thinking: The case for the use of simulation software

Name of supervisor (for student projects): Dr. N.V. Trakulphadetkrai

Please complete the form below including relevant sections overleaf.

	YES	NO
Have you prepared an Information Sheet for participants and/or their parents/carers that:		
a) explains the purpose(s) of the project	√	
b) explains how they have been selected as potential participants	√	
c) gives a full, fair and clear account of what will be asked of them and how the information that they provide will be used	√	
d) makes clear that participation in the project is voluntary	√	
e) explains the arrangements to allow participants to withdraw at any stage if they wish	√	

f) explains the arrangements to ensure the confidentiality of any material collected during the project, including secure arrangements for its storage, retention and disposal	√	
g) explains the arrangements for publishing the research results and, if confidentiality might be affected, for obtaining written consent for this	√	
h) explains the arrangements for providing participants with the research results if they wish to have them	√	
i) gives the name and designation of the member of staff with responsibility for the project together with contact details, including email . If any of the project investigators are students at the IoE, then this information must be included and their name provided	√	
k) explains, where applicable, the arrangements for expenses and other payments to be made to the participants	√	
j) includes a standard statement indicating the process of ethical review at the University undergone by the project, as follows: 'This project has been reviewed following the procedures of the University Research Ethics Committee and has been given a favourable ethical opinion for conduct'.	√	
k)includes a standard statement regarding insurance: "The University has the appropriate insurances in place. Full details are available on request".	√	
Please answer the following questions		
1) Will you provide participants involved in your research with all the information necessary to ensure that they are fully informed and not in any way deceived or misled as to the purpose(s) and nature of the research? (Please use the subheadings used in the example information sheets on blackboard to ensure this).	√	
2) Will you seek written or other formal consent from all participants, if they are able to provide it, in addition to (1)?	√	
3) Is there any risk that participants may experience physical or psychological distress in taking part in your research?		√
4) Have you taken the online training modules in data protection and information security (which can be found here: http://www.reading.ac.uk/internal/imps/Staffpages/imps-training.aspx)?	√	
5) Have you read the Health and Safety booklet (available on Blackboard) and completed a Risk Assessment Form to be included with this ethics application?	√	
6) Does your research comply with the University's Code of Good Practice in Research?	√-	

	YES	NO	N.A.
7) If your research is taking place in a school, have you prepared an information sheet and consent form to gain the permission in writing of the head teacher or other relevant supervisory professional?			√
8) Has the data collector obtained satisfactory DBS clearance?			√
9) If your research involves working with children under the age of 16 (or those whose special educational needs mean they are unable to give informed consent), have you prepared an information sheet and consent form for parents/carers to seek permission in writing, or to give parents/carers the opportunity to decline consent?			√
10) If your research involves processing sensitive personal data[1], or if it involves audio/video recordings, have you obtained the explicit consent of participants/parents?			√
11) If you are using a data processor to subcontract any part of your research, have you got a written contract with that contractor which (a) specifies that the contractor is required to act only on your instructions, and (b) provides for appropriate technical and organisational security measures to protect the data?			√
12a) Does your research involve data collection outside the UK?		√	
12b) If the answer to question 12a is "yes", does your research comply with the legal and ethical requirements for doing research in that country?			√
13a) Does your research involve collecting data in a language other than English?		√	
13b) If the answer to question 13a is "yes", please confirm that information sheets, consent forms, and research instruments, where appropriate, have been directly translated from the English versions submitted with this application.			√
14a. Does the proposed research involve children under the age of 5?		√	
14b. If the answer to question 14a is "yes": My Head of School (or authorised Head of Department) has given details of the proposed research to the University's insurance officer, and the research will not proceed until I have confirmation that insurance cover is in place.			√
If you have answered YES to Question 3, please complete Section B below			

[1] Sensitive personal data consists of information relating to the racial or ethnic origin of a data subject, their political opinions, religious beliefs, trade union membership, sexual life, physical or mental health or condition, or criminal offences or record.

Please complete **either** Section A **or** Section B and provide the details required in support of your application. Sign the form (Section C) then submit it with all relevant attachments (e.g. information sheets, consent forms, tests, questionnaires, interview schedules) to the Institute's Ethics Committee for consideration. Any missing information will result in the form being returned to you.

A: My research goes beyond the 'accepted custom and practice of teaching' but I consider that this project has **no** significant ethical implications. (Please tick the box.)	√

Please state the total number of participants that will be involved in the project and give a breakdown of how many there are in each category e.g. teachers, parents, pupils etc.

There are four stages of enquiry which are going to be conducted and have been provided with proposed number of participants as follows:

1. On-line survey
About 70 undergraduate students, 10 postgraduate students and 20 lecturers lasting for about 15-20 minutes.

After the on-line survey, three lecturers who teach on computers networks, who will have put down their email address to further participate in an in-depth interview will be selected. Students will be selected to ensure that the range of ability levels is included in the sample for in-depth interview and focus group

2. In-depth interviews
About 9 undergraduate students, 3 postgraduate students and 3 lecturers who teach on computer networks. These interviews will be audio recorded for about 20-30 minutes.

3. Focus groups interviews
Focus group will mainly be targeted at 8-10 undergraduate students and 6-8 postgraduate students. These focus groups will be audio recorded for about 30 minutes.

4. Problem-solving tasks and observations
Two problem-solving tasks covering the aspects of designing computer network on local area network, wider area networks and network security implementation will be given to 9 undergraduate and 3 postgraduate students. These tasks will be conducted in the second semester during their normal lab activities, for 4-6 sessions, each lasting for approximately an hour. Those who will not consent to participate in the study will still do the same tasks since they will be conducted within their normal lab activities but no

data will be collected from them. Three lecturers will observe and record students' activities on their problem-solving tasks.

Give a brief description of the aims and the methods (participants, instruments and procedures) of the project in up to 200 words noting:

Herewith is a working title of the project:

Developing computer science students' computational thinking: The case for the use of simulation software

This project intends to address the following research questions:

(i) *What are Computer Networks students' and lecturers' understanding of computational thinking (CT)?*
(ii) *What are students' and lecturers' perceptions of the use of simulation software to facilitate students' CT?*
(iii) *How might the use of simulation software facilitate students' CT?*

The following are methods in collecting relevant data in this study:

1. On-line survey

About 70 undergraduate students, 10 postgraduate students and 20 lecturers in computing department will be asked to participate in an on-line survey about their understanding and perceptions of computational thinking and how simulation software may facilitate students' computational thinking. They will be asked to provide contextual data about their thoughts, experiences and opinions on the use of simulation software to facilitate students' computational thinking when designing network systems. The survey will also have a few other questions about other aspects of computational thinking when designing computer systems. This survey will likely take around 15-20 minutes of their time.

2. In-depth interviews

Following the survey results, I will also choose 9 undergraduate students and 3 postgraduate students who will have indicated to further participate in an in-depth interview and three specific lecturers who teach on computer networks for an in-depth interview in order to get a deeper understanding and application of their computational thinking in designing network systems. Much of these questions will follow structured approach but the flow will be fluid so as to capture more of their understanding and experience. Subsequently, some questions may prompt them to explain more like "Can you explain what you mean by […] which I found interesting… These in-depth interviews are scheduled to last about 20-30 minutes;

3. Focus groups interviews

In-depth interviews will be followed by focus groups with 10-12 undergraduate and 6-8 postgraduate students respectively. Focus groups will last for about 30 minutes. Focus

groups will be selected based on their performance (higher, medium and lower). In-depth interviews will solicit student's individual perceptions and experiences of their computational thinking without the interferences of other students. Group interviews will provide consolidated information based on the tasks they did in class. I will coordinate the focus group so as to manage the debates making sure that the voice of every student is heard and recorded as appropriately.

4. Problem-solving tasks and observations

Two problem-solving tasks covering the aspects of designing computer network on local area network, wider area networks and network security implementation will be given to 9 undergraduate and 3 postgraduate students in order to assess their application of computational thinking on three main elements namely: *abstract, decomposition and generalisation*. These tasks will be conducted during their normal lab activities. Students will be selected based on their performance (higher, medium and lower). The task data will be collected by the use of students' reflective reports outlining the strategies they applied to work out problems in designing their networks, the results they produce and observation forms which will be filled by 3 lecturers teaching on computer network course. These lecturers will be briefed on key elements to observe and record.

Due to time sensitivity, I intend to start collecting data from the second semester (i.e. after January 2017).

B: I consider that this project **may** have ethical implications that should be brought before the Institute's Ethics Committee.	

Please state the total number of participants that will be involved in the project and give a breakdown of how many there are in each category e.g. teachers, parents, pupils etc.

Give a brief description of the aims and the methods (participants, instruments and procedures) of the project in up to 200 words.

1. title of project
2. purpose of project and its academic rationale
3. brief description of methods and measurements
4. participants: recruitment methods, number, age, gender, exclusion/inclusion criteria
5. consent and participant information arrangements, debriefing (attach forms where necessary)
6. a clear and concise statement of the ethical considerations raised by the project and how you intend to deal with then.
7. estimated start date and duration of project

C: SIGNATURE OF APPLICANT:

Note: a signature is required. Typed names are not acceptable.

I have declared all relevant information regarding my proposed project and confirm that ethical good practice will be followed within the project.

Signed: … Print Name…Steve Mvalo…… Date…16-11-16…….

STATEMENT OF ETHICAL APPROVAL FOR PROPOSALS SUBMITTED TO THE INSTITUTE ETHICS COMMITTEE

This project has been considered using agreed Institute procedures and is now approved.

Signed: Print Name…Xiao Lan Curdt-Christiansen Date…17-11-16

(IoE Research Ethics Committee representative)*

* A decision to allow a project to proceed is not an expert assessment of its content or of the possible risks involved in the investigation, nor does it detract in any way from the ultimate responsibility which students/investigators must themselves have for these matters. Approval is granted on the basis of the information declared by the applicant.

Appendix 3.13 Consent Form to the Head of Computing Department

University of Reading

INFORMATION SHEET FOR Head of Computing Department

Research Project:	Developing computer science students' computational thinking: The case for the use of simulation software
Researcher:	Mr. Steve Mvalo (s.h.e.mvalo@pgr.reading.ac.uk)
Supervisor:	Dr. N.V. Trakulphadetkrai (n.trakulphadetkrai@reading.ac.uk)

I would like to invite your department to take part in a study of developing computer science students' computational thinking: The case for the use of simulation software.

What is the study?
The study is part of doctorate degree in education that I am undertaking at the University of Reading. The study aims to explore students' and lecturers' understanding on computational thinking. The study further explore how simulation software may facilitate students' computational thinking. The study will involve students designing computer network systems on simulation software with the aim of investigating their computational thinking. Lecturers and students will be interviewed to investigate their perceptions of the use of simulation software to facilitate students' computational thinking.

Why has my department been chosen to take part?
Your department has been invited to take part in the project because your department specialises in computation and also that is where I have an access to students who are pursing computing courses. Above all, I and my fellow lecturers are involved in teaching computing to students. Lecturers, particularly those who use simulation software when teaching students how to design computer networks, will be invited to participate in this investigation. Both lecturers and students who will put down their email address at the end on the on-line survey may be requested for an in-depth interview. Students will further be selected to ensure that the range of ability is included in the sample for in-depth interview and focus group.

Does the department have to take part?
It is entirely up to you whether you wish to participate. You may also withdraw your consent to participation at any time during the project, without any repercussions to you, by contacting me directly using the contact details above.

What will happen if the department takes part?
With your agreement, I would like to invite some of your students and lecturers to take part in the following activities:

1. On-line survey
About 70 undergraduate students, 10 postgraduate students and 20 lecturers in your department will be asked to participate in an on-line survey about their understanding and perceptions of computational thinking and how simulation software may facilitate students' computational thinking. They will be asked to provide contextual data about their thoughts, experiences and opinions on the use of simulation software to facilitate students' computational thinking when designing network systems. The survey will also have a few other questions about other aspects of computational thinking when designing computer systems. This survey will likely take around 15-20 minutes of their time.

2. Audio-recorded In-depth interviews and focus groups interviews
Following the survey results, I will also choose 9 undergraduate students and 3 postgraduate students who will have indicated to further participate in an in-depth interview and three specific lecturers who teach on computer networks for an in-depth interview in order to get a deeper understanding and application of their computational thinking in designing network systems. Much of these questions will follow structured approach but the flow will be fluid so as to capture more of their understanding and experience. Subsequently, some questions may prompt them to explain more like "Can you explain what you mean by […] which I found interesting… These in-depth interviews are scheduled to last about 20-30 minutes; followed by focus groups with 10-12 undergraduate and 6-8 postgraduate students respectively. Focus groups will last for about 30 minutes. Focus groups will be selected based on their performance (higher, medium and lower). In-depth interviews will solicit student's individual perceptions and experiences of their computational thinking without the interferences of other students. Group interviews will provide consolidated information based on the tasks they did in class. I will coordinate the focus group so as to manage the debates making sure that the voice of every student is heard and recorded as appropriately.

3. Problem-solving tasks, observations and reflective reports
Two problem-solving tasks covering the aspects of designing computer network on local area networks, wider area networks and network security implementation will be given to 9 undergraduate and 3 postgraduate students in order to assess their application of computational thinking on three main elements namely: abstract, decomposition and generalisation. These tasks will be conducted in the second semester during their normal lab activities, for 4-6 sessions, each lasting for approximately an hour. Students will be selected based on their performance (higher, medium and lower). Those who will not consent to participate in the study will still do the same tasks since they will be conducted within their normal lab activities, but no data will be collected from them. The task data will be collected by the use of students' reflective reports outlining the strategies they applied to work out problems in designing their networks, the results they produce and observation forms which will be filled by 3 lecturers teaching on computer network course. These lecturers will be briefed on key elements to observe and record.

What are the risks and benefits of taking part?
The information given by participants will remain confidential and will only be seen by me and my supervisor. Neither the participants nor the department will be identifiable in my report resulting from the study.

I anticipate that the findings of the study will be useful for lecturers in developing assessments which develop and facilitate students' computational thinking when designing computer network systems.

What will happen to the data?
Any data collected will be held in strict confidence and no real names will be used in this study. The records of this study will be kept private. No identifiers linking the participants or the school to the study will be included in any sort of report. Participants will be assigned a pseudonym and will be referred to by that pseudonym in all records. Research records will be stored securely in a locked filing cabinet and on a password-protected computer and only the research team will have access to the records. The data will be destroyed securely once the findings of the study are written up, after five years. If you would like a brief summary of the findings, you can e-mail me directly using the contact details above.

The results of the study may be disseminated at educational conferences and form part of an article submitted to one or more educational journals. A draft copy of any paper or report can be sent to you electronically in advance of submission for publication on request.

What happens if I change my mind?
You can change your mind at any time without any repercussions. If you change your mind after data collection has ended, we will discard the department's data.

Who has reviewed the study?
This project has been reviewed following the procedures of the University Research Ethics Committee and has been given a favourable ethical opinion for conduct. The University has the appropriate insurances in place. Full details are available on request.

What happens if something goes wrong?

In the unlikely case of concern or complaint, you can contact my supervisor, Dr. N.V. Trakulphadetkrai, at the University of Reading's Institute of Education by phone on 0118 378 2665 or by e-mail on n.trakulphadetkrai@reading.ac.uk

Where can I get more information?
If you would like more information, please contact me via telephone on 07895979279 or by e-mail on s.h.e.mvalo@pgr.reading.ac.uk

What do I do next?
I do hope that you will agree to your participation in the study. If you do, please complete the attached consent form and return it by e-mailing it back to s.h.e.mvalo@pgr.reading.ac.uk

Yours sincerely,

Mr Steve Mvalo

CONSENT FORM FOR Head of Computing Department

Research Project:	Developing computer science students' computational thinking: The case for the use of simulation software
Researcher:	Mr Steve Mvalo (s.h.e.mvalo@pgr.reading.ac.uk)
Supervisor:	Dr. N.V. Trakulphadetkrai (n.trakulphadetkrai@reading.ac.uk)

I have read the Information Sheet about the project and received a copy of it.

I understand what the purpose of the project is and what is required of me and members of my Department. All my questions have been answered.

Please tick ('√') as appropriate:

	Yes	No
I consent to the involvement of my Department in the project as outlined in the Information Sheet		

Name of **Head of Computing Department**	
Name of Department:	
Signed:	
Date:	

THANK YOU VERY MUCH FOR YOUR KIND ASSISTANCE!

Appendix 3.14 Consent Form to the Participants

University of Reading

INFORMATION SHEET FOR STUDENTS

Research Project:	Developing computer science students' computational thinking: The case for the use of simulation software
Researcher:	Mr. Steve Mvalo (s.h.e.mvalo@pgr.reading.ac.uk)
Supervisor:	Dr. N.V. Trakulphadetkrai (n.trakulphadetkrai@reading.ac.uk)

I would like to invite you to take part in a study of developing computer science students' computational thinking: The case for the use of simulation software.

What is the study?
The study is part of doctorate degree in education that I am undertaking at the University of Reading. The study aims to explore students' and lecturers' understanding on computational thinking. The study further explore how simulation software may facilitate students' computational thinking. The study will involve students designing computer network systems on simulation software with the aim of investigating their computational thinking. Lecturers and students will be interviewed to investigate their perceptions of the use of simulation software to facilitate students' computational thinking.

Why have you been chosen to take part?
You have been chosen to take part in the project because you have been identified as one of the students in computing department studying for computer networks course. You have also been selected to ensure that the range of ability levels is included in the sample.

Do you have to take part?
It is entirely up to you whether you wish to participate. You may also withdraw your consent to participation at any time during the project, without any repercussions to you, by contacting me directly using the contact details above.

What will happen if you take part?
With your consent, you will participate in an *on-line survey* which I am trying to investigate your understanding, experience and perceptions of computational thinking when designing computer networks on simulation software lasting for approximately 20 minutes. If you put down your email address at the end of the online-survey, you may also be requested to participate in an *in-depth interview* lasting for about 20-30 minutes and a *focus group* of between 6-10 participants lasting for about 30 minutes. Finally, you may also be requested to participate in two problem-solving tasks covering the aspects of designing computer network on local area network, wider area networks and network security implementation. These tasks will be conducted in the second semester during your normal lab activities for 4-6 sessions, each lasting for approximately an hour. Those who will not consent to participate in the study will still do the same tasks since they will be conducted within your normal lab activities, but no data will be collected from them. Therefore, you may be involved in four enquiries namely: *online survey, in-depth interviews, focus group and problem-solving tasks*. With your permission, the interview and focus group will be audio-recorded and transcribed.

Much of the in-depth and group questions will follow structured approach but the flow will be fluid so as to capture more of your understanding and experience. Subsequently, some questions may prompt you to explain more like "Can you explain what you mean by […] which I found interesting… In-depth interviews will solicit your individual perceptions and experiences of computational thinking without the interferences of other students. Group interviews will provide consolidated information based on the problem-solving tasks you did in class.

What are the risks and benefits of taking part?
The information given by participants will remain confidential and will only be seen by me and my supervisor. Neither the participants nor the department will be identifiable in my report resulting from the study.

I anticipate that the findings of the study will be useful for lecturers in developing assessments which develop and facilitate students' computational thinking when designing computer network systems.

What will happen to the data?
Any data collected will be held in strict confidence and no real names will be used in this study. The records of this study will be kept private. No identifiers linking the participants or the department to the study will be included in any sort of report. Participants will be assigned a pseudonym and will be referred to by that pseudonym in all records. Research records will be stored securely in a locked filing cabinet and on a password-protected computer and only the research team will have access to the records. The data will be destroyed securely once the findings of the study are written up, after five years. If you would like a brief summary of the findings, you can e-mail me directly using the contact details above.

The results of the study may be disseminated at educational conferences and form part of an article submitted to one or more educational journals. A draft copy of any paper or report can be sent to you electronically in advance of submission for publication on request.

What happens if I change my mind?
You can change your mind at any time without any repercussions. If you change your mind after data collection has ended, we will discard your data.

Who has reviewed the study?
This project has been reviewed following the procedures of the University Research Ethics Committee and has been given a favourable ethical opinion for conduct. The University has the appropriate insurances in place. Full details are available on request.

What happens if something goes wrong?
In the unlikely case of concern or complaint, you can contact my supervisor, Dr. N.V. Trakulphadetkrai, at the University of Reading's Institute of Education by phone on *0118 378 2665* or by email on n.trakulphadetkrai@reading.ac.uk

Where can I get more information?
If you would like more information, please contact me via telephone on 07895979279 or by email on s.h.e.mvalo@pgr.reading.ac.uk

What do I do next?
I do hope that you will agree to your participation in the study. If you do, please complete the attached consent form and return it by e-mailing it back to s.h.e.mvalo@pgr.reading.ac.uk

Thank you for your time.

Yours sincerely,

Mr Steve Mvalo

CONSENT FORM FOR STUDENTS

Research Project:	Developing computer science students' computational thinking: The case for the use of simulation software
Researcher:	Mr Steve Mvalo (s.h.e.mvalo@pgr.reading.ac.uk)
Supervisor:	Dr. N.V. Trakulphadetkrai (n.trakulphadetkrai@reading.ac.uk)

I have read the Information Sheet about the project and received a copy of it.

I understand what the purpose of the project is and what is required of me. All my questions have been answered.

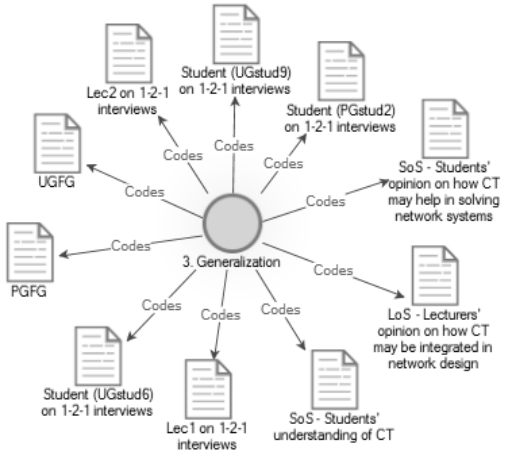
Please tick ('√') as appropriate:

	Yes	No
I consent to be involved in the project as outlined in the Information Sheet		

Name of student:	
Signed:	
Date:	

THANK YOU VERY MUCH FOR YOUR KIND ASSISTANCE!

Appendix 4.1a Source Codes for Predetermined Themes for RQ1

<table>
<tr><th colspan="3">Sources of Codes for Predetermined Themes (abstraction, decomposition & generalisation)</th></tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>SoS (2)
• about CT = 3
• how CT help = 10

<table>
<tr><td>Stud 1-2-1 (3)
• PGstud2 = 2
• PGstud1 = 2
• UGstud9 = 5</td><td>Stud 1-2-1 (3)
• PGstud2 = 2
• PGstud1 = 2
• UGstud9 = 5</td><td>LoS (3)
• about CT = 3
• how CT may help = 2
• how CT integrated = 3</td></tr>
</table>
Lec 1-2-1 (2)
• Lec1 = 3
• Lec3 = 5
UGFG = 2

Total codes = 40 from 11 coding occurences</td>
<td>SoS (2)
• about CT = 15
• how CT help = 15

<table>
<tr><td>Stud 1-2-1 (5)
• UGstud6 = 7
• UGstud9 = 2
• PGstud1 = 3
• PGstud2 = 2
• PGstud6 = 3</td><td>LoS (3)
• about CT = 3
• how CT may help = 5
• how CT integrated = 2</td><td>Lec 1-2-1 (3)
• Lec1 = 5
• lec2 = 4
• Lec3 = 8</td></tr>
</table>
UGFG = 3
PGFG = 7

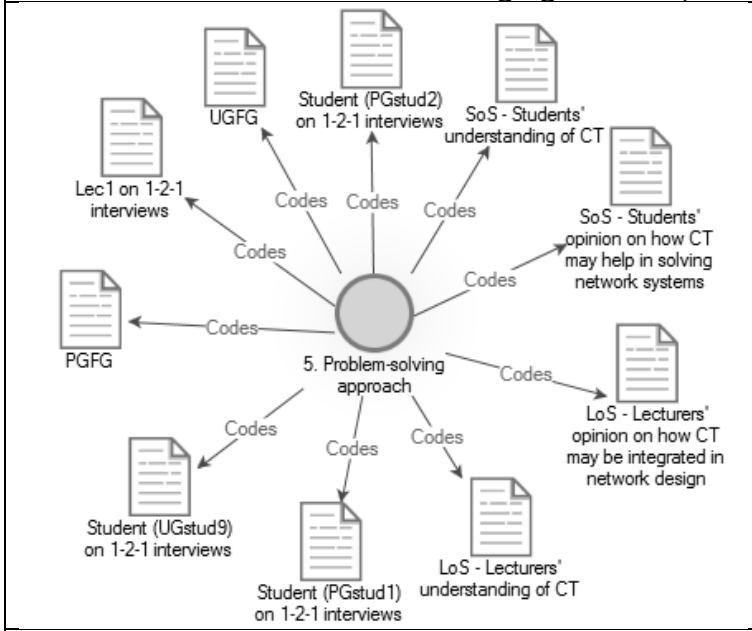
Total Codes = 84 from 15 coding occurences</td>
<td>SoS (2)
• about CT = 1
• how CT may help = 3

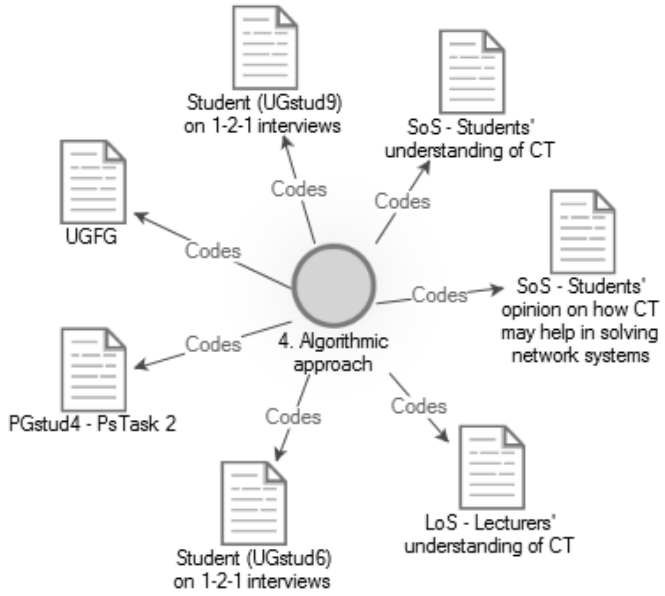
<table>
<tr><td>Stud 1-2-1 (3)
• UGstud9 = 2
• UGstud6 = 3
• PGstud2 = 2</td><td>LoS (1)
• how CT integrated = 1</td><td>Lec 1-2-1 (2)
• Lec1 = 1
• Lec2 = 1</td></tr>
</table>
UGFG = 5
PGFG = 6

Total Codes = 26 from 10 coding occurences</td>
</tr>
</table>

Appendix 4.1b Source Codes for Emerged Themes for RQ1

Sources of Codes for Emerging Themes *(Problem-solving approaches and Algorithmic approaches)*





SoS (2)

- about CT = 39
- how CT may help = 13 (23)

Stud 1-2-1 (3)	LoS (2)	Lec 1-2-1 (1)
• UGstud9 = 1 • PGstud1 = 1 • PGstud2 = 8	• about CT = 6 • how CT integrated = 1	• Lec1 = 3

UGFG = 8
PGFG = 2

Total codes = 92 from 15 coding occurences 10

SoS (2)

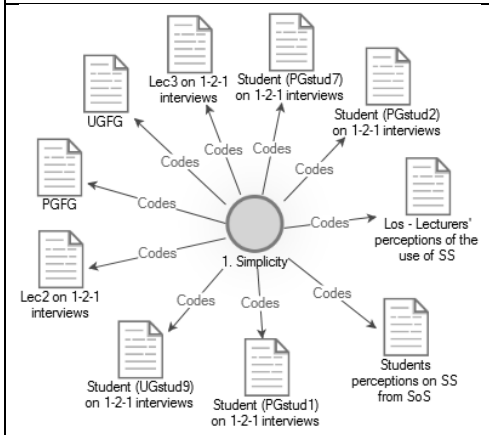
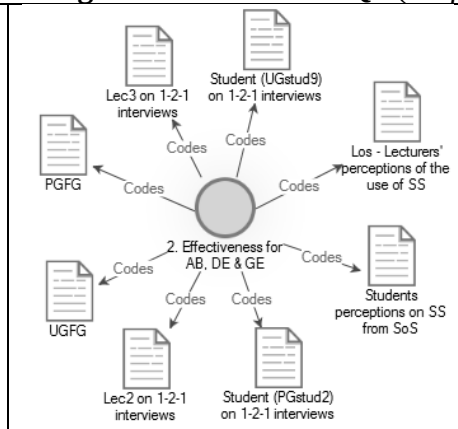
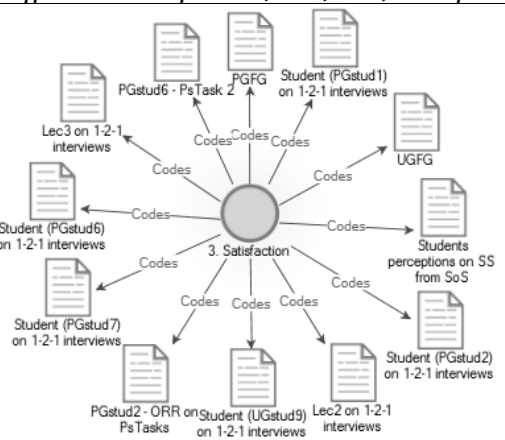
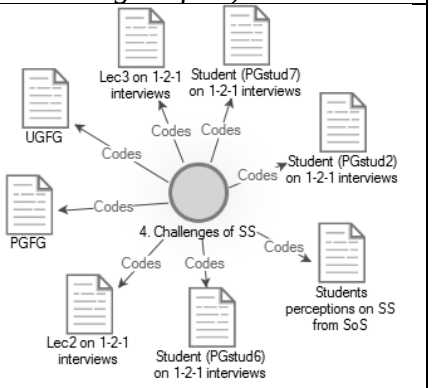
- about CT = 1
- how CT may help = 2

Stud 1-2-1 (2)	Los (1)	PsTask2 (1)
• UGstud6 = 1 • UGstud9 = 2	• about CT = 2	• PGstud4 = 1

UGFG = 3

Total codes = 12 from 7 coding occurences

Appendix 4.2 Source Codes for RQ2

<table>
<tr><th colspan="4">Sources of Codes from Emerged Themes from RQ2 (simplicity, effectiveness for AB, DE, GE, satisfaction and challenges of SS)</th></tr>
<tr>
<td> 
</td>
<td> 
</td>
<td> 
</td>
<td> 
</td>
</tr>
<tr>
<td>SoS – perception of SS = 24

Stud 1-2-1 (9)
PGstud1 = 3
PGstud2 = 3
PGstud7 = 1
UGstud7 = 2
<table>
<tr><td>Los – perception of SS = 6</td><td>Lec 1-2-1 (9)
Lec2 = 6
Lec3 = 3</td></tr>
</table>
UGFG = 1
PGFG = 3

Total codes = 52 from 10 coding occurences</td>
<td>SoS – perception of SS = 29

Stud 1-2-1 (2)
PGstud2 = 1
UGstud9 = 1
<table>
<tr><td>LoS – perception of SS = 7</td><td>Lec 1-2-1 (6)
Lec2 = 4
Lec3 = 2</td></tr>
</table>
UGFG = 3
PGFG = 2

Total codes = 49 from 8 coding occurences</td>
<td>SoS – perception of SS = 19

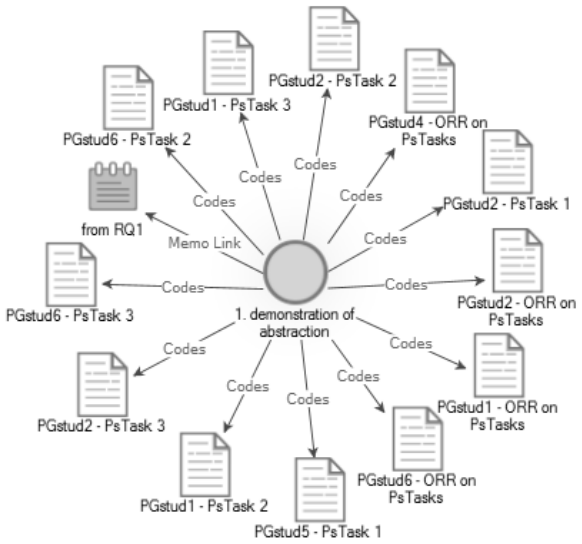
Stud 1-2-1 (20)
PGstud1 = 1
PGstud2 = 3
PGstud6 = 3
PGstud7 = 2
PGstud9 = 11
<table>
<tr><td>Lec 1-2-1 (11)
Lec2 = 8
Lec3 = 3</td><td>ORR on PsTask2 (1)
PGstud2 = 1</td><td>PsTask2 (1)
PGstud6 = 2</td></tr>
</table>
UGFG = 10
PGFG = 16

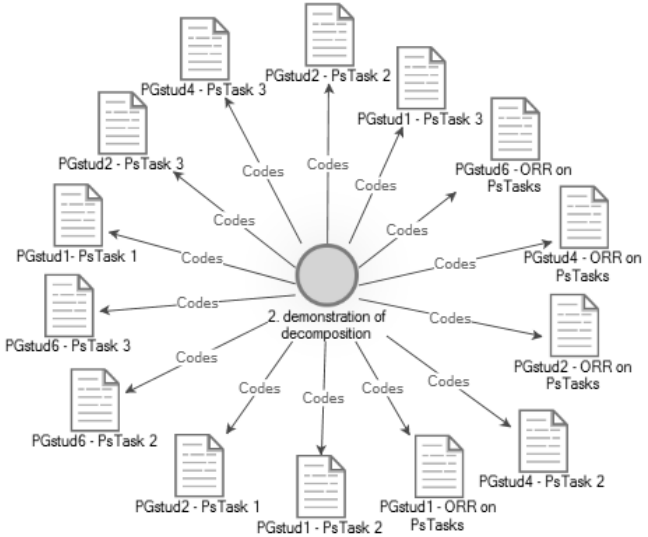
Total codes = 79 from 12 coding occurences</td>
<td>SoS – perception of SS = 11
not sure (5/11)
explained challenges (6/11)
<table>
<tr><td>Stud 1-2-1 (6)
PGstud2 = 3
PGstud6 = 1
PGstud7 = 2</td><td>Lec 1-2-1 (7)
Lec2 = 3
Lec3 = 4</td></tr>
</table>
UGFG = 2
PGFG = 1

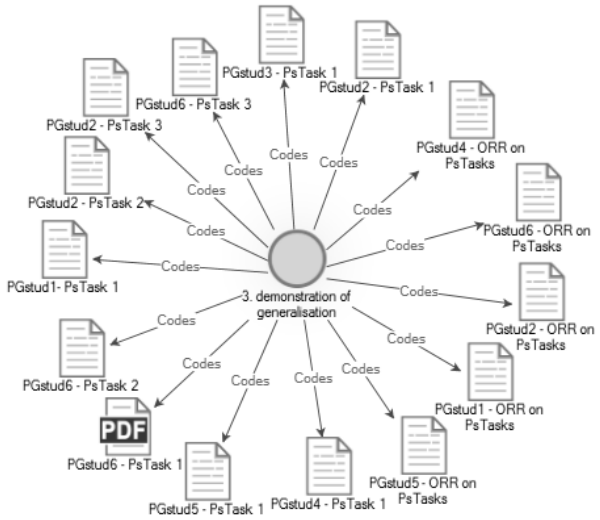
Total codes = 27 from 8 coding occurences</td>
</tr>
</table>

Appendix 4.3 Source Codes for RQ3

Sources of Codes for Demonstrating Concepts of CT (*abstraction, decomposition & generalisation*)







PsTask1(2)	PsTask2 (3)
• PGstud5 = 1 • PGstud2 = 1	• PGstud2 = 3 • PGstud6 = 1 • PGstud1 = 3

PsTask3 (3)	ORR on PsTasks (4)
• PGstud1 = 2 • PGstud2 = 1 • PGstud6 = 1	• PGstud6 = 2 • PGstud1 = 2 • PGstud2 = 3 • PGstud4 = 1

Total codes = 21 from 12 coding occurences

PsTask1(2)	PsTask2 (4)
• PGstud1 = 1 • PGstud2 = 2	• PGstud1 = 8 • PGstud4 = 1 • PGstud2 = 5 • PGstud6 = 1

PsTask3 (4)	ORR on PsTasks (4)
• PGstud1 = 2 • PGstud4 = 1 • PGstud2 = 1 • PGstud6 = 1	• PGstud1 = 2 • PGstud2 = 2 • PGstud4 = 1 • PGstud6 = 1

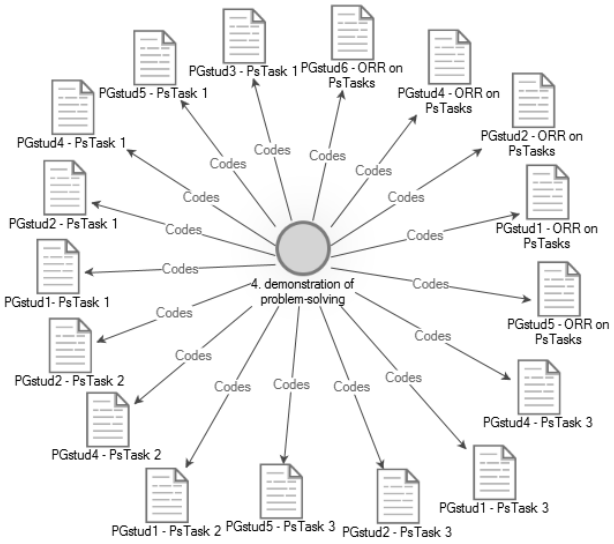
Total Codes = 29 from 14 coding occurences

PsTask1(6)	PsTask2 (2)
• PGstud1 = 3 • PGstud2 = 9 • PGstud3 = 3 • PGstud4 = 4 • PGstud5 = 3 • PGstud6 = 2	• PGstud2 = 1 • PGstud6 = 2

PsTask3 (2)	ORR on PsTasks (5)
• PGstud2 = 1 • PGstud6 = 2	• PGstud1 = 1 • PGstud2 = 5 • PGstud4 = 1 • PGstud5 = 3 • PGstud6 = 1

Total codes = 41 from 15 coding occurences

(*Problem-solving involving AB, DE, GE*)



PsTask1(5)	PsTask2 (3)
• PGstud1 = 8 • PGstud2 = 6 • PGstud3 = 3 • PGstud4 = 5 • PGstud5 = 2	• PGstud1 = 2 • PGstud2 = 4 • PGstud4 = 3

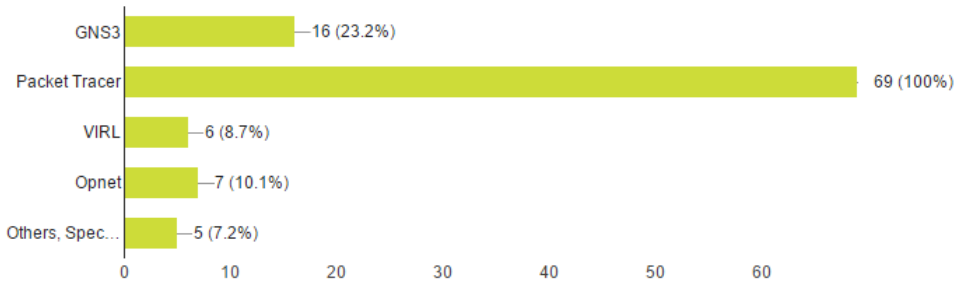
PsTask3 (4)	ORR on PsTasks (4)
• PGstud1 = 1 • PGstud2 = 1 • PGstud4 = 1 • PGstud5 = 1	• PGstud1 = 2 • PGstud2 = 3 • PGstud4 = 1 • PGstud5 = 1 • PGstud6 = 1

Total Codes = 45 from 17 coding occurences

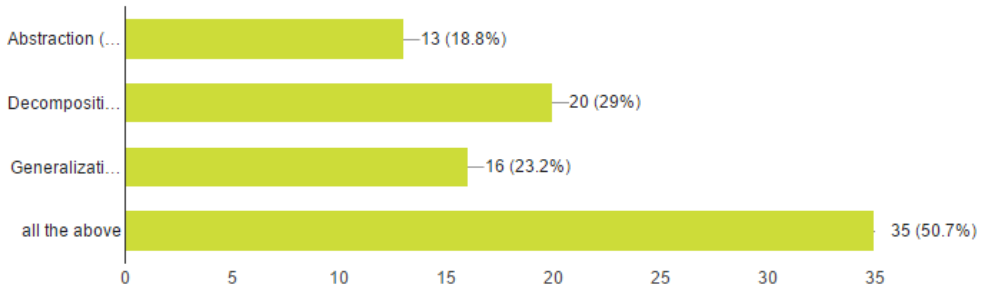
Appendix 4.4 Some Responses from SoS

Perception of simulation software to facilitate students computational thinking

15. Which of the following simulation softwares are you familiar with?
(69 responses)



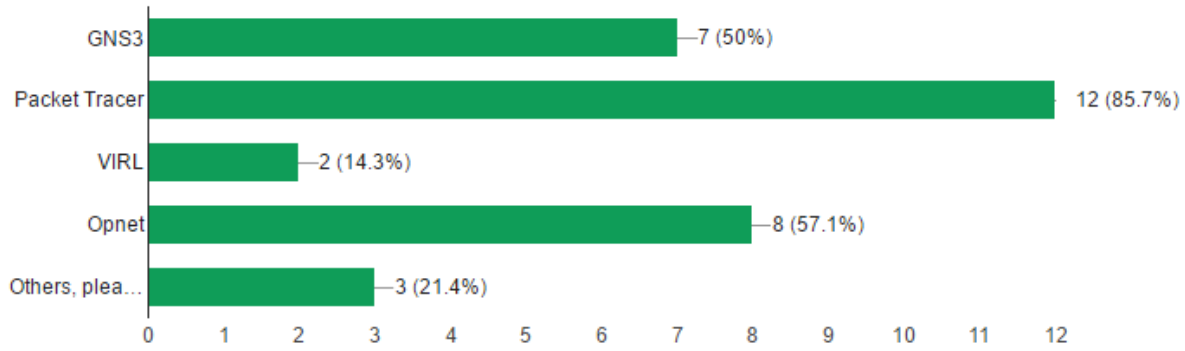
16. Which of the following elements of computational thinking can be demonstrated by the use of simulation software you have selected on question 15
(69 responses)



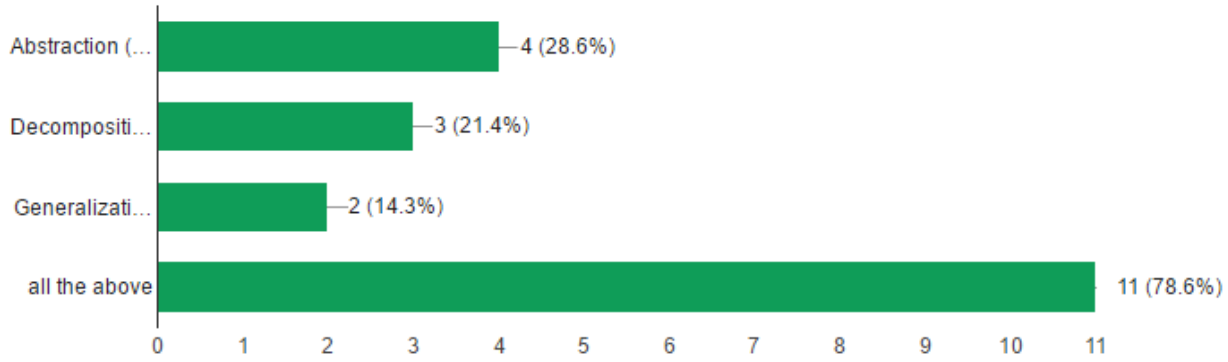
Appendix 4.5 Some Responses from LoS

Perception of simulation software to facilitate students computational thinking

17. Which of the following simulation softwares are you familiar with?
(14 responses)

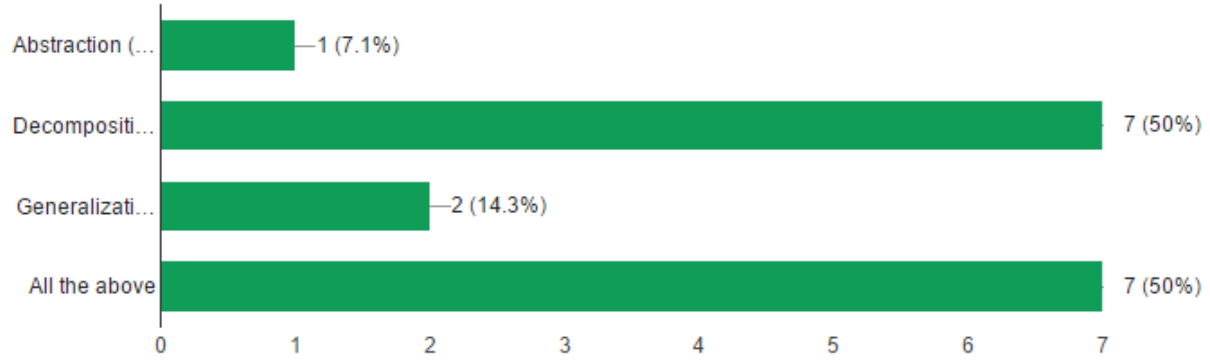


18. Which of the following elements of computational thinking can be demonstrated by the use of simulation software you have selected on question 17
(14 responses)



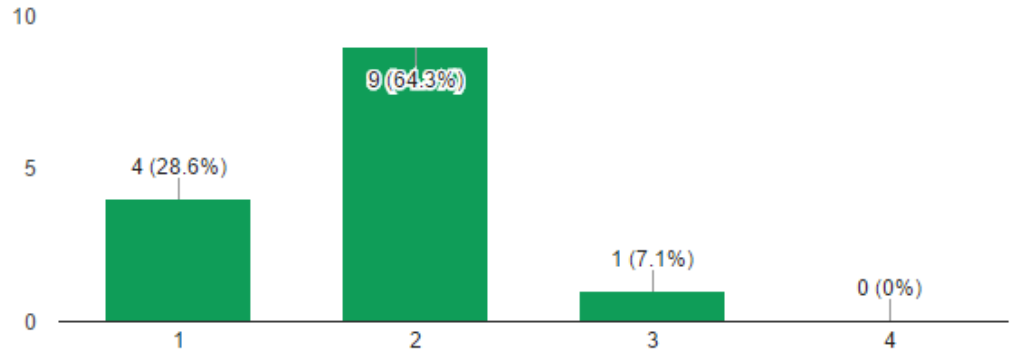
20. When designing a piece of assessment, which elements do you like your students to demonstrate when using simulation software?

(14 responses)



23. Simulation software provides a good platform to troubleshoot network design and therefore develops students' computational thinking

(14 responses)



Appendix 4.6 Sample from Student's Reflective Report

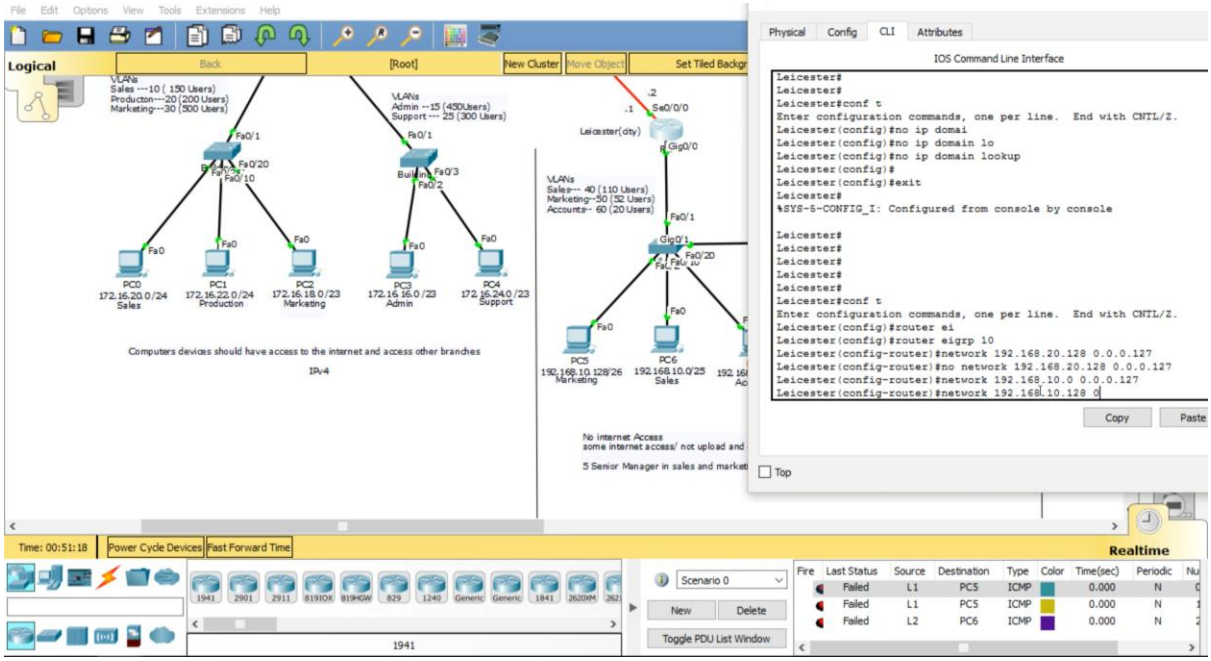
	Reflective Report
Reflective report with recommendations. This should include: v. challenges encountered vi. how your computational thinking skills may have helped in tackling tasks e.g. design considerations (i.e. making abstract concrete) troubleshooting techniques (i.e. decomposition and generalisation) applied vii. areas to improve and add in the network infrastructure with justification viii. recommendation (i.e. security, utilization, performance, training, maintenance & management)	Once again, I am delighted for this huge opportunity given to me, which is clearly a crucial point assessment for my academic curriculum. According to the relevant work I must have a clear explanation about the all the process which I have been going through from the challenges faced during the design as well as the implementation of this complex Wide Area Network on packet tracer. Talking about challenges will be decisive because it wasn't easy for me to establish connectivity and convergence into the whole network following the requirements, I had to think beyond the written requirements to elaborate a successful technical plan for this network, I faced issues in implementing connectivity between IPv6 Users located between SHEFFIELD and MANCHESTER Branch even though the DHCPv6 ,EIRGPv6 was well implemented. Applying ACLs and NAT was also a big issue when restricting internet access to some other users, and deploying the NAT with the ACL really gave a headache, therefore I finally decided to go beyond the issues and implement first the less complex solutions allowing full convergence and user reachability from all the branches through the headquarters that contains the Loopback (simulated as ISP). The way of using computational thinking was accurate because it gives you an ability to evaluate each step as long as you progress with your work from assigning IP address on Interfaces, creating virtual circuit Frame-relay on sub interfaces, Assigning VLANs, security routine on specified branch such as LEICESTER. My thought process was completely involved into finding the right method, establishing a structured work enabling the all process to be understood as one piece of work therefore develop a solution to solve all critical issues in the network.

	Basically, these steps bellow helped me to successfully achieve the deployment and implementation from the abstract to the concrete • Developing a simple understanding on how to design a complex network on simulation software (Packet tracer); • Preparing different equipment needed to be implemented onto the network; (Choosing Routers, Switches, PCs, Laptops etc.) • Establishing a physical connectivity to match the network requirements (Specifying serial interface cables, gigabit Ethernet cables, frame-relay (Pvc), etc. • Configuring IP addressing, implementing DHCP server, Wireless AP, VLANs, Frame-relay sub-interfaces (Main branches), IPv6 Configuration and DHCPv6, Routing protocols, NAT and ACLs. • Implementing, a computational thinking approach to identify diverse ways, and finding solutions to the relevant problem. • Breaking the major problem into small pieces (structuring the network elements: routers, Switches, Access Point, Frame-Relay, where should we implement ACLs and NAT requirements, based on the given policies), • 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 LEICSETER and DERBY we had a same requirement, ACLs on VLANs); • Decomposing complex breakdowns (Troubleshooting issues for Wireless, IPv6 and ACLs nearly became a nightmare, then later I understood that the network must convergence first approximatively 60-120 second after opening Packet Tracer, then the hosts connected to the access Point (AP) was pinging the Loopback, other VLANs and Branches except the IPv6 users who was almost isolated from the network deployed on IPv4. • Step-by-step approach in solving issues (Being the last point of checking all configurations, this method is a strong as possible allowing you to solve each issue not all in once but one after another, writing down the mistakes, evaluating the assigned IP- addresses, DHCPv4 implemented, internal routing protocols such as EIGRP, moreover that I had to demonstrate the

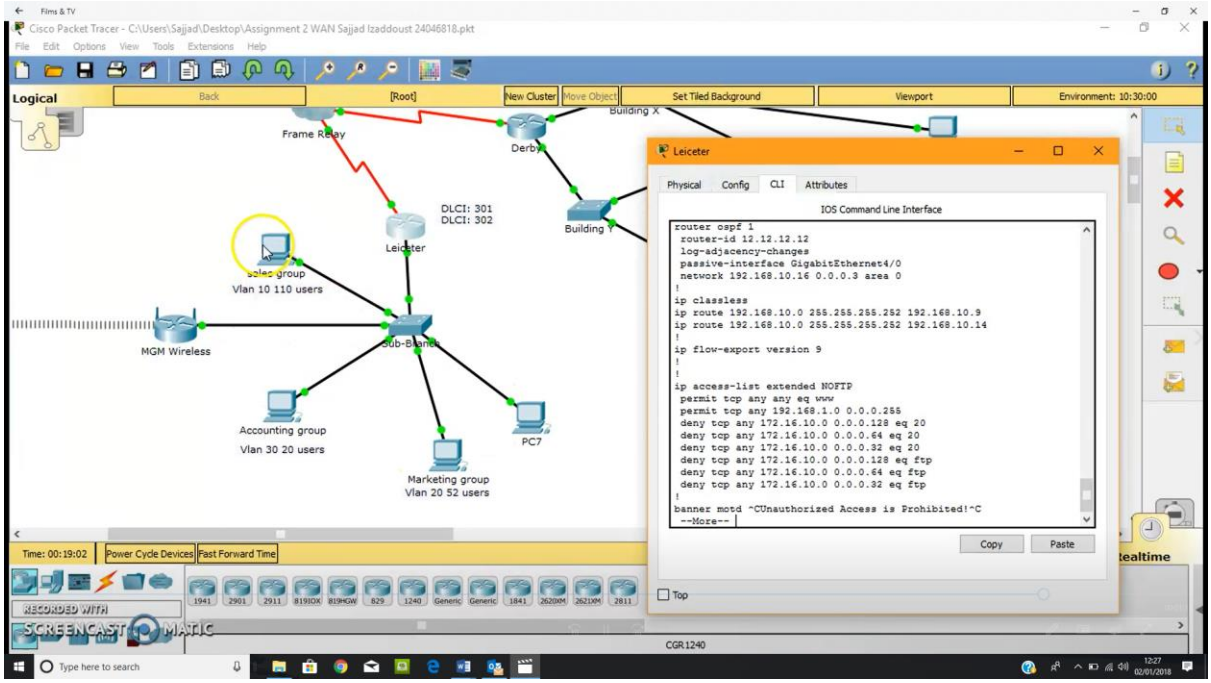
	technical aspects of each branches through their performances while communicating in the WAN, however this steps much involve into issuing commands such as **:** • **# show frame-relay pvc, # show ip eigrp neighbors**, **# sh access-lists, # sh ip nat translations, # debug ppp authentication # debug ppp negotiation, #show vlan brief, #sh ip dhcp binding, #show ip route, #show running-config etc.** **Using these commands was efficient to troubleshoot the network issues encountered while implementing the relevant requirements.** This piece of assessment helped me understand the process involve for Planning, Designing and Implementing network requirements from a written policy and turn it into a physical and acceptable Network Design capable of replicate the enterprise policies, as well as in the way of thinking how shall I start configuring such a complex network? Is there any unavailable devices, Cables, or other equipment? getting use to the Computational thinking which is probably the main point giving me intuition and stability in solving issues like this I gained ability in looking thoroughly into the routing tables of every devices, collect information and troubleshoot issues as quickly as possible. This work brought to me inventiveness and innovation in developing security policies, managing users in the network, preparing and applying filtering rules with access list, enabling connectivity with the ISP, segmenting network into different VLANs, also assigning appropriate IP address according to the requirements, therefore the training obtain doing this work has considerably increased my ability to not only design or implement but moreover finding issues and troubleshooting problems.

Appendix 4.7 Students' Sample video clips

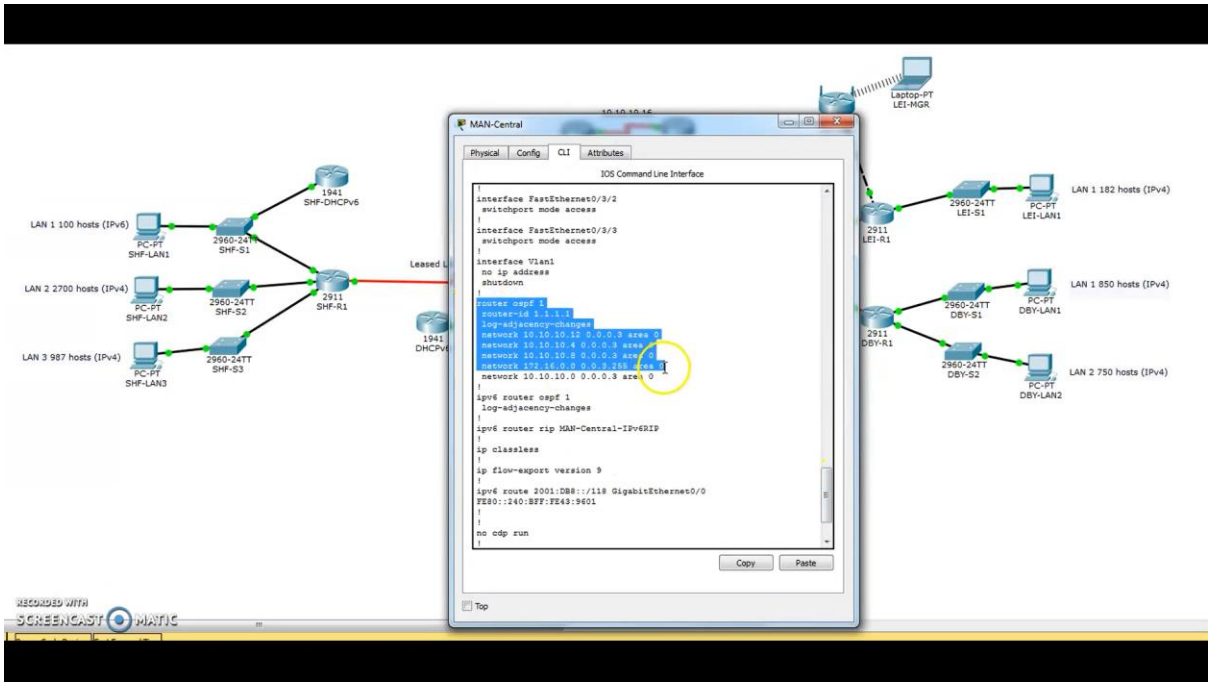
Design A



Design B

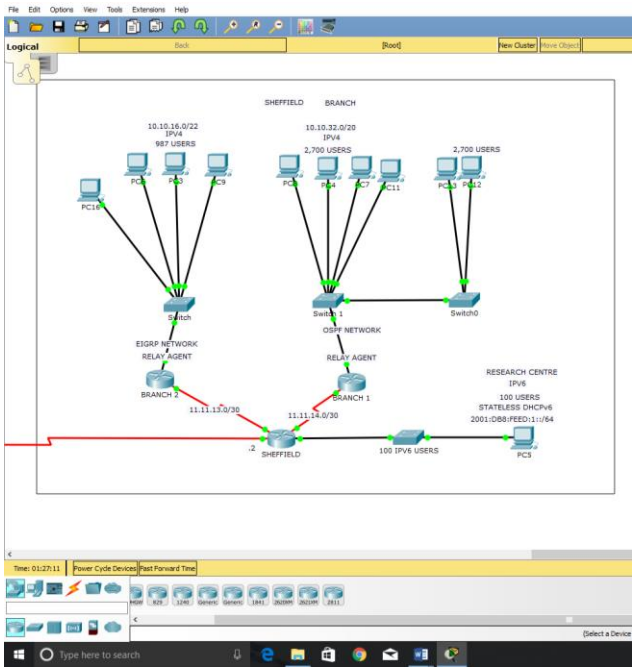


Design C

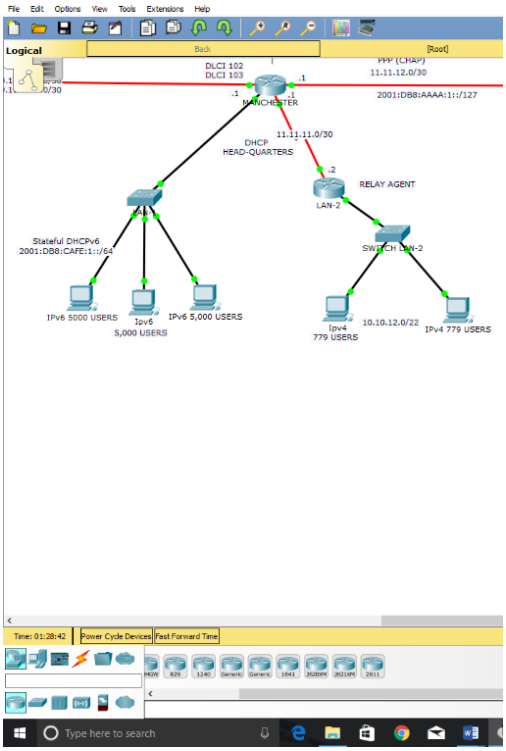

MAN-Central
Physical
Config
CLI
Attributes
IOS Command Line Interface
interface FastEthernet0/3/2
 switchport mode access
!
interface FastEthernet0/3/3
 switchport mode access
!
interface Vlan1
 no ip address
 shutdown
!
router ospf 1
 router-id 1.1.1.1
 log-adjacency-changes
 network 10.10.10.12 0.0.0.3 area 0
 network 10.10.10.4 0.0.0.3 area 0
 network 10.10.10.8 0.0.0.3 area 0
 network 172.16.0.0 0.0.3.255 area 0
 network 10.10.10.0 0.0.0.3 area 0
!
ipv6 router ospf 1
 log-adjacency-changes
!
ipv6 router rip MAN-Central-IPv6RIP
!
ip classless
!
ip flow-export version 9
!
ipv6 route 2001:DB8::/118 GigabitEthernet0/0
FE80::240:BFF:FE43:9601
!
!
no cdp run
!
Copy
Paste
Top
Laptop-PT
LEI-MGR
2911
LEI-R1
2960-24TT
LEI-S1
PC-PT
LEI-LAN1
LAN 1 182 hosts (IPv4)
2911
DBY-R1
2960-24TT
DBY-S1
PC-PT
DBY-LAN1
LAN 1 850 hosts (IPv4)
2960-24TT
DBY-S2
PC-PT
DBY-LAN2
LAN 2 750 hosts (IPv4)
1941
SHF-DHCPv6
LAN 1 100 hosts (IPv6)
PC-PT
SHF-LAN1
2960-24TT
SHF-S1
LAN 2 2700 hosts (IPv4)
PC-PT
SHF-LAN2
2960-24TT
SHF-S2
2911
SHF-R1
LAN 3 987 hosts (IPv4)
PC-PT
SHF-LAN3
2960-24TT
SHF-S3
Leased L
1941
DHCPv6
RECORDED WITH
SCREENCAST MATIC

Appendix 4.8 Student's Sample Simulated Network Designs Demonstrating the Application of the Concepts of Decomposition

Task A

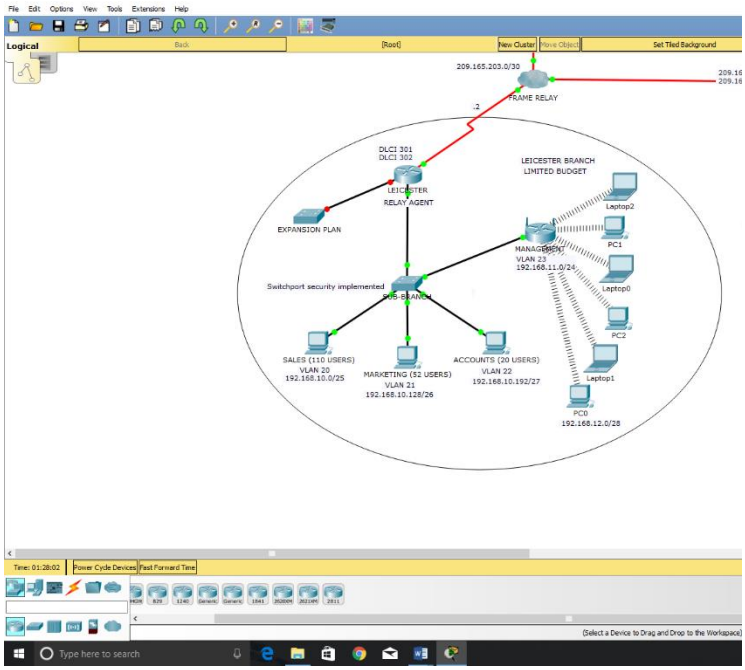


Task B



Task C

Final Task that shows the link of all the tasks above



Final Task that shows the link of all the tasks above