Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Software Engineering

USING VIRTUAL MACHINES TO CREATE STANDARDIZED LEARNING ENVIRONMENT FOR SOFTWARE ENGINEERING COURSES

Examiners: Associate Professor Jussi Kasurinen
Associate Professor Uolevi Nikula

TIIVISTELMÄ

Lappeenrannan-Lahden teknillinen yliopisto LUT
School of Engineering Science
Tietotekniikan koulutusohjelma
Karri Pyykkö

Using virtual machines to create standardized learning environment for software engineering courses

Diplomityö 2022

77 pages, 18 figures, 13 tables, 2 appendices
Työn tarkastajat: Apulaisprofessori Jussi Kasurinen
Tutkijaopettaja Uolevi Nikula

Hakusanat: Virtualisaatio, Opetus, Virtuaalikone, VM, Kontti, Unikernel
Keywords: Virtualization, Education, Virtual machine, VM, Container, Unikernel

Virtualisaatiota on vuosien varrella tutkittu paljon ja sen suosio on viime aikoina kasvanut kevyeiden virtualisaatioteknologioiden kuten konttien ja unikernelien ansiosta. Tässä diplomityössä tutkitaan eri virtualisaatiovaihtoehtoja aiheeseen liittyvän kirjallisuuden kautta, keskittyen erityisesti virtualisaation sovellusalueisiin opetuksessa. Diplomityön osana luodaan räätälöity virtuaaliympäristö, jonka tarkoituksena on tukea ohjelmistotuotannon opiskelijoiden kurssityöskentelyä ja oppimista. Virtuaaliympäristö toteutetaan osana yliopiston ohjelmistotuotannon kurssia ja arvioidaan kurssin opiskelijoille jaetun kyselyn kautta. Diplomityön päämääränä on syventää ymmärrystä eri virtualisaatioteknologioiden hyödyistä ja haitoista, sekä kartoittaa ohjelmistotuotannon opiskelijoiden asenteita räätälöityjä virtuaaliympäristöjä kohtaan. Tulokset osoittivat, että räätälöidyille virtuaalikoneille on kysyntää opiskelun työkaluna. Pienestä otannasta johtuen tuloksia ei voida juurikaan yleistää kurssin ulkopuolelle, mutta siitä huolimatta tulokset osoittivat, että vastaavanlaiset virtuaaliympäristöt vastaanotettaisiin lämpimästi myös muilla yliopiston kursseilla.

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Degree Programme in Software Engineering
Karri Pyykkö

Using virtual machines to create standardized learning environment for software engineering courses

Master's Thesis 2022

77 pages, 18 figures, 13 tables, 2 appendices
Examiners: Associate Professor Jussi Kasurinen
Associate Professor Uolevi Nikula

Keywords: Virtualization, Education, Virtual machine, VM, Container, Unikernel

Virtualization has been extensively researched over the years and has gained popularity recently due to the popularization of light-weight virtualization options such as containers and unikernels. In this thesis the different virtualization options are explored through literature review, with the focus being on educational applications for virtualization. As a part of the thesis, a tailored virtual environment is created with the aim to facilitate course work and learning for software engineering students. The virtual environment is implemented on a university offered software engineering course and evaluated through a survey administered to the course attendants. The objective of the thesis is to gain deeper understanding on the observed benefits and drawbacks of virtualization as a technology to assist learning, and ultimately the thesis aims to gain insight into the attitudes of software engineering students towards tailored virtual environments. The results showed that there is a definite demand for tailored VMs in education. While due to the small sample size, the results can't be generalized outside the scope of the course, the results indicated that tailored virtual environments were overall positively received by the students.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

BYOD	Bring Your Own Device
CaaS	Containers-as-a-Service
CI/CD	Continuous Integration and Continuous Delivery
CLI	Command Line Interface
CPU	Central Processing Unit
DSR	Design Science Research
ESM	Extended Security Maintenance
FTP	File Transfer Protocol
GL	Grey Literature
GPU	Graphics Processing Unit
GUI	Graphical User Interface
I/O	Input/Output
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IoT	Internet of Things
IP	Internet Protocol
IT	Information Technology
LAN	Local Area Network
LMS	Learning Management System
LTS	Long Term Support
LXC	LinuX containers
LXDE	Lightweight X11 Desktop Environment
OCI	Open Container Initiative
OS	Operating System
OVF	Open Virtualization Format
SLR	Systematic Literature Review
SQL	Structured Query Language
STEAM	Science, Technology, Engineering, Arts and Mathematics
USB	Universal Serial Bus
VCL	Virtual Computing Lab
VDI	Virtual Desktop Infrastructure

VM	Virtual machine
VMM	Virtual Machine Monitor
VNC	Virtual Network Computing
WSL	Windows Subsystem for Linux

1 INTRODUCTION

1.1 Background

With the ever-increasing number of different OSs (Operating Systems), software, with the exception of interpreted software, face the challenge of compatibility with the underlying architecture. Not just between different OSs, compatibility issues can arise even within the OS itself. For example, the software might be designed to operate on 64-bit Windows, while the host has a 32-bit copy of Windows. It's also possible that the hardware doesn't support technology employed by the software, e.g. virtualization. (Halsey and Bettany, 2015) While these problems are mostly issues of the past, they still highlight the prevalent themes regarding compatibility. In academic context issues with compatibility place a burden on both the student and the staff as significant amount of time is spent troubleshooting issues with software installation and deployment. Even in the case of deploying intra-institutional software, the process can become a tedious and time consuming task as explained by Corbi and Burgos (Corbi and Burgos, 2017).

With the appearance of Covid-19 (Coronavirus Disease 2019) in Wuhan, China, December 31st 2019, much of the world was caught off guard and many businesses operations were disrupted, and in some cases even completely halted indefinitely. (Sohrabi et al., 2020; World Health Organization, 2021) For Finnish academic facilities the situation was largely the same. As of spring of 2020 both lower – and higher education adopted distance learning, or in some cases hybrid learning, as their primary mode of operation (Vuorio et al., 2021). The sudden pivot to distance learning created a pressing need for solutions to facilitate remote learning. Consequently, much of the educational material, including lectures and workshops, were made available online. While Universities in Finland are slowly in the process of returning to their normal, pre-Covid-19, modes of operation following the regional coronavirus regulations and recommendations provided by Finnish institute for health and welfare, and Ministry of Education and Culture, the need for remote solutions hasn't disappeared. If anything, the epidemic made it apparent that there is a clear demand for remote learning.

While distance learning has its benefits, e.g., elimination of school commutes, flexible schedules, and reform of schools' digital environments, it has its fair share of problems as well. Traditionally much of the academic work is done within the faculty premises. Some of these activities include but are not limited to laboratory work, workshops, seminars, and meetings. while certain activities, such as seminars and meetings, can be transferred online with relative ease using video broadcasting software such as Zoom Meetings, Microsoft Teams, etc., other activities such as workshops and laboratory work are harder to reproduce online. This is largely due to the demand for a very specific toolset or software. Additionally, despite several studies suggesting that distance learning can be as effective as the traditional in-person learning, some studies, such as a study conducted by Butnaru et al., found that the student satisfaction is closely tied to their proficiency at using the provided online tools (Butnaru et al., 2021). While it's hard to teach someone to be proficient with the provided toolset overnight, it's possible to make the tools easier to use and more readily available for the student.

Virtualization is one way to combat these aforementioned problems with software deployment and distance learning. Virtual images can be used to deploy a specific set of software on multiple devices. Following traditional software deployment process, one has to download, install and configure software separately for each device, but thanks to virtualization it's possible to create a pre-configured image with pre-installed and pre-configured software as shown in Figure 1. This reduces the time that is needed to deploy identical software on multiple devices and makes it easier to distribute software directly to students. Not only does the image have the necessary software installed, but it also guarantees that every student is operating on the same operating system, with identical configurations, essentially eliminating majority of potential compatibility issues.

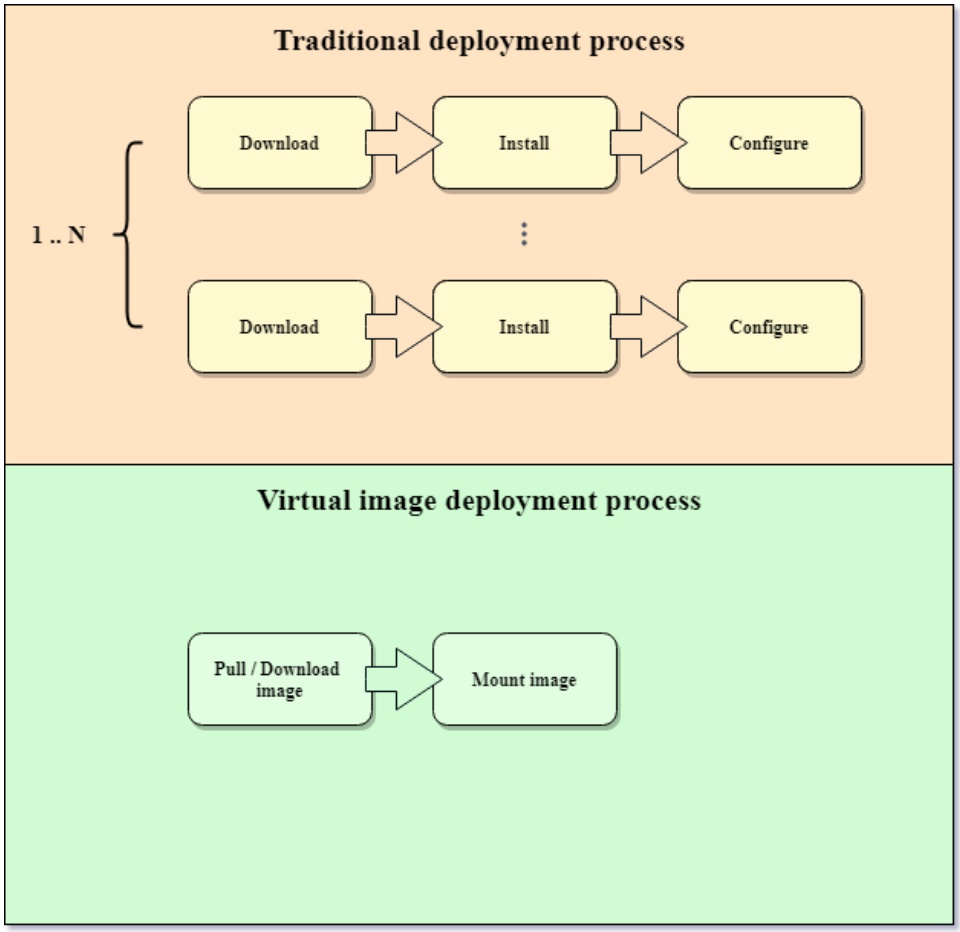

Figure 1. Traditional – and virtual image software deployment processes.

1.2 Goals and limitations

In this thesis we explore the prospect of virtualization for creating homogenous learning environments for educational purposes. The thesis is carried out under the Erasmus+ funded FABLE project, which aims to identify the needs, challenges and expectations of higher education students, lecturers, and administrative staff regarding online and blended learning (Mohammadi, 2021). The goal of the thesis is to build virtual environments for two primary purposes dictated by the needs of LUT university. These are (1) course exercises and (2) course examinations. The aim of the virtualization is to provide the students with a flipped classroom type of learning environment that is homogenous among their peers and educators, eliminating many of the potential issues with installation, deployment, and compatibility of related tools and libraries. In the thesis three main categories of virtualization technologies are explored, which are virtual machines, virtual containers and

unikernel. Suitability of each technology is evaluated based on the application context, i.e., whether the virtualization is used for course work or for examinations.

As a part of the thesis, a tailored virtual environment is created and evaluated by incorporating it as a part of an ongoing university course. Prior literature is used to guide the selection of a fitting virtualization method for artefact creation. The method selection is also heavily guided by the requirements set by the relevant stakeholders in respect to the two primary use cases: *classroom* and e*xamination environment*. The classroom use case is further divided into three university courses, which are (1) *Operating systems and systems programming*, (2) *Basics of Linux*, (3) and *Basics of C programming*. For the artefact evaluation part of the study, the artefact is integrated into a university offered software engineering course *Operating systems and systems programming* and the data collection is conducted by using an opt-in-based survey as the primary data collection method. Additionally, specialized virtual images are considered for the university's examination class environment to better support examinations requiring specialized tools or environments. While the creation of examination environment isn't within the scope of this thesis, it's still weighed in the selection of the virtualization technology, creating a foundation for future research. Furthermore, to uphold the integrity of the examination environment, some restrictions are imposed on the image, e.g., the virtualization software must support resetting of the system image to its original state to ensure no student created files or other committed changes are preserved between examination sessions. Additional restrictions and requirements are further discussed in chapter *4.1.2 Case: Examination environment.*

While cloud-based virtualization is explored through previous literature and widely regarded as the go-to alternative to virtualization, it's outside the scope of the thesis due to restrictions imposed by available resources and other organizational barriers. Currently LUT uses a VMware Workspace ONE for Virtual Desktop Infrastructure (VDI) environment management. While the infrastructure for accessing LUTs virtual resources is already in place, in its current form it doesn't allow teaching staff to readily host their own virtual environments that have been customized to fit their courses' needs. Despite being outside the scope of this thesis, cloud-based virtualization is briefly explored through relevant literature, and the benefits and cons are weighed in relation to other virtualization options.

1.3 Thesis structure

The thesis consists of seven main chapters: *Introduction*, *Literature review*, *Research method*, *Implementation*, *Results*, *Discussion*, and *Conclusion*. The first chapter, Introduction, provides background and the general problem statement for the thesis. It also outlines the scope of the thesis in terms of goals and limitations. The second chapter focuses on previous literature, outlining relevant studies, concepts and conclusions. In the chapter educational use cases for virtualization are explored and different virtualization approaches are explained. At the end of the chapter, existing literature is used to compare the advantages and disadvantages of each virtualization approach, and the most fitting alternatives for the thesis' application context are highlighted. The third chapter creates the foundation for the implementation. In the chapter, selected research methodology and research questions are presented. The fourth chapter details the steps and details for the implementation. The chapter also includes description of the data collection method and – administration. In the fifth chapter results of the carried-out data collection are presented. Following in the sixth chapter the implications of said results are further discussed and opportunities for future studies are proposed. Finally, in the seventh and final chapter the thesis is summarized, and the most important findings are highlighted.

2 LITERATURE REVIEW

In this section relevant concepts for this thesis are explored through prior literature. FABLE project being the initiator for this research, the chapter provides a brief introduction to the project and previous research conducted as a part of the project. The section also discusses motivations for this research through identification of barriers to teaching, and approaches of virtualization in education. Following, relevant virtualization technologies are identified and discussed through prior literature, and lastly conclusions are drawn based on the general consensus of the prior research on the subject.

2.1 FABLE project

FABLE project is an Erasmus+ funded project, which aims to assist higher education teaching staff in implementing blended learning. The objectives of the project, as described by The Research Institute for Innovation & Technology in Education (UNIR iTED), are identifying the needs and expectations of both students and educators related to blended learning, developing methodology to transform in-class courses to blended learning format, design and development of blended learning tools, and issuing a white paper to guide education systems steer their digital transformation effectively. The project started on 1st of March 2021 and is carried out as a collaboration between The International University of La Rioja (UNIR), IPAG Business School, Lappeenranta-Lahti University of Technology (LUT), FH Münster University, Széchenyi István University, Haikara, and Innogate to Europe. The project has a total duration of two years and is being coordinated by IPAG. (©UNIR, 2021)

As a part of the FABLE project, Mohammadi studied current blended learning trends and assessed both student and faculty satisfaction at five different universities in Finland, Germany, France, Hungary, and Spain. In his thesis, Mohammadi also identifies social and technological challenges students face. While the study can be primarily classified as a systematic mapping study, the study also investigates student and faculty satisfaction through a survey. In total 355 participants were included in the survey, with an overwhelming majority of the respondents, 61%, being from Germany. In the study he found that up to 41% of faculty reported being either satisfied or very satisfied with online – and blended learning, while 34% were either unsatisfied or very unsatisfied. Similar percentages

can be observed among students and administrators, with the largest difference being with administrators, who are seemingly the most satisfied with up to 19% being very satisfied. The study also showed that a vast majority of students find using different interfaces and organizing online materials easy. Another interesting finding is that a vast majority of students identified social challenges as the most challenging aspects of online – and blended learning, as opposed to technological challenges. The survey shows that 73% of students find technological challenges the least challenging aspect of online – and blended learning, while 20% identify technological challenges as the most challenging aspect.

2.2 Barriers to teaching

In this chapter we examine some of the existing barriers to online education through prior literature. Education and teaching, especially for higher education, are increasingly moving towards online or blended formats. With online education, as with more traditional face-to-face learning, the role of the faculty is to facilitate learning, and with the classrooms moving online, the educators need to face a whole new set of challenges. The effectiveness of online courses is often questioned due to the higher drop-out rates as opposed to traditional face-to-face courses. (Andreas Altmann et al., 2019) In this chapter we aim to identify some of the barriers that contribute to the observed increased drop-out rates. We also explore other challenges the students face, but don't necessarily correlate with the students dropping out of the course.

In their book Altmann et al. identify social aspects, e.g., distance and social isolation, as the primary culprits for the students dropping out of online courses. In this context, distance refers to transactional distance, i.e., the psychological and communicational distance between the teacher and the student. This correlation between social isolation and student dropouts is also supported by a multitude of other studies as shown by Altmann et al. This notion is also consistent with the findings of Mohammadi as detailed in chapter *2.1 FABLE project*, according to which social challenges are the most challenging aspect of online learning. In their study Altmann et al. propose blended – and collaborative learning as a potential solution to the lack of social interaction that is common with online courses. (Andreas Altmann et al., 2019; Mohammadi, 2021)

In addition to social barriers, Altmann et al. point out that students also face technological barriers, such as assimilating to a new technology. While in their study, they illustrate this problem by using a video recording software as an example, this problem can be generalized to any piece of software the student isn't already proficient in. That said, the technological barriers aren't only affecting the students. In their book Uden et al. write that the faculty staff itself is opposed to learning to use new technologies. While this isn't the case for every teacher, it was observed that many of the older teachers didn't want changes and instead preferred traditional teaching methods. (Uden et al., 2021)

Other than student-oriented barriers, such as social – and technological barriers mentioned earlier, some of the barriers originate from within the faculty. In their book Uden et al. mention the undervaluing of online work as one factor; borrowing their words: "An hour of lecture is equal two hours of working online", which leads to teachers valuing traditional lectures more as opposed to online or blended alternatives. (Uden et al., 2021) Teaching can also be disrupted due to outside factors if the faculty doesn't have contingency plans set in place for scenarios such as natural disasters, wars, and epidemics (Jacqueline Dohaney et al., 2020). Specifically, the effect of the Covid-19 pandemic on higher education was investigated by Nuere & de Miguel. In their paper, they analyze the steps taken by two universities, The university of La Rioja (UNIR) and the Polytechnic University of Madrid (UPM), to adapt teaching so that it may continue uninterrupted despite the crisis. Much like Uden et al., in their study they highlight the importance of teaching technology and the potential barriers that can arise if the faculty isn't prepared for the new technology or unwilling to adapt to the changing environment. They also emphasize that while it's essential to provide online education to continue teaching under the pandemic, it's equally important not to forget the benefits of face-to-face contact in education. (Nuere and de Miguel, 2020; Uden et al., 2021)

While the impact of technological barriers is seemingly overshadowed by social barriers as observed through relevant literature, that doesn't mean the technological barriers should be ignored. Especially in the field of software engineering the importance of different technologies is amplified, and with that the burden of learning to use different tools is higher. For example, for a typical programming course the student needs to know how to use an associated Integrated Development Environment (IDE) and version control system among

other related software. The importance of technology was also demonstrated by Nuere & de Miguel. In their paper they mention Information and Communication Technology (ICT) as a major consideration for facilitating remote education, allowing both asynchronous and synchronous communication between the teaching staff and the student. (Nuere and de Miguel, 2020) While Nuere & de Miguel focused on boosting communication through the use of ICT, in this thesis the focus will be on facilitating the student workflow through virtualization.

2.3 Virtualization in education

For several decades now, ever since its conception in late 1960s to early 1970s, virtualization has been explored as a technology to assist education. The core concept of virtualization is to allow a singular computing environment to run multiple independent systems simultaneously while sharing the same hardware resources (Corbi and Burgos, 2017; Oracle, 2012a). This can either be achieved by complete emulation of hardware components or by leveraging the host operating system to create isolated processes. The emulation process is dependent on the type of used virtualization technology; traditionally virtualization was achieved utilizing virtual machines. Since then, with the release of Docker in 2013, virtualization has drifted towards containerization using container management software such as Docker or Kubernetes. (IBM Cloud, 2021) For virtual machines, the emulation process is managed by a Virtual Machine Monitor (VMM), more commonly known as Hypervisor, while containers utilize the host kernel directly (Oracle, 2012b). In their paper Chen and Zhou point out that while directly utilizing the host kernel solves many of the problems associated with conventional virtual machines, e.g., large overhead and reduced performance, the shared kernel raises some security concerns. To combine the best of each approach, several studies suggest usage of virtualization technology called unikernel, which is able to provide both high level security and small file size. (Chen and Zhou, 2021; Corbi and Burgos, 2017) The pros and cons of each approach are further discussed in the chapters *2.4 Virtual machine*, *2.5 Container*, and *2.6 Unikernel*.

With the increasing prevalence of microservice - and cloud computing paradigms, low-overhead virtualization is becoming more and more popular. The claim is further supported by a study conducted in 2017 by Plauth et al. in which they point out that the number of search results pertaining to Docker and Unikernel have increased notably in the four year

span since 2013. (Plauth et al., 2017; Westphall et al., 2017) Since then the number of search results has more than doubled, with more than 2000 peer reviewed articles for Docker and 18 for Unikernel. It's easy to see why virtualization has become so popular, as it offers a plethora of benefits; in their documentation Oracle lists six main reasons to use virtualization, which are (1) resource optimization, (2) consolidation, (3) maximizing uptime, (4) protection from server and hardware failure, (5) ease of migration, and (6) support of legacy systems (Oracle, 2012c). In educational context the benefits fall mostly in the categories of consolidation, resource optimization, protection from hardware failure, ease of migration, and compatibility. Using virtualization, it's possible to virtualize multiple virtual computers on a single machine, effectively reducing the number of physical computers needed for various school subjects or courses. While it's possible to cram all the required software on a single system, this can quickly lead to excessive software bloat. As such, the creation of specialized images can be highly effective, especially if the focus is on system performance. As an additional benefit, virtualization allows for easier mobilization of software as briefly mentioned in chapter 1.1 and illustrated in Figure 1. In education this enables easy adaptation of the teaching environment for the course's needs and ensures that every student is using identical software (Maleh et al., 2017). Full virtualization, i.e., virtualization of full operating system and hardware, also allows safe education of important concepts such as disk partitioning and networking without affecting the host OS; if an unrecoverable mistake is made, the image can simply be restored to its initial state with relative ease. Due to the isolation of the virtual environment, the students can also have access to administrative privileges without having to risk the integrity of the underlying system. The isolation also helps in protecting the underlying OS from malware by containing the malware within the virtual machine. Lastly, virtualization enables remote access to the educational institution's computers, provided the necessary infrastructure is in place. This eliminates the need for the student to be physically present within the institution's premises and allows the student to access the computing resources whenever needed. (Griffin and Jourdan, 2012; Hu, 2020; Tomer, 2017) Of course, it's also possible to download the full virtual image over the internet or transport it using a portable data storage such as a Universal Serial Bus (USB) flash drive. That said, this is more so the use case for the low-overhead virtualization, e.g., container and unikernel.

Despite its clear advantages, the adoption of virtualization in schools has faced its fair share of barriers such as lacking expertise of teachers and students, insufficient resources, and disagreements between educational institution's stakeholders (Bruce, 2010; Corbi and Burgos, 2017). Similar organizational and behavioral barriers have been observed in wide range of prior literature and they appear to be the main barriers standing in the way of innovation, as pointed out by Salvini et al. (Salvini et al., 2020). In addition to barriers originating from human factors, there are many other barriers as identified by Peter Her in his dissertation. In his paper he writes that the adoption of virtual desktops was hindered by barriers such as lack of resources and competition over available resources, lack of guidance, slow connection, and lab congestion. (Her, 2017) While there doesn't appear to be a universal solution to these barriers, some of them, such as issues caused by slow connection and lack of computational resources, can be alleviated by adopting lighter and more efficient virtualization options such as containers (Corbi and Burgos, 2017). As for the organizational and behavioral barriers, unfortunately they aren't as easily solved. In most cases it requires larger scale socio-institutional changes, i.e., definition of new roles and agreements, change in stakeholder communication, workflow schemes and skills (Salvini et al., 2020).

Virtualization is extensively researched, as can be deduced from the sheer number of research papers related to the subject, with most of the modern studies focusing on light-weight virtualization technologies. Corbi and Burgos examined containers, specifically Docker, as a tool to improve student experience in solving complex exercises related to mathematics, physics, and other STEAM (Science, Technology, Engineering, Arts and Mathematics) subjects. The difficulty levels of the given assignments were variable; some tasks required the student to write simple python scripts, while other tasks involved complex simulation scenarios with a very specific toolset. In their study they observed that student engagement and satisfaction increased gradually throughout the duration of the case study. According to their results, compared to previous years, the percentage of submitted alternative activities rose by roughly 40 percentage points. The results also show that vast majority, 80 percent to be exact, of the students were either very satisfied or satisfied with the application of containers, while only 10 percent were dissatisfied, and the remaining 10 percent were indifferent. (Corbi and Burgos, 2017) Similar results were observed by Chen in their report, where they designed and implemented a container based virtual lab system called CvLabs. Their system received positive feedback from learners with varying

backgrounds, and consequently showed that containers are a noteworthy solution to building virtual labs. (Chen, 2020)

Virtual machines (VMs) being the predecessor to containers, particularly the older studies focus on the utilization of virtual machines. While today VMs are largely regarded as outdated, they still can have their uses and have several advantages over modern virtualization options as will be later detailed in chapter *2.4 Virtual machine*. Much like containers, largely similar results were observed for VMs in related literature. In their book, Childs and Soetanto experimented with decentralized virtual labs – decentralized meaning the labs are hosted directly on students' personal computers instead of a centralized server. Of course, electing to use decentralized approach works based on the assumption that every student has access to a personal computer outside classrooms. While this can sound like a dangerous assumption to make, a study conducted back in 2014 showed that minimum of 85 percent of students enrolled in colleges and universities owned laptops (Tomer, 2017). Another study conducted in 2015 showed that 42 percent of colleges or universities in US have Bring Your Own Device (BYOD) policies in place (Johnson et al., 2016). It should also be noted that if the studies were conducted today and they included other devices besides laptops, the numbers would more than likely be significantly higher. In their approach, Childs and Soetanto distributed instructor created pre-configured Linux VMs to the students through the class File Transfer Protocol (FTP) server. In their experiment the students were using two separate VMs to simulate a penetration testing environment where one VM acts as the attacking client and the other as the defender. In its early stages, the attacking client was hosted on Debian Linux 3.1 and the defending client on CentOS Linux 5.0, each running on VMware Player and VMware Server – later replaced by VMware Workstation. During the following iterations the images were customized for smaller file size and in the third year the hypervisor was replaced by Sun xVM VirtualBox, also known as Oracle VM VirtualBox. Over the duration of the experiment, they reported that the student feedback was overall positive and the student performance was consistent. In the same study, Childs and Soetanto experimented with centralized virtual lab approaches, which in this case refers to cloud hosted central server that can be used to access hosted VMs. Their arrangements included tools such as Virtual Computing Lab (VCL), VMware Lab Manager and ProxMox Virtual Environment for resource pooling, allocation, and virtualization management. The experiment showed that while centralized cloud-based approach was cost effective, it didn't

scale well with the number of VMs. (Childs and Soetanto, 2017) That said, many aspects such as growing need for digital teaching and expansion of research projects result in universities investing more and more into servers and other computing resources, which in turn leads to waste of resources if they are left unutilized. In cases such as this, pooling resources and making them available remotely through a cloud can be beneficial. (Ma et al., 2014)

2.4 Virtual machine

Earlier we defined the core concept of virtualization as enabling one physical computer to run multiple independent systems simultaneously while sharing the same hardware resources. Expanding upon this definition, virtualization can be considered as the implementation of partial or complete hardware components through the means of emulation (Corbi and Burgos, 2017). A virtual machine (VM) is a term we use to refer to these isolated virtual systems that operate on top of emulated hardware. The emulation, and other tasks such as creation, management, and isolation of VMs, is handled by software called a hypervisor, sometimes also called virtual machine monitor (VMM). The physical hardware that a hypervisor operates on is generally called the host, while the VMs created and managed by the hypervisor are called guests. Besides creation of VMs and high-level management, a hypervisor is also responsible for allocation and management of hardware resources. A hypervisor borrows resources such as CPU, GPU, and memory from the host and allocates them between guests. To ensure that the host is left with sufficient resources for its own operation, the hypervisors typically set a maximum limit for each resource. The isolation of guests also ensures that the concurrently running OSs don't interfere with each other. Meaning that if one OS is compromised, the other OSs aren't affected by it. (Hu, 2020; IBM Cloud Education, 2019; Red Hat, 2020)

Two different types of hypervisors exist, and they are called *type 1* and *type 2* hypervisors. Type 1 hypervisor, also known as bare-metal hypervisor, runs directly on the computer's hardware replacing the host OS. As the hypervisor runs directly on top of the hardware, it schedules resources directly to the hardware. Type 2 hypervisor, also known as a hosted hypervisor, runs on OS as a separate application. Instead of directly accessing the hardware resources, the resources are scheduled by using the host OS as an intermediary. Visual comparison of the hypervisors is shown in Figure 2. Type 2 hypervisors typically have a

higher latency than type 1 hypervisors, as type 1 hypervisors can directly access the hardware. The direct access to the hardware also comes with increased security, since due to the direct coupling of the hypervisor and hardware the attackers have reduced number of attack vectors they can exploit. That said, type 1 hypervisors typically need a separate machine for management of VMs and resource allocation, which raises the threshold for adoption in the form of higher initial investment compared to Type 2 hypervisors. For the above reasons Type 1 hypervisors are a good fit for enterprise data centers or other server-based environments, while type 2 hypervisors are a better fit for individual users that are looking to take advantage of virtualization on their personal computers. (IBM Cloud Education, 2019; Red Hat, 2020)

One consideration that seems to always come up is how VMs compare to native. Over a decade ago in 2008, VMware conducted their own performance study on VMs that were running SQL server databases. Their results showed that VMs provided near native performance. Specifically, they note that the hypervisor was capable of seamlessly handling the varying demand in resources and was able to prevent resource hogging by a few virtual machines, ensuring that each virtual machine had sufficient computational resources. (VMware, 2008) Since then the hypervisors have progressed significantly and especially type 1 hypervisors are able provide performance that is nearly identical with native environments. Currently there are many different hypervisors on the market; some examples of popular type 1 hypervisors are *KVM*, *Microsoft Hyper-V*, and *VMware vSphere*. As for type 2, some of the more known type 2 hypervisors are *VMware Workstation* and *Oracle VirtualBox*. Ultimately the choice of hypervisor depends on multiple factors. For example, if the focus of the virtualization is on performance, type 1 might be preferred. On the other hand, if the hypervisor is in direct use of the end-user such as a student, the focus might be on usability and available documentation on the hypervisor. (IBM Cloud Education, 2019; Red Hat, 2020)

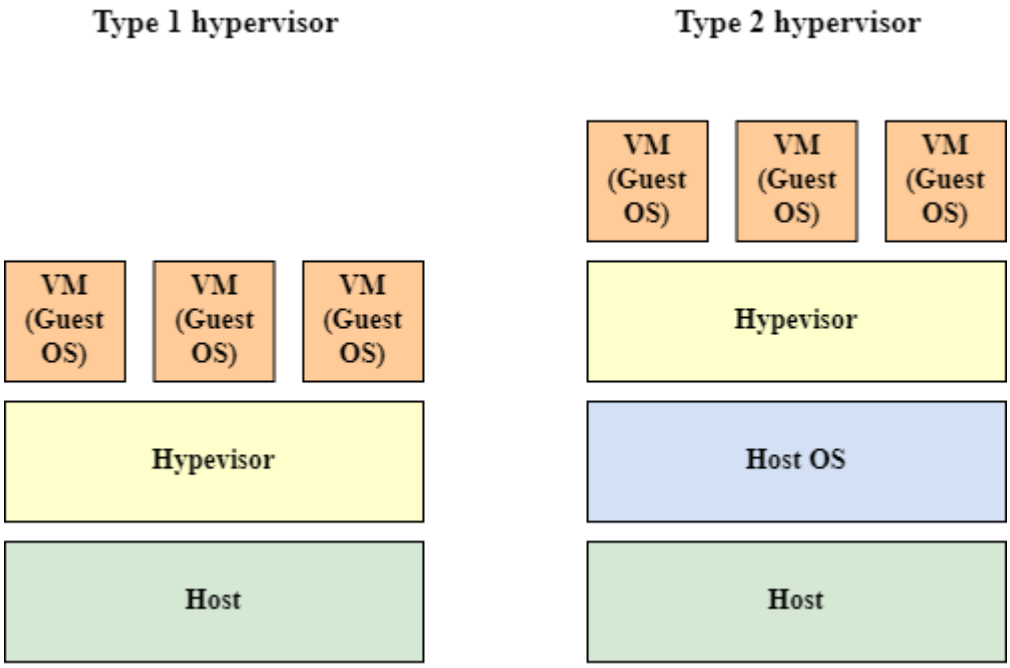

Figure 2. Type 1 hypervisor vs. Type 2 hypervisor.

Table 1. Observed benefits of virtual machines in related literature.

Item	**Justification**	**Literature**
Cost effectivity	Reducing the physical infrastructure footprint.	(Childs and Soetanto, 2017; IBM Cloud Education, 2019)
Agility / Portability	Ease of setting up new environments.	(Childs and Soetanto, 2017; IBM Cloud Education, 2019)
Reliability / Stability	Reducing downtime, ease of migration from one hypervisor to another. Ease of maintenance due to recoverability. Performance stability.	(Behravesh et al., 2019; Childs and Soetanto, 2017; IBM Cloud Education, 2019)
Multiple OSs	Hypervisors allow for multiple different types of OSs.	(Red Hat, 2020)
Monolithic workloads	VMs are capable of running more operations than a single container.	(Red Hat, 2020)
Isolation	Isolation of sensitive and high risk-development cycles.	(Red Hat, 2020)
Infrastructural resources	Provision of emulated resources such as networks, servers, and databases.	(Red Hat, 2020)

Expandability / Scalability	VMs are easily expandable as long as there are available hardware resources.	(Childs and Soetanto, 2017)
Responsiveness / Latency	locally hosted VMs have better responsiveness than remotely hosted physical – or virtual machines	(Childs and Soetanto, 2017)

Table 2. Observed cons of virtual machines in related literature.

Item	**Justification**	**Literature**
Size	VMs contain their own OS, making them much heavier than light-weight virtualization options, such as containers, that use a shared OS.	(Red Hat, 2020; Watada et al., 2019; Westphall et al., 2017)
Computational resource footprint	VMs can have relatively large resource utilization and the host computer can easily run out of resources especially when multiple VMs are running concurrently. Especially the memory and CPU consumption are high due to the high amount of background processes.	(Behravesh et al., 2019; Chen and Zhou, 2021; Childs and Soetanto, 2017; Watada et al., 2019; Westphall et al., 2017)
Start-up time	The start-up times for VMs are significantly longer when compared to containers and unikernels due to the raw size of the system.	(Chen and Zhou, 2021)
Large attack surface	Full kernel leaves the OS more open to vulnerabilities and most common exploits.	(Chen and Zhou, 2021)
License restrictions	Licenses can be problematic when implementing decentralized VMs.	(Childs and Soetanto, 2017)

2.5 Container

One of the main problems of VMs is that the virtualization of the whole kernel results in excessive overhead, especially for the purpose of packaging small applications (Westphall et al., 2017). Containers were created as an answer to this problem; since containers share the kernel with the host OS, the resulting image size is significantly smaller in comparison. The sharing of the kernel does however come with a major downside; all containers on the same host must use the same kernel, which means all the containers are restricted to one type of OS determined by the host (Mehringer and Barker, 2020; Red Hat, 2020). That said, this shortcoming can be worked around by using a VM to virtualize a container host that uses a separate kernel. For example, Docker, a popular open source PaaS (Platform as a Service) containerization software, uses Moby VM and Microsoft Hyper-V to run Linux containers on top of a Windows desktop (Cooley et al., 2021). The main application of containers is distribution services and applications, which also makes it desirable for educational purposes. Containers are also becoming increasingly popular within research groups as a tool to aid reproducibility of experiments. (Corbi and Burgos, 2017; Mehringer and Barker, 2020)

There are two main types of containers: application- and OS-oriented containers. Difference being that application-oriented containers are used to run single applications, while OS-oriented containers replicate the entire OS environment. Application-oriented containers are fundamental building blocks for microservices, and they allow one to run an application independent of the platform. OS-oriented containers are largely similar to VMs, however they are lighter as only the OS is virtualized instead of complete hardware. Docker is one of the more popular application-oriented containerization software currently on the market. Whereas VM isolation is handled by a hypervisor through hardware emulation, Docker implements isolation by utilizing features of Linux kernel such as namespaces and cgroups (Chen, 2020; Tomer, 2017). Docker is open source, which is without a doubt a big part of its success. Besides high performance, one of the key features of Docker is that its containers can be easily shared as images through the Docker registry, which is also known as the Docker Hub. Docker Hub allows for cloud hosting of pre-configured images, making the distribution of different images easy. The public docker repositories include pre-configured images for various purposes; for example, if one wishes to set up a database server that uses

document-oriented NoSQL database model, one can simply download an image that has been pre-configured to run a MongoDB server. (Tomer, 2017) Docker also supports plethora of different development and management features such as automated builds, access management, and webhooks, making it a fitting tool for organizations who are looking to level up their DevOps practices by incorporating continuous integration and continuous delivery (CI/CD) as a part of their development pipeline (Docker, 2021).

Besides Docker, several other containerization software exist, some of the more known ones being AWS Fargate and Microsoft Azure; both of which are examples of Containers-as-a-Service (CaaS) pattern that keeps gaining traction especially when paired with microservices. Back in 2016 it was suggested by Singh and Singh that in the future majority of microservice applications will be built on containers (Singh and Singh, 2016). While it's hard to give exact numbers or generalize to a larger group, it's fair to say that their claim wasn't far off; two researchers at O'reilly media ran a survey in early 2020 that indicated that more than 75 percent of the respondents have used microservices, and almost 62 percent are using containers to deploy at least some of their services. They also reported that respondents who used containers to deploy microservices were much more likely to report success than those who didn't. While containers don't seem to be the most popular option for deploying microservices just yet, it's certainly progressing in that direction. (Loukides and Swoyer, 2020) Docker, being an example of application-oriented containerization, is useful for running single applications. However, if the objective is full OS virtualization it is better achieved by OS containerization software such as LinuX containers (LXC). Besides the examples given in this chapter, there are several container orchestration and management software such as Kubernetes and Vagrant that are used to manage containers, but they are outside the scope of this chapter.

Table 3. Observed benefits of containers in related literature.

Item	**Justification**	**Literature**
Size	Containers are typically light weight due to their shared OS. They are also usually used to package single functions or applications.	(Behravesh et al., 2019; Mavridis and Karatza, 2021; Red Hat, 2020; Watada et al., 2019)

Portability	The light weight of containers makes them easy to move between different environments.	(Behravesh et al., 2019; Chen and Zhou, 2021; Mavridis and Karatza, 2021; Red Hat, 2020; Watada et al., 2019)
Consistency	All the dependencies needed to run the application are enclosed inside the container, making containers consistent across platforms.	(Mavridis and Karatza, 2021)
Compatibility with microservices, CI/CD, cloud practices	The smaller units related to microservices and CI/CD are compatible with the light-weight containers. Ideal environment for cloud-native apps.	(Red Hat, 2020)
Easy and fast deployment	Container start-up times are significantly lower in comparison to VMs. Containers only need few milliseconds to start up while VMs typically need at least a few seconds.	(Chen and Zhou, 2021; Mavridis and Karatza, 2021; Watada et al., 2019)
Efficiency	The light-weight virtualization approach of containers reduces the resource overhead and improves performance. More efficient process management in comparison to unikernels and overhead of hypervisor is eliminated.	(Behravesh et al., 2019; Chen and Zhou, 2021; Mavridis and Karatza, 2021; Watada et al., 2019)
Rapid development	Being at the cutting edge of virtualization, the technology is developing and improving fast.	(Chen and Zhou, 2021)
Scalability	Containers enable easy scaling of resources based on the demand.	(Behravesh et al., 2019)
Ease of maintenance	Simplified maintenance	(Chen and Zhou, 2021)

Table 4. Observed cons of containers in related literature.

Item	Justification	Literature
Single OS	Containers are restricted to one type of OS.	(Red Hat, 2020)
Security and isolation	Isolated processes can utilize the underlying host kernel and are vulnerable to majority of kernel exploits. Process-level isolation enabled by namespaces is considered weaker than hardware-level isolation of traditional VMs.	(Chen and Zhou, 2021; Madhavapeddy et al., 2013; Mavridis and Karatza, 2021; Watada et al., 2019)

2.6 Unikernel

Unikernels can be considered an attempt to combine the best of both traditional VMs and containers, being both light-weight and offering robust security through reduced attack surface. Compared to containers, unikernels take the step even further by cutting out the middleman, i.e., the OS, and operate directly on top of the host OS by leveraging the concept of library OSs. Instead of employing the full OS, library OSs provide regular OS features to the applications through libraries. While Unikernels are fundamentally different from traditional VMs and containers, they still use a hypervisor for resource allocation and hardware abstraction. (Madhavapeddy et al., 2013; Williams et al., 2018)

In literature unikernels aren't quite as extensively researched as other virtualization technologies. The research surrounding unikernels is typically focused on its two selling points: (1) small resource footprint and (2) security. The small resource footprint is a result of eliminating OS services, features, and libraries that aren't needed for the application. By eliminating the unnecessary libraries, also the attack surface shrinks in the process, making them less susceptible to being exploited. Unikernels can run only one application inside a single unikernel, making them especially suitable for microservice architecture and it's no wonder related literature around unikernels is typically focused on deployment of unikernels in conjunction with cloud computing (Behravesh et al., 2019; Chen and Zhou, 2021; Madhavapeddy et al., 2013; Mavridis and Karatza, 2021; Williams et al., 2018). Another common theme in literature related to unikernels is performance benchmarking in

comparison to other virtualization technologies, which is a topic that will be further discussed in chapter *2.8 Final comparison and consensus*.

Table 5. Observed benefits of unikernels in related literature.

Item	Justification	Literature
Size	Reduced image sizes through elimination of OS and usage of only essential libraries.	(Behravesh et al., 2019; Madhavapeddy et al., 2013; Mavridis and Karatza, 2021)
Efficiency	Improved efficiency and low overhead in comparison to VMs. Elimination of context switching results in reduced overhead.	(Behravesh et al., 2019; Chen and Zhou, 2021; Madhavapeddy et al., 2013; Mavridis and Karatza, 2021)
Security	Reduced attack surface and the isolation techniques of hypervisor provide a high level of security.	(Chen and Zhou, 2021; Madhavapeddy et al., 2013; Madhavapeddy and Scott, 2014; Mavridis and Karatza, 2021)
Cost	Lower operational costs.	(Madhavapeddy et al., 2013)
Ease of deployment	Small binary size enables fast deployment to cloud or other physical devices.	(Chen and Zhou, 2021; Madhavapeddy and Scott, 2014)
Independency from the underlying system	Unikernels can be deployed directly on top of bare-metal hardware.	(Chen and Zhou, 2021)

Table 6. Observed cons of unikernels in related literature.

Item	Justification	Literature
Compatibility	Containerization using unikernels can cause compatibility issues due to the images not following the OCI format.	(Mavridis and Karatza, 2021)
Inefficient process and memory management	The process completion times for unikernels are longer due to more inefficient memory management.	(Behravesh et al., 2019)
High burden on cloud orchestration	Creates the need for more VM deployments as every changed configuration requires redeployment.	(Madhavapeddy and Scott, 2014)

2.7 Cloud-based solutions

Instead of being its own virtualization technology, cloud-based solutions in this context refer to utilization of the full array of virtualization techniques, i.e., VMs, containers, and unikernels, as a part of a cloud environment. Virtualization technologies are often used in cloud computing to fragment the resources of the physical host into smaller chunks. For a long time, VMs have been used in cloud computing by utilizing virtualization software such as VMware and VirtualBox, however with microservices architecture becoming more popular, it soon became apparent that VMs were far too monolithic for smaller applications and incurred a large resource overhead. (Chen and Zhou, 2021) As an answer to this problem, containers quickly gained popularity as they were better suited for smaller single-purpose applications. However, in multitenant cloud architecture, the smaller size of containers didn't come without a cost. Due to the shared host kernel, albeit having a smaller attack surface, containers pose their own unique security risk in the form of weaker isolation (Chen and Zhou, 2021; Mavridis and Karatza, 2021).

From an educational standpoint, cloud-hosted VMs allow students a streamlined remote access to a working environment. These cloud-hosted VMs can be hosted directly by the institution or a third-party provider such as AWS (Amazon Web Services), Microsoft Azure, and Google cloud. While from a student's point of view cloud-hosted virtual machines might be preferred, hosting VMs in the cloud poses significant challenges such as lack of computational resources, reduced performance and higher operational costs. (Corbi and Burgos, 2017; Maleh et al., 2017) That said, some of these claims such as the operational costs have been debated in relevant literature. In his study, Tomer claims that cloud-based teaching solutions can even produce cost reductions (Tomer, 2017). These claims are supported by a study conducted by Koch et al., which states that cloud computing can produce cost reductions all the way up to 30%, compared to conservative resource allocation strategies (Koch et al., 2016).

Table 7. Observed benefits of cloud computing in related literature.

Item	Justification	Literature
Cost and resource efficiency	Dynamic allocation of computational resources	(Chen and Zhou, 2021; Madhavapeddy and Scott, 2014; Maleh et al., 2017)
Simplicity	Simplified lab environment setup and lower management effort	(Chen, 2020)
Flexibility	Cloud computing enables a highly flexible environment that is ideal for learning.	(Tomer, 2017)

Table 8. Observed cons of cloud computing in related literature.

Item	Justification	Literature
resource allocation	Resource allocation in a multi-tenant cloud environment can be problematic.	(Maleh et al., 2017)
reliability / flexibility	Difficulty of accessing VNC (or VDI) server. Dependence on internet connection.	(Maleh et al., 2017; Xu et al., 2012)
learning curve	Depending on the cloud platform	(Maleh et al., 2017)

2.8 Final comparison and consensus

In their study Watada et al. compared the performance of containers, unikernels, and VMs on five primary factors: (1) CPU performance, (2) memory usage, (3) network bandwith, (4) execution time, and (5) technological maturity. Their results showed that the CPU performance of Docker containers are near native systems for a single instance, however their performance is diminished for multiple instances. (Chen and Zhou, 2021; Watada et al., 2019) In their study they accentuate that containers delivered satisfactory performance in nearly all measured aspects. They also point out that despite the satisfactory benchmarks, containers have their fair share of problems related to isolation, security, stability, cross-platform support, and persistent storage. Overall, their results show that both containers and VMs show good and consistent CPU performance, and while unikernels show promise and offer significantly better isolation, similar to that of VMs, they still haven't achieved the

desired technical maturity. As for network bandwidth benchmarking, they found out that the performance of containers (Docker and rkt) and unikernel (OSv) were found to be better than that of VM (kvm). The performance of VM was also weakest out of its peers for memory management. Watada et al. measured the performance for random memory accesses using a GUPS rating, which profiles the system memory architecture. Based on their results VM obtained the lowest GUPS score, while containers and unikernels achieved near native performance. Overall, from their research it can be concluded that despite their sizeable legacy and improvements over the years, VMs are found lacking in comparison to containers and unikernels in performance. (Watada et al., 2019) The findings of Watada et al. are supported by research conducted by Behravesh et al. and Aggarwal and Thangaraju. In their paper Behravesh et al. detailed largely similar results for image sizes, memory management, and CPU usage. They also point out that while unikernels have smaller image sizes and smaller memory consumption, the process completion times for unikernels are longer due to more inefficient memory management. Adding to this, Aggrawal and Thangaraju point out unikernels are likely to perform better for applications with high context switching as unikernels operate under single address space (Aggarwal and Thangaraju, 2020; Behravesh et al., 2019)

While performance is often the primary characteristic for consideration, especially in related literature, in reality the selection of virtualization technology isn't always quite as black and white, mainly due to various different aspects one needs to consider; for example, if the stability of the system is important, VMs are seemingly still the superior option (Behravesh et al., 2019; Watada et al., 2019). On the other hand, if security takes the priority, then both VMs and unikernels have containers beat (Madhavapeddy and Scott, 2014; Mavridis and Karatza, 2021). Another aspect that should be considered is the underlying architecture and application purpose. For example, due to its small size and small resource footprint, unikernels are seemingly a perfect fit for IoT (Internet of Things) applications, where the computing resources are often very limited (Watada et al., 2019). In the end there is no clear winner, and the selection of virtualization technology should be carefully considered based on application context.

3 RESEARCH METHODOLOGY

In this section the used research method is explained, and the steps taken to conduct the research are presented. The section details the theoretical framework of this thesis, which in this case comprises of the selected research questions, their justifications, and the literature review methodology.

3.1 Research method

In this thesis, the research method falls within the category of design science research (DSR). DSR is methodology that is focused on artefact creation as a way to solve real-life problems, and emerged as a way to address the problem of design and implementation of information systems in a scientific manner (De Sordi, 2021; Prat et al., 2014). While there are many ways to approach DSR, typically the process consists of four primary steps, which are *observation*, *theory building*, *artefact design and implementation*, and *artefact evaluation* (Ahmed and Sundaram, 2011; Herselman and Botha, 2015). In this thesis, observation and theory building are the two steps largely covered by the theoretical part of this thesis. Firstly, following the DSR process, the research questions, use cases, and elicited requirements are used to identify the problem area. Once the problem area is identified, and a course of action is selected based on the observed requirements and literature review, the artefact is built as a part of the artefact design and implementation. Finally, the artefact is evaluated using a survey as the primary data collection method. Due to the iterative nature of DSR process, the artefact is constantly evaluated against a set of criteria and evolves with subsequent iterations (Herselman and Botha, 2015). While due to time constraints, in this thesis, the iterative nature of DSR is heavily restricted, the research creates a foundation for future research on the subject of using virtualization to create standardized learning environments.

3.2 Research questions

In this chapter three primary research questions are presented. Each research question highlights an important topic that are further examined in this thesis. As the purpose of this thesis is to facilitate learning and course administration, all three of the research questions are closely related to education.

RQ-1 *Can virtualization be used to facilitate learning?*

RQ-1 attempts to find out whether virtualization can have a positive impact on student's learning. The question is primarily answered through literature review's chapter *2.3 Virtualization in education* and chapter *5 RESULTS*. To answer the research question, both previous literature and data collected from a survey administered to students is used.

RQ-2 *What are the application areas for light-weight virtualization in education?*

RQ-3 *How does light-weight virtualization compare to virtual machines?*

RQ-2 and RQ-3 are both related to light-weight virtualization options such as containers and unikernel. RQ-2 explores the educational application areas for light-weight virtualization, as well as the feasibility of light-weight virtualization in the context of the thesis' usage scenarios listed in chapter *4.1 Use cases*. The feasibility of light-weight virtualization as an educational tool for environment distribution is based on previous literature in chapters *2.3 Virtualization in education*, *2.5 Container*, and *2.6 Unikernel*. The feasibility is further evaluated in chapter *4.2.1 Docker containers*, where container artefact implementation is detailed. RQ-3 focuses on mapping the primary differences between light-weight virtualization and traditional hypervisor-based virtualization, i.e., virtual machines.

RQ-4 *What are the barriers for adopting virtualization?*

RQ-4 considers the potential barriers to adopting virtualization, with the focus being on education. The purpose of the research question is to identify possible problem areas that can be improved upon either through virtualization technology and – tool selection or by implementing organizational changes. Some of the prevalent barriers identified in the previous literature are covered in chapters *2.2 Barriers to teaching* and *2.3 Virtualization in education*.

3.3 Literature selection

To pick relevant studies, a Multivocal Literature Review (MLR) process is used. MLR is a type of Systematic Literature Review (SLR) that on top of peer reviewed literature also includes Grey Literature (GL) (Garousi et al., 2022; Pereira-Vale et al., 2021). GL has

multiple definitions, however in the context of this thesis we will be using the so-called Luxembourg definition, which is regarded as the most accepted definition of GL. The definition defines GL as "…that which is produced on all levels of government, academics, business and industry in print and electronic formats, but which is not controlled by commercial publishers, i.e., where publishing is not the primary activity of the producing body". (Farace and Schöpfel, 2010; Garousi et al., 2022; Schöpfel, 2010)

The literature review framework used in this thesis is based on the Pereira-Vale et al.'s MLR process model. However, instead of the 10 steps detailed in their paper, in this thesis the process is consolidated into six primary steps: (1) search engine and database selection, (2) search terms and searching strategy, (3) study selection, (4) snowballing, (5) merging and filtering, and (6) data extraction. Largely similar steps were also suggested by Petersen et al. in their guidelines for conducting systematic mapping studies in software engineering, with a heavier emphasis on data validity (Petersen et al., 2015).

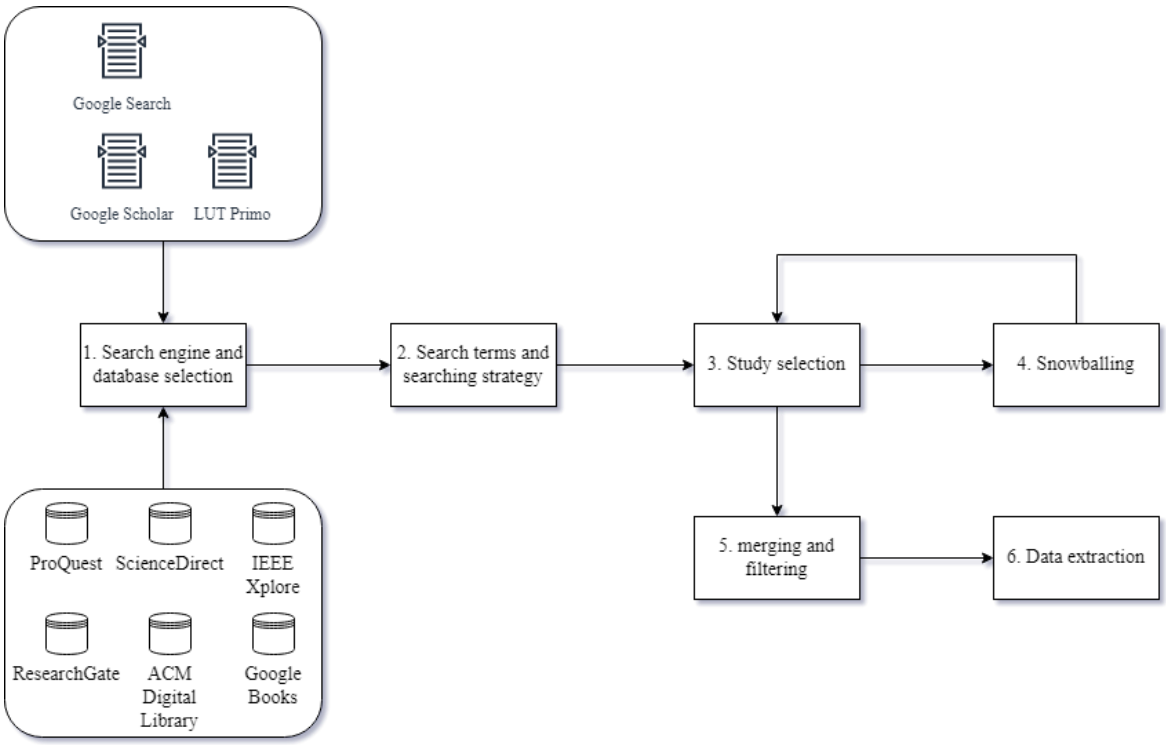

Figure 3. MLR framework.

3.3.1 Search engine and database selection

The selection of search engine is primarily determined by available resources and the scale of the search engine's indexed literature, i.e., which search engines offer an extensive catalog and are readily accessible under LUT license without incurring additional costs. Three primary search engines are selected for this thesis: LUT Primo, Google Scholar, and Google Search. LUT Primo is used as the primary search engine for academic resources such as books, journal articles, review articles and databases. To complement LUT Primo, Google Scholar is mainly used to explore literature that isn't included in LUT Primo's supported databases (ProQuest, ScienceDirect, IEEE Xplore, ResearchGate, ACM Digital Library and more). While Google Scholar has been criticized for its inaccuracy when it comes to systematic searches, it's suitable for exploratory searches. Google Scholar being a crawler-based search engine, it boasts a very large catalog of literature, making it especially useful for discovering literature that can't be found in other databases. (Gusenbauer and Haddaway, 2020) Finally, Google Search is used for exploring relevant Grey Literature. While one would be hard pressed to find peer reviewed articles from Google Search, it can be used for complementary information such as expert blogs.

3.3.2 Search terms and searching strategy

Exploration of literature is conducted using relevant search terms that answer the research questions set in chapter *3.2 Research questions*. Some of the search terms used in the literature review include keywords such as: Virtual machine, Docker, Virtualization, Teaching, Education, VM, Unikernel. The keywords can be used individually or combined to yield more accurate search results, e.g., "Virtualization education" or "Docker teaching". In some cases, truncation of words is applied to get a greater number of relevant search results. For example, truncating the word "strategy" as "strateg" allows for results including words "strategies", "strategic", etc. The truncation of keywords is primarily applied with LUT Primo search engine as truncation isn't a viable search tactic for Google Scholar and Google Search, since they already use built-in truncation.

3.3.3 Study selection

There are three criteria for initial study selection: (1) relevance, (2) publication date, and (3) trustworthiness. At this stage, relevance is primarily evaluated through the publication's title and abstract. Publication date is used to guide the selection by giving priority to more recent publications. Trustworthiness is evaluated based on two factors: number of citations and whether the publication has been peer reviewed or not. Out of the two, heavier emphasis is placed on peer reviews as the number of citations can greatly vary based on the publication date.

3.3.4 Snowballing

Snowballing is the process of discovering new literature and expanding the pool of studies utilizing the references of initially selected studies. In this thesis snowballing is conducted on a smaller scale; instead of adding every referenced study into the pool, the studies were selected based on context. As pointed out by Petersen et al., one should aim for a better sample instead of finding all the available articles (Petersen et al., 2015). While this advice was given in the context of conducting systematic mapping studies, it's a good philosophy to follow in general.

3.3.5 Merging and filtering

Filtering and exclusion of publications is initially done based on titles and abstracts as mentioned earlier in chapter *3.3.3 Study selection*. The second round of filtering and merging of similar studies is done upon closer examination. The primary reasons for exclusion during the second round of filtering are the publication date and relevance of the study. While during the study selection phase the relevance was evaluated based on the publication's title and abstract, in this step the evaluation is done based on the full text.

3.3.6 Data extraction

The goal of this step is identifying relevant information and summarizing the principal findings of the included studies though data synthesis. While in the previous step *Merging and filtering* the related studies were grouped together, the goal of this step is to synthesize and present all the available information in the form of literature review. The themes for the

literature review are largely guided by the prevalent themes in the source literature, as well as the research questions set in chapter *3.2 Research questions*.

4 IMPLEMENTATION

In this section, two use cases for virtual environments are presented. These use cases include a high-level description of the core problems associated with each environment, as well as a set of preliminary requirements that are elicited from relevant stakeholders. Following, the experimentation details for the artefact creation, for both container and VM implementations, are detailed. Finally, the data collection and data analysis method, including its benefits and cons, are discussed largely through prior literature. The purpose of this section is to provide a detailed description of both the artefact implementation and test arrangement for increased reproducibility.

4.1 Use cases

4.1.1 Case: Classroom

In total there are three LUT courses that have a need for customized virtual environments for classroom purposes. These courses are (1) *Operating systems and systems programming*, (2) *Basics of Linux*, (3) and *Basics of C programming*. The need for virtual environments stems from the demand for uniform environments for course exercises. While the university offers on-premises workshops and classrooms that are equipped with desktop computers that operate under similar software configurations, the capacity of the classrooms is limited, forcing some of the students to use their personal computers for course work. There is also the case of distance learning, which is increasingly prevalent due to the shift in educational policies introduced as a response to the Covid-19 pandemic. In remote education, it's often unreasonable to assume that each student is utilizing the same software configuration. As such, much of the educational resources such as the effort of teaching staff is directed to solving issues related to environmental differences, e.g., software compatibility, missing or misconfigured libraries, PATH system variable configurations etc.

While compatibility and configuration issues are a common place within many disciplines, they are especially common for software engineering and computer science courses. The aforementioned three courses are all part of the LUT university's software engineering undergraduate curriculum. These courses need an environment for basic exercises and a sand box for writing code and executing programs. Many of the course exercises are coding

related and vary from writing simple command line instructions to writing, compiling, and executing complete programs. *Basics of Linux* -course is focused on teaching the student entry-level skills related to deployment and usage of Linux system; some of the core concepts include Linux terminal and basic commands, basic functionality of Linux graphical user interface (GUI), writing command line applications, and file management. In contrast, the two other courses *Basics of C programming* and *Operating systems and systems programming* are significantly more programming focused. *Basics of C programming* aims to teach the core concepts of C-programming, while *Operating systems and systems programming* teaches more advanced topics such as advanced I/O operations, process and memory management, threads, and signals. Each of the courses place their own requirements based various factors such as the operating system, libraries, and tools. Additionally, since all the courses utilize the OS or other GUI elements such as text editors as a part of their course exercises, in their current state, the courses need GUI support to be functional, which greatly restricts the selection of virtualization technology. As each of the courses are unique, they come with their unique requirements, which are detailed in Table 9.

Table 9. Case: Classroom, primary requirements.

ID	Requirement	Description
CR-1	Supported host OSs	Windows 7 or newer
CR-2	Image OS	*Basics of C programming*: Windows 7 or newer *Operating systems and systems programming*: Ubuntu 19.04 or newer *Basics of Linux*: Ubuntu 19.04 or newer
CR-3	Maximum image size	15GB or less
CR-4	GUI support	GUI support is required
CR-5	User costs	The virtualization should incur zero costs for the user

4.1.2 Case: Examination environment

LUT has an (EXAM) examination classroom with multiple desktop computers for course examination purposes. The computers are managed and maintained by the university's

department of information management, and deviations from the standard installation are avoided. Additionally, as the examination classroom is used for a number of different courses, it's not possible to have course specific software pre-installed on each desktop. As such, there a need for virtual machines with pre-configured images, which mimic the same environment that is used for course exercises, and include all the necessary tools for the exam, such as IDEs, compilers, and libraries.

As course evaluation typically places a heavy emphasis on end-of-course assessment, e.g., course projects or final exams, it's important to ensure the integrity of the examination. To achieve this, there are two primary requirements that need to be considered; (1) The image needs to be reset after every examination session in order to prevent students from leaving information behind that can compromise the integrity of the exam, and (2) the virtual machines need to have several access restrictions in place. For example, the student may not access the internet or other devices in the same local area network (LAN). The students should also have restricted permissions, allowing writing and execution only in pre-determined directories. Full list of requirements is shown in Table 10.

Table 10. Case: Examination environment, primary requirements.

ID	Requirement	Description
ER-1	Supported host OSs	Windows 7 or newer
ER-2	Image OS	Ubuntu 19.04 or newer Windows 7 or newer
ER-4	GUI support	GUI support is required
ER-5	Restricted write/execute permissions	Writing and execution is only allowed in pre-determined directories
ER-6	Blank image	Image needs to reset after each examination session.
ER-7	Restricted internet access	No access to internet

4.2 Experimentation

4.2.1 Docker containers

In many cases containers are considered the future of virtualization, especially when it comes to deploying microservice applications or reducing infrastructure costs. In this chapter a container-based approach to implementing a sandbox environment for LUT university's software engineering course *Operating systems and systems programming* is detailed. The purpose of the virtual environment is to allow students to work on course exercises in a

uniform environment that is the same for each student and teaching staff. The virtual environment provides the student with all the necessary tools and course material that are needed to complete the exercises. The advantages of docker are the ease of deployment and portability that it offers with its light-weight images and image library called *Docker Hub*.

Images that function as the base for the created containers were selected based on the use case requirements listed under chapters *4.1.1 Case: Classroom* and *4.1.2 Case: Examination environment*. The primary considerations were the supported operating systems (CR-1, ER-1) and the support for GUI (CR-4, ER-4). For all the courses, the primary supported OS is Windows 7 or newer version of windows. While support for other OSs (Linux and macOS) is preferred, they are secondary in priority. As containers share the host OS kernel, if one wants to host a container using a different OS, the assistance of a hypervisor is needed, which increases to computational overhead needed to run the container. For example, it's possible to run a Linux container on Windows 10 using Windows Subsystem for Linux 2 (WSL2), which is based on Microsoft Hyper-V architecture (Loewen et al., 2022). The support for GUI is necessary for both classroom and examination environments as majority of the exercises and tasks require some sort of graphical representation. For example, if the task is to write and run code within the container, the support for a GUI enabled text editor is needed. While it's possible to use CLI-based editors such as Vim or GNU nano, they come at an additional cost as they are not as accessible and user-friendly for majority of the students.

Two primary types of images were explored: minimal Ubuntu command line interface (CLI) images and Ubuntu graphical user interface (GUI) images. While one of the primary requirements was support for GUI, CLI options are explored as a future option for courses such as *Basics of Linux* that mainly use the command line for course exercises – granted, this is only possible if larger scale course reform is conducted. The used images and their information can be seen in the Table 11 below. In the experimentation Docker version 20.10.11 is used.

Table 11. Used Docker images.

Image name	Size	Version	Digest
Ubuntu	72.78 MB	20.04	7c9c7fed23de
dorowu/ubuntu-desktop-lxde-vnc	483.12 MB	n.d.	775184eaa5bb
alpine/git	13.65 MB	2.32.0	4812deaa1070
phusion/baseimage	81.09 MB	n.d.	7621c55af729

Ubuntu

The *Ubuntu*-image is Docker's official image for Ubuntu distribution of Linux OS. It differs from the standard full image in that much of its functionality and features have been completely dropped or reduced to minimize the resulting image size. The container is interactable through a CLI as seen in Figure 4. The CLI supports most of the basic shell commands such as ls, echo, mv, mkdir, etc. As mentioned, it does however have limited functionality when it comes to many of your typically pre-installed software such as gedit text editor; and while it's possible to manually install the missing software that is normally included with the distribution, they may operate differently, if at all, within the container.

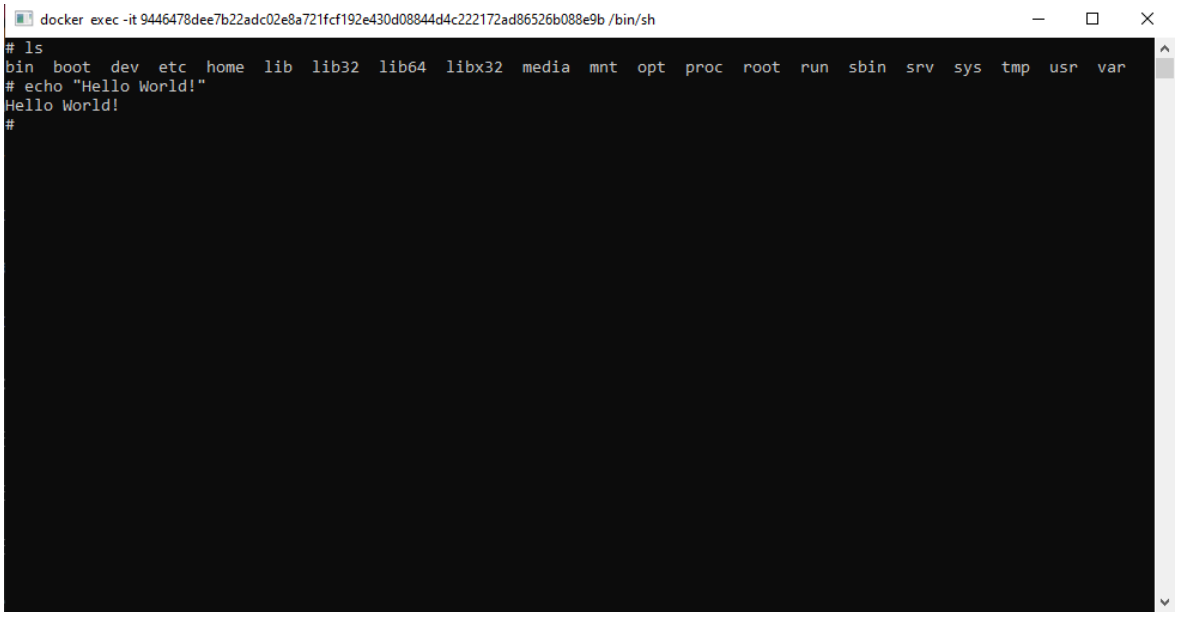

Figure 4. Docker Ubuntu CLI.

Alpine Linux

Alpine Linux is a security-oriented very light-weight Linux distribution, which is often used for server services (Alpine Linux Development Team, 2022a). The image was mainly explored as a super light-weight alternative for use cases, where fast boot-up times and small image sizes are vital, e.g., due to file hosting size limitations. Due to its small size, it's missing a lot of the packages included in other distributions, as the packages have either been removed completely or replaced by lighter alternatives. For example, in Alpine Linux' latest release 3.15.0, *sudo* was replaced by *doas* as the primary privilege escalation tool, which a lighter alternative to *sudo* (Alpine Linux Development Team, 2022b). Due to the image's emphasis on security, it's also a fitting candidate for the thesis' examination environment use case: *4.1.2 Case: Examination environment*. The image's limitations are largely the same as with Docker's Ubuntu image, i.e., missing software and packages, and lack of GUI support.

Phusion baseimage

Phusion baseimage is largely similar to the Docker's official Ubuntu image. Like the official image, Phusion baseimage only contains the essential components that are needed to run the operating system within a container. The image is based on Ubuntu 16.04 LTS, which means some of the features of newer Ubuntu releases are missing. For example, Ubuntu 16.04 is still shipping with Python 2, while Ubuntu 20.4 comes with Python 3. The same applies to many other libraries and binaries, which may result in compatibility issues with third party software. Ubuntu 16.04 reached its end of life for maintenance and hardware updates in April 2021; however, it's still receiving extended security maintenance (ESM) until 2026. The main difference between the two is that Phusion baseimage has been customized to run better inside a docker a container, resulting in faster builds and easier deployment. (phusion, 2021; Ubuntu, 2022) While the focus of the image is in software development where there is a need to build frequently, for the purpose of this thesis, Phusion baseimage is explored as an alternative to the Docker's official ubuntu image.

Ubuntu Virtual Networking Computer

dorowu/ubuntu-desktop-lxde-vnc is a Docker image designed to enable Ubuntu desktop environment with fully functional GUI through Virtual Networking Computing (VNC) interface. VNC enables remote connection to the virtual computer running within the Docker container. While VNC is typically used to allow remote connection to a device over a network, for example for the purpose of sharing computing resources, in this case it's applied to display the desktop of the guest OS hosted locally. Once the docker container is running, it can be accessed either through a web browser or VNC client using the localhost Internet Protocol (IP) address and designated port. *dorowu/ubuntu-desktop-lxde-vnc* is explored as an alternative to VMs that otherwise offers similar functionality as a VM, including a fully interactive desktop environment that isn't natively supported by Docker containers.

The caveats of this approach are the inevitably higher latency that is introduced by remote connection and differences in the desktop GUI appearance and functionality as shown in Figure 5. The desktop appearance is arguably a little rough around the edges, and while it can be a noteworthy option for applications where only minimal GUI is needed, it is unlikely to be suitable for this thesis' use cases as detailed in chapter *4.1 Use cases*.

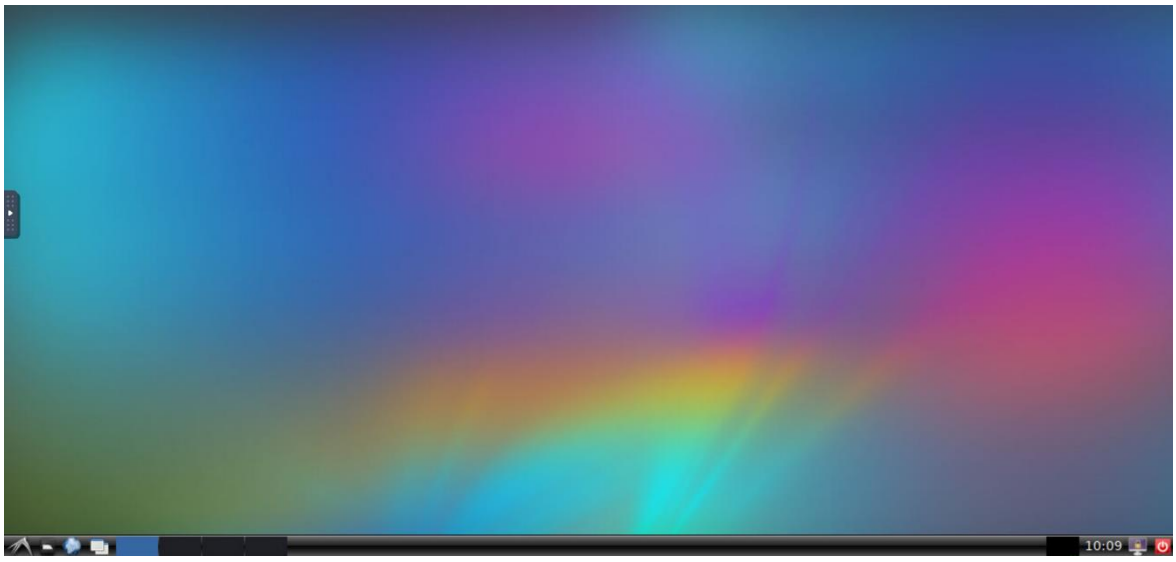

Figure 5. dorowu/ubuntu-desktop-lxde-vnc, desktop interface.

4.2.2 Virtual machine

In this thesis, virtual machines (VMs) are explored as an alternative approach to classroom virtualization. While containers are the light-weight and more modern approach to virtualization, VMs are the time-tested candidate for full OS virtualization. VMs come with the caveat of larger file size and system resource utilization, however virtualization of full OS ensures that the environment is as close to authentic as possible through emulation. Unlike Docker containers, VMs can display OSs graphical user interface (GUI) without extensive configuration, e.g., without incorporating virtual network computing (VNC) interface like we did using *dorowu/ubuntu-desktop-lxde-vnc*-image in chapter *4.2.1 Docker containers*. The presence of GUI and native desktop environment, such as GNOME or Unity desktops for Ubuntu, are vital for education of some core concepts such as basic navigation and file management, both through command line interface (CLI) and GUI. For this reason, it's still too early to adopt purely CLI supported options such as Docker containers. While for certain courses, such as *Basics of Linux*, which is largely a CLI focused course aiming to teach basic Linux shell commands as detailed in chapter *4.1.1 Case: Classroom*, is able to convey most of the core concepts even without a GUI, at least for now the GUI is still a big part of the course implementation.

The tools used to both deploy and host the virtual machines are mainly chosen based on the virtualization software's capabilities, supported OSs and active LUT-university licenses. LUT university advocates the use of VMware Workstation Player for virtualization purposes and offers student licenses for other paid software such as VMware fusion. The elicited requirement CR-1 also states that the minimal requirement for host OS support is Windows, while support for Linux and MacOS are preferred but optional. For aforementioned reasons, in this thesis, VMware Workstation Player 16.2.2 is used for VM hosting and customization purposes. Due to limitations of the free version of VMware Workstation Player, VMware Workstation Pro is used to export a custom image in Open Virtualization Format (OVF). The full list of used software and images is detailed in Table 12.

Table 12. Virtual machine, used software and images.

Software / Image	**Version**
VMware Workstation Player	16.2.2
VMware Workstation 16 Pro	16.2.2
Ubuntu	20.04.4 LTS

While the long-term plan, as detailed under chapter *4.1 Use cases*, is to create tailored VM images for each of the individual use cases, i.e., for three university offered courses *Operating systems and systems programming*, *Basics of Linux*, and *Basics of C programming*, and for the university *EXAM* examination environment, in this thesis the focus is on creating a proof of concept for a single course *Operating systems and systems programming*. The first step in creation of the VM is selection of the base image, which in this case is determined by the requirements imposed by the course detailed in Table 9. For the course *Operating systems and systems programming* the preferred OS is Ubuntu 19.04 or newer. Leaning on the requirements and the current Ubuntu releases at the time of this thesis, the selected base image is Ubuntu 20.04.4 LTS.

The primary consideration in the building of the artefact is to provide students both ease of access to course material and a pre-configured sandbox for course related tasks. To achieve these goals, there are three primary steps taken concerning VM customization, which are (1) system configuration, (2) software deployment, and (3) organizing course material. The purpose of these steps is to eliminate any additional configuration that is typically needed before the student can start working on the course exercises. System configuration includes activities such as installing system updates, language and keyboard configuration, alias creation, etc. The second and third steps are focused on deploying both software and material needed to complete the course exercises. These activities include for example the updating and installation of necessary libraries, and creation of folder structure that supports learning.

4.3 Data collection and data analysis method

In this thesis, survey was selected as the primary data collection method. Each survey method has its pros and cons, and as such it's important to choose a method that fits one's

own research area. For example, if there is a need to extrapolate the data to a larger population, a probability sample is needed; however, if time is a limiting factor and the research is more exploratory in nature, a non-probability sample can be sufficient. (Sue and Ritter, 2012) For this thesis, the survey population is the students who have enrolled on LUT offered course *Operating systems and systems programming*. Out of the population, the sample is selected on an opt-in basis, which means the students themselves choose whether they wish to participate in the survey. The population consists primarily of software engineering students, which means that the sample is largely homogeneous. Everything considered, the sample can be classified as a non-probability convenience sample, and as such in data analysis we'll focus on descriptive statistics instead of inferential statistics.

In this thesis, the survey will be conducted as an online survey. The primary advantage of online surveys is in its speed, scalability, and low cost. One of the major concerns of online surveys is generalizability of the data, however as previously discussed, due to the homogeneity and size of the sample, reduced generalizability of the data is inevitable. (Schonlau and Couper, 2017) Besides convenience, online surveys also reduce inaccuracies in the survey data caused by social or political correctness. In a face-to-face interview the interviewee's answer can often be influenced by what is socially acceptable. In an online survey the interviewee doesn't face the same pressure to give the "right" answer. (Sue and Ritter, 2012) The survey is created using *Webropol 3.0* survey tool and administered as a part of the course's end-of-course survey.

In this thesis, the survey process is largely based on the guidelines of Sue and Ritter in their book *Conducting Online Surveys*. Following their methodology, the survey consists of five steps: (1) sampling, (2) writing survey questions, (3) developing the survey instrument, (3) conducting the survey, (4) processing and analyzing the survey data, and (5) reporting the survey results. In this chapter we will be focusing on two steps, which are *survey planning and survey creation*, and *survey administration*. The survey research process as proposed by Sue and Ritter is shown in Figure 6. In the figure, the steps that are relevant for this chapter are indicated by darker background. The two last steps *data analysis* and *report survey results*, which are indicated by a dashed border, are detailed in greater detail in chapter *5.2.1 Survey*. The rest of the boxes are steps that have either been completed previously or are completely left out as irrelevant for this study. For example, literature review was previously

covered in chapter *2 LITERATURE REVIEW*, survey objectives are guided by research questions set in chapter *3.2 Research questions*, and sampling option evaluation is omitted, since the sample is pre-determined. More information on the survey sample is given in chapter *5.2.1 Survey*.

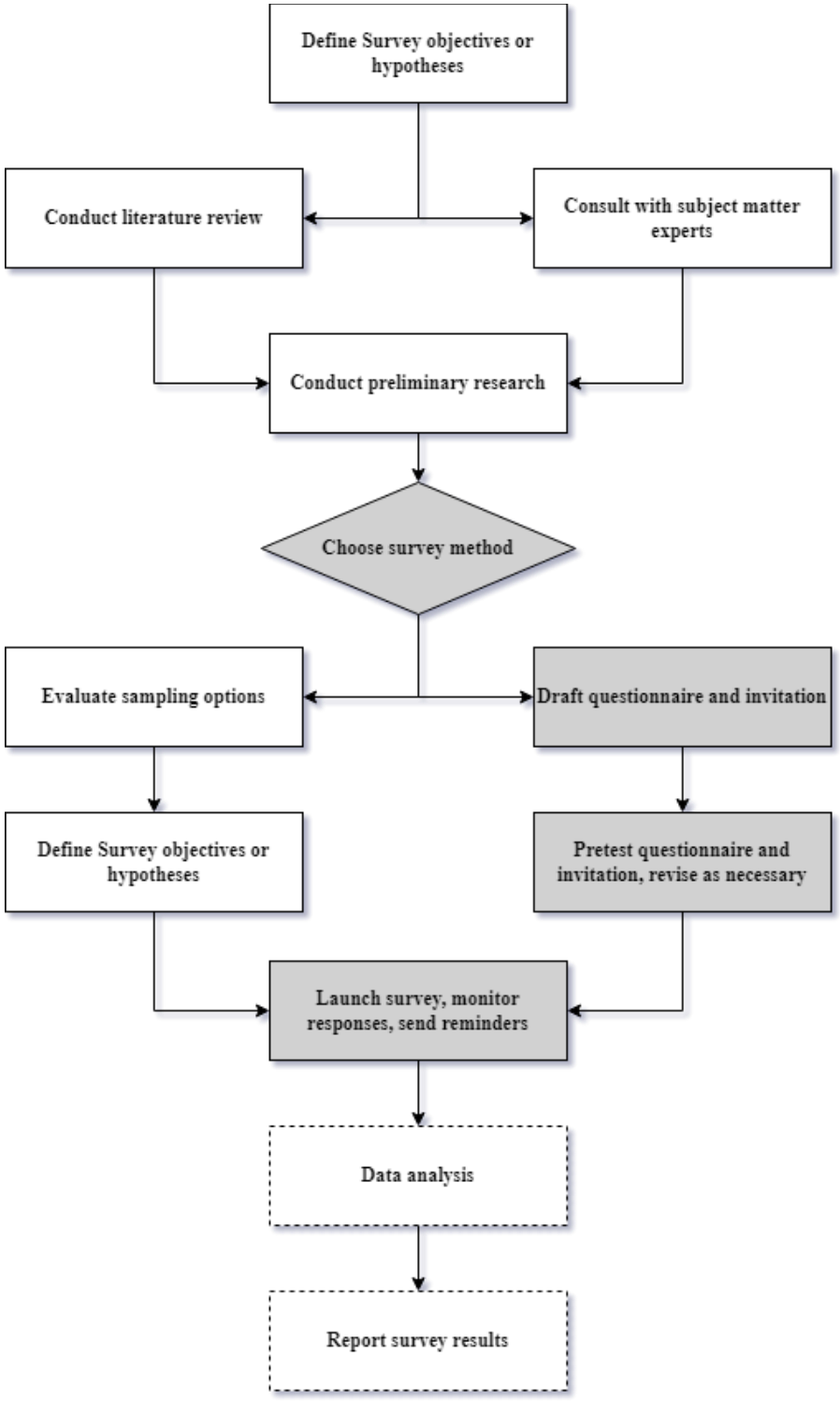

Figure 6. Survey research process (Sue and Ritter, 2012).

In their book, Sue and Ritter detail general guidelines for writing survey questions. In this thesis, these guidelines are used to guide survey question formulation. The following 8 points adapted from Sue and Ritter's book are the primary considerations when writing survey questions for this thesis:

1. Every question should be related to the survey's objectives (research questions).
2. Keep the survey concise.
3. Formulate questions in such manner that they are specific, easy to understand, and written in complete sentences.
4. Avoid leading questions.
5. Keep the response options consistent across the survey.
6. Use open-ended questions when the list of response options would otherwise be too long or when exploring unfamiliar topics.
7. Close-ended questions should cover every possible answer and the options should be mutually exclusive.
8. Choose the question scale (e.g., 10-point scale or 5-point scale) based on the survey's purpose. For example, if more granular options are necessary for more accurate results, a 10- or 11-point scale might be preferred. If the emphasis is on ease of analysis and making the questions easier for the respondent to answer, a 5-point, or even 3-point scale can be sufficient.

While the above rules are used as a guideline in formulating survey questions, they are not absolute. For example, while some recommend that "no opinion"-option should be avoided, in the survey the option is included when it's unclear whether the respondent has enough information to answer the question. In this way, instead of forcing the respondent to give an erroneous answer, we provide the respondent with an option to opt-out from answering. The survey consists of two primary sets of questions, which are determined based on an initial contingency question (Appendix 1, Question 5), i.e., the decision whether the respondent is allocated questions in set A or set B depends on their answer to the contingency question.

The survey questions are a mixture of statistical and qualitative questions. For this reason, the survey responses are analyzed using both statistical and qualitative methods, depending

on the type of question. The statistical question response options are given as a 5-point Likert scale, and they are each given a numerical value. First the responses are also tallied and reported as percentages to provide a better, more legible, overview of the responses. The responses are then evaluated using inferential statistics such as sample mean and median. Lastly, correlation between individual categories is inferred from the available data, e.g., finding out whether age or education level affects one's proficiency with virtualization software. For the qualitative questions, the results are interpreted using content analysis. The content analysis includes steps such as familiarizing with the subject matter and organizing the data. As a part of content analysis, the researcher reads through the responses and attempts to identify common themes between the responses. This process of deductive analysis starts with preselected themes and the themes are expanded as new ones emerge during the analysis. (Fink, 2003)

5 RESULTS

In this section, the results for the Docker container and VM implementation are presented. The section also details the results of the conducted survey as the primary data collection method. Due to the nature of the sample, the results aren't fit for statistical analysis. Instead, in this section the survey results are presented and dissected through descriptive statistics. Furthermore, the collected data is further analyzed to infer meaning and connections from the underlying data.

5.1 Docker

In this thesis Docker was explored as a light-weight modern approach to virtualization. The experimentation consists of four Docker base images: (1) *Ubuntu*, (2) *dorowu/ubuntu-desktop-lxde-vnc*, (3) *alpine/git,* and (4) *phusion/baseimage*. Each of the images was built and tested as a stand-alone Docker container. More specific information on the used images and software is listed under chapter 4.2.1 Docker containers.

The results were largely as expected; image sizes being under 100 MB, apart from *dorowu/ubuntu-desktop-lxde-vnc* -image, the Docker containers are very fast to deploy with the start-up times being significantly faster than that of VMs. The docker desktop application is also easy to use and shouldn't cause problems even for an inexperienced user. To deploy a container, one only needs to download an image from Docker Hub and launch the container either through the desktop application GUI or the CLI. While Docker containers operated within the scope of limitations that were observed during the planning phase, the containers' inability to display GUI was largely overlooked. Based on the experimentation it quickly became apparent that Docker isn't suitable for full OS virtualization. For example, deploying a Linux-based container on a Windows-based host OS uses WSL or WSL 2, which introduces additional overhead compared to a container that shares the host OS. Moreover, vast majority of the publicly distributed container images have been configured to be minimalistic, meaning that if one were to enable the full functionality of Ubuntu environment, the resulting image size would be similar to that of a VM.

Containers based on *Ubuntu, alpine/git,*and *phusion/baseimage* produced similar results. The containers boasted a CLI capable of understanding the basic shell commands but were

lacking when it came to many of the ubuntu native applications. For example, due to the lack of functional GUI the usage of graphical file managers such as Nautilus or Nemo is not possible. This also applies to applications that utilize graphical elements such as Ubuntu's default text editor Gedit. As the support for GUI isn't compromisable, as shown by the initial elicited requirements listed in Table 9 and Table 10, *dorowu/ubuntu-desktop-lxde-vnc* - image was explored as an option to GUI enabled containers. As the name suggests, the GUI is enabled by Lightweight X11 Desktop Environment (LXDE) and Virtual Network Computing (VNC). While the approach sounds good in theory, in practice it's not quite as refined as one might expect; while the LXDE provides a light desktop environment, it's limited in terms of its functionality and aesthetics. Additionally, as the desktop is accessed over the network using VNC, a fair amount of latency is introduced in the process. Another problem of containers is the data persistence. By default, Docker resets the image to its original state when the container is shut down. As a solution Docker offers the ability to either create a volume or a bind mount, out of which volumes are the Docker's recommended option for data persistence. Based on the experimentation, the Docker containers are recommended for applications where GUI and persistent data isn't needed, such as hosting a web server or testing applications.

The evaluation of containers was conducted by comparing the created artefacts against the requirements for each use case presented in chapters 4.1.1 Case: Classroom and 4.1.2 Case: Examination environment, with the focus being on classroom use case, since the final artefact will be evaluated and tested by students participating in university offered course *Operating systems and systems programming*. Upon expert evaluation and through comparison against the requirements, namely requirements CR-4 and ER-4, Docker containers were deemed unfitting for this thesis' use cases. While the presence of GUI isn't a necessity for every university offered course, this consideration is left for future work.

5.2 Virtual machine

As a more traditional means of virtualization, VMs were explored as an option to provide the students with a tailored virtualized environment for course work. For this purpose, a VM was created with Ubuntu as the base-image. The information on used software and images is explained in greater detail under the chapter *4.2.2 Virtual Machine*. As mentioned in the

chapter *4.2.2 Virtual Machine*, the virtual machine was created and customized using VMware Workstation Player and later exported in OVF format, which was then distributed to the students by making it available on a student accessible cloud storage. The testing and evaluation of the artefact, i.e., the tailored VM, was implemented as an elective course project. As a part of the project the students were given installation and deployment instructions to minimize potential issues that can arise when setting up the environment. The students were also instructed to complete course tasks using the given VM image and record potential issues and other observations arisen during the testing. The final evaluation of the artefact is conducted as a part of the course evaluation survey, which is given to the students at the end of the course. In the survey the students were asked to report their experiences with the VM by answering a set of written questions. The full set of questions can be seen in Appendices 1 and 2, and the results of said survey are presented in next chapter *5.2.1 Survey*.

The final image size of the image being 6.69 GB, the image can be considered to be sizeable. The large size of the image places some limits on the potential image distribution channels; as an example, Moodle, a popular open-source Learning Management System (LMS), has a very limited maximum file size for uploaded files with the limit typically being somewhere below 300 MB. While the upper limit can be increased by the service provider, in most cases this isn't a viable course of action. Even most of the commercial file hosting services place heavy restrictions on free accounts on the maximum allowed file size and storage space. In our implementation OneDrive was used for the distribution of the OVF package as LUT university owns an active license for Office 365 products. While the image was created for one specific course, *Operating systems and systems programming*, it can be easily expanded to include multiple courses. By adding other courses' dependencies, the resulting image will inevitably grow in size. arguments can be made both for and against such approach; while one should avoid creating overly bloated systems, in many cases it's more convenient for the student to have all the tools in one place.

5.2.1 Survey

The survey, as described in chapter 4.3 *Data collection and data analysis method*, was conducted as an opt-in-based online survey and it focused on evaluating one test artefact, a customized VM environment, which was deployed on university offered course *Operating systems and systems programming*. The survey's sample size was limited by two primary factors: (1) scope of survey administration and (2) freedom of choice. The scope of the survey refers to the scale on which the survey was carried out, i.e., how many courses were used to collect the data, and over how long of a time period the data was collected. In this case, the survey was limited to a single course *Operating systems and systems programming*, and the data was administered over a period of two weeks. The second factor, the freedom of choice, applies to both course project selection and survey participation; the students were free to select any of the given course project topics, the VM testing related project being only one possible option. Further restricting the number of responses, the end-of-course -survey isn't compulsory and typically yields a response rate of around less than half of the total course participants. The sample being a nonprobability convenience sample, the data obtained through the survey isn't a good fit for statistical analysis. Furthermore, since statistical significance is directly tied to the number of responses, with the small sample size of 26 respondents, it's hard to get any kind of statistically meaningful result. That said, in this chapter we'll focus on visualizing the survey results, infer key values from the data, and analyze underlying connections and meaning from the data.

The survey questions are divided into two primary categories, which are demographic questions and opinion-based questions. The opinion-based questions are a mixture of categorical -, dichotomous -, contingency -, and Likert scale questions, depending on the question's objective. For example, dichotomous questions are used when a simple binary "yes" or "no" answer is sufficient. In some cases, a "no opinion" -option is added if it can be assumed that the respondent may have insufficient foundation to answer the question. On the other hand, categorical -, 5-point Likert scale -, and contingency questions are used when the objective is to gather more granular feedback on a specific subject or to direct survey flow. The demographic questions are used to both identify the respondents' background and potential correlations within the data. In total there are four demographic questions administered at the start of the survey, which aim to collect information such as respondent

gender, age, education, and IT capabilities. The full description of the demographic questions and the complete survey can be seen in Appendix 1.

Over the period of two weeks, the survey was taken by a total of 26 students. Figure 7 shows the respondents' gender distribution; out of the 26 respondents 21 were men and 5 were women. Meaning that the vast majority, 81% of the respondents, were men, while only 19% were women. While it's hard to determine the exact reason for the large difference in attendance between men and women, a big part of it can likely be attributed to the course being a software engineering course – software engineering historically being a field that is highly male dominated. The respondent ages largely fall into one dominant group, ages 20-25, as seen in Figure 8. Those aged 20-25 correspond to 85% of respondents, while only 12% of the respondents are aged between 26-30, and 4% are more than 36 years old. The students were also asked about their education and IT capabilities. These results are visualized in Figure 9 and Figure 10. According to the responses 85% of the students reported their highest attained degree as upper secondary education, 8% polytechnic, and 4% master's degree. As the evaluation of the respondents' IT capabilities was conducted by the respondents themselves through introspection, the results should be taken with a grain of salt. The respondents were asked to evaluate their IT capabilities on a scale from 1-5, where 1 is nonexistent, 2 is weak, 3 is average, 4 is good, and 5 is excellent. According to the survey results, the respondents evaluate their IT capabilities high, with the average score being 3.8 with a median of 4 (good).

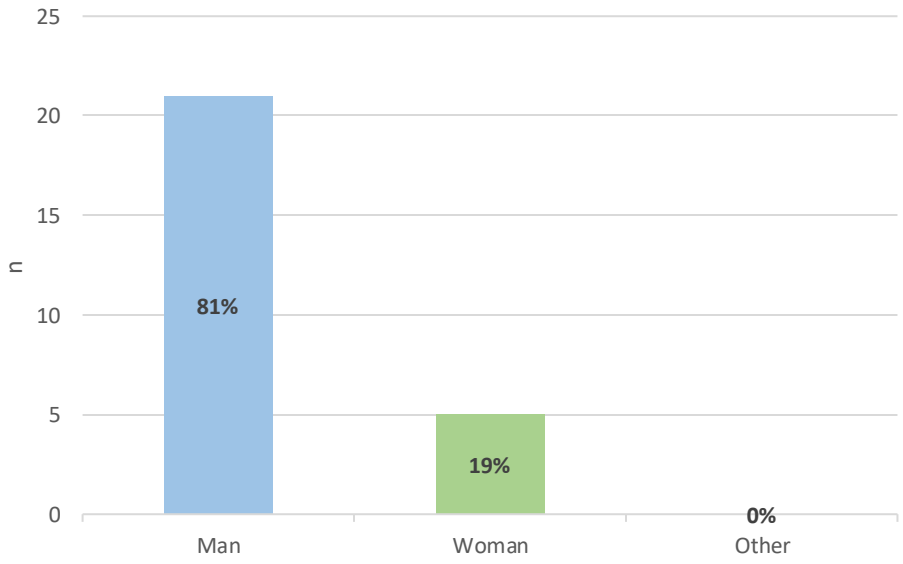

Figure 7. Respondent gender.

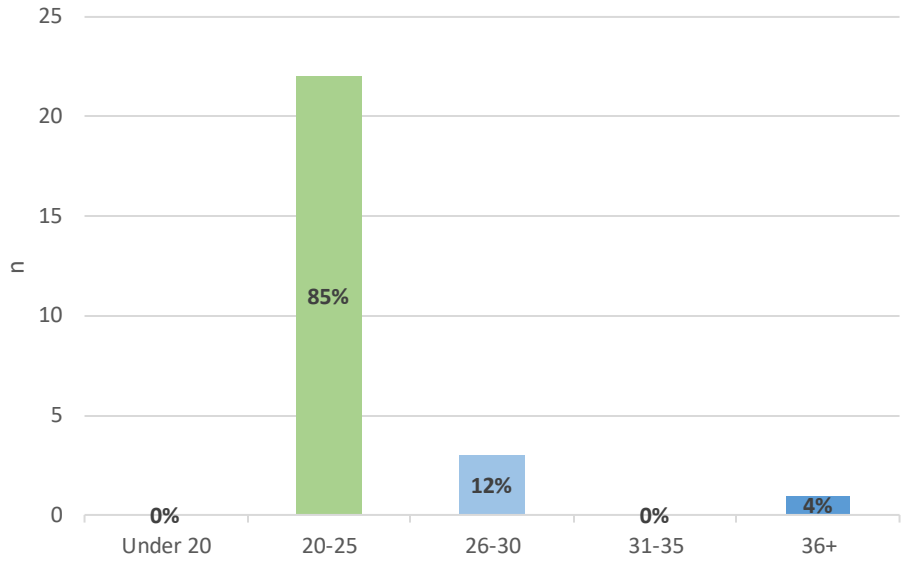

Figure 8. Respondent age distribution.

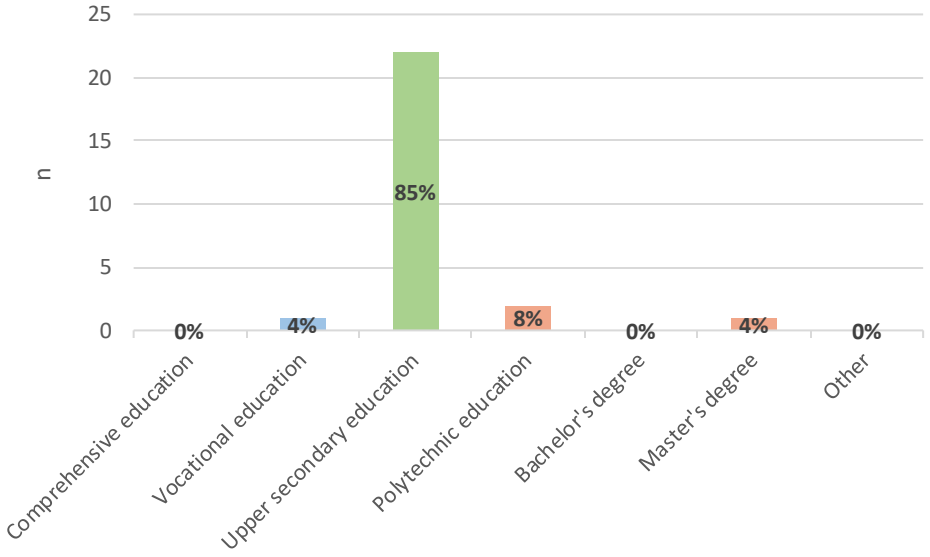

Figure 9. Respondents' reported highest attained degree.

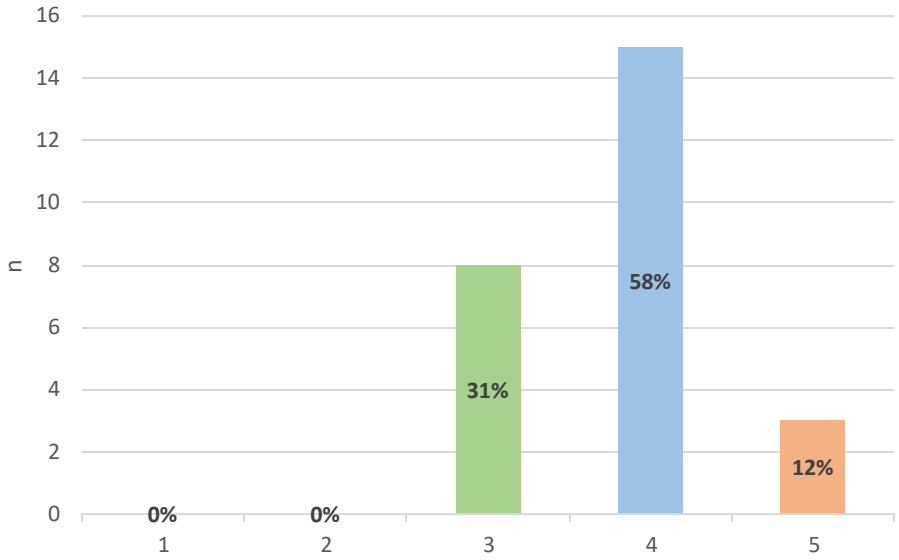

Figure 10. Respondents' reported IT capabilities.

To direct survey flow, a contingency question (Appendix 1, Question 5) was used. The students were asked to report whether they had used the artefact, and the survey flow was directed based on their response. According to the results shown in Figure 11, 27% of the respondents reported having tested the artefact, while 73% reported having abstained. Those that abstained were asked to report their motivations for doing so; based on the responses, the most common reason for picking a different project was personal preference. 46% of those that abstained reported that the project simply didn't interest them. Second largest group, at 29% of the respondents, reported that the project was too challenging, and 4% of the respondents chose a different project as their personal computers didn't support virtualization. While only 4% reported that these technical requirements were the primary motivation for choosing a different project, the open feedback shows that the actual number might be higher as multiple respondents expressed having encountered issues with virtualization, mainly due to conflicts with xv6 or macOS. Lastly, 21% of the respondents reported having chosen a different project for other reasons that weren't included in the categorical question. Those that selected "other" were asked to further elaborate on their choice. From the open-ended responses, two main themes were identified, which were: (1) personal preference, and (2) workload. While the identified themes are largely similar to the existing categories, their emergence is likely due to differences in interpretation of the given categories. The responses indicated that while some perceived the project as being easy,

others found the project too challenging, particularly due to the workload, and consequently led to them selecting a different project.

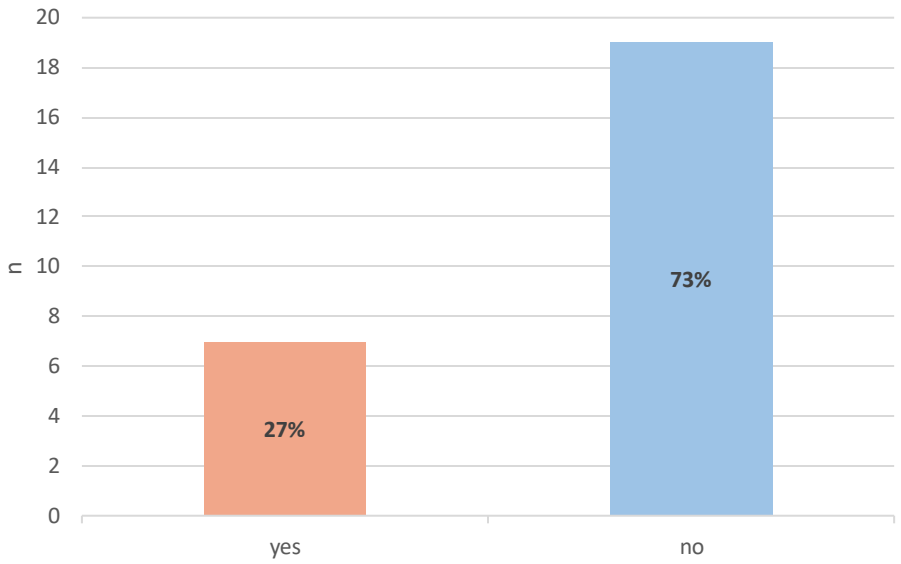

Figure 11. Proportion of respondents who reported having tested the tailored VM.

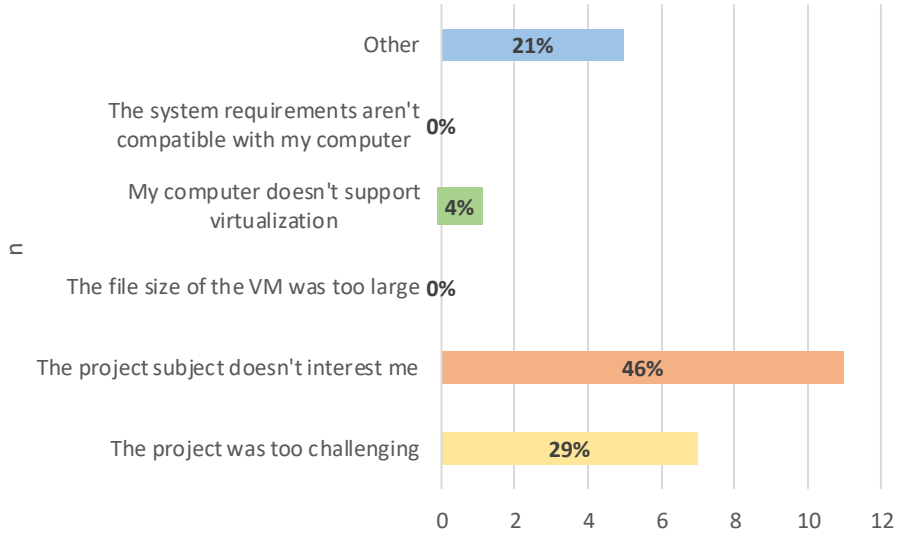

Figure 12. Respondents' reported reason for picking another project.

Both groups, those who tested the artefact and those that didn't, were asked if they'd see a tailored VM environment being beneficial for other university courses. The groups' combined responses indicate that similar tailored VM environments would be positively received. Out of the 27 responses 33% indicated that they could see a similar VM environment being beneficial, a majority of the responses, 67%, were indifferent, and not a

single respondent saw no benefits to a tailored VM environment. Based on the open feedback, those that saw benefit to VM environments indicated that said environment could be especially useful for programming- and Linux-oriented courses. While many of the programming courses already use VMs or other subsystems such as WSL/WSL2, these environments still often require extensive configuration from the user, and a tailored VM has the potential to deliver necessary tools with significantly less friction.

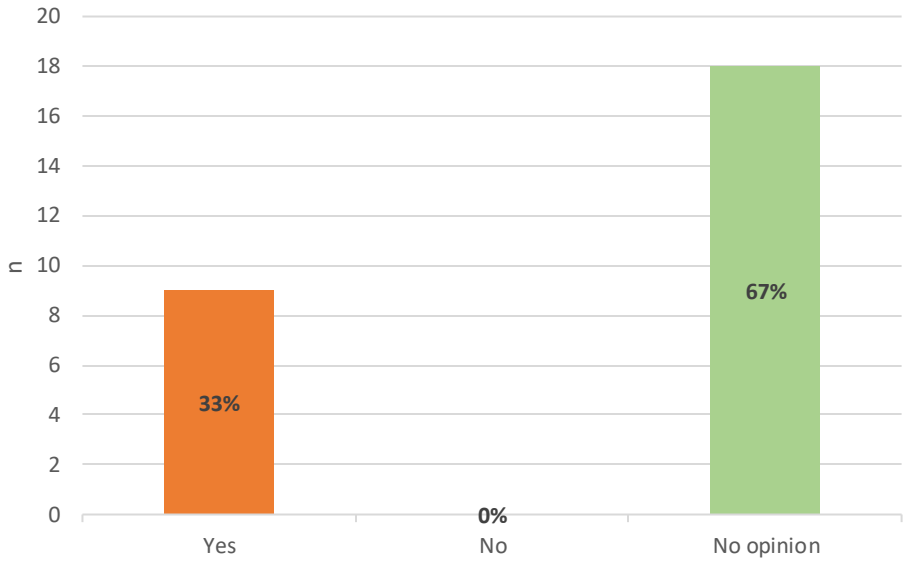

Figure 13. Proportion of respondents who consider the tailored VM to be beneficial.

To gain a better understanding of the demographic's technical prowess, the respondents were asked to evaluate their prior experience with VMs on a scale from 1 to 5, where 1 is nonexistent, 2 is very low, 3 is fairly low, 4 is fairly high, and 5 is very high. These results are visualized in Figure 14. Out of the responses 71% indicated that their prior experience with VMs is low. The remaining 29% indicated that they have fairly high experience with VMs. With virtualization being a common theme on software engineering courses, it doesn't come as a big surprise that none of the respondents reported having either nonexistent or very low experience with VMs. It should be also noted that the collected responses only include responses from students that selected testing the artefact as their final project. The project being VM focused, it can be assumed that the project topic would be more attractive to those with prior experience on the topic, which would be consequently reflected in the survey results. Adding to the evaluation of the respondents' previous experience, the respondents were asked if they had any prior experience with Ubuntu or any other Linux

distribution. According to the results as shown in Figure 15, the vast majority, 86% of the respondents, reported having used Ubuntu or other Linux distribution. Only 14% of the respondents reported having no prior experience with any Linux distributions.

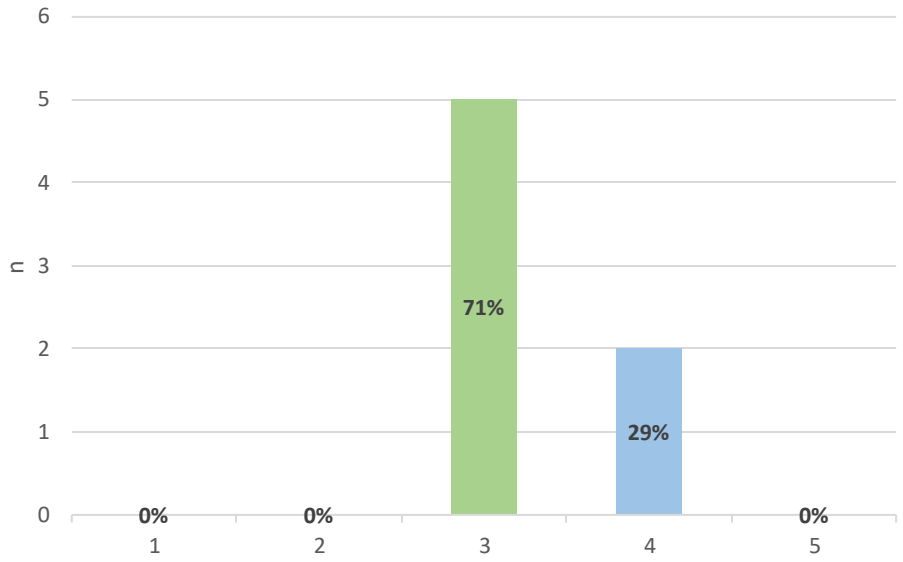

Figure 14. Respondents' reported prior experience with VMs.

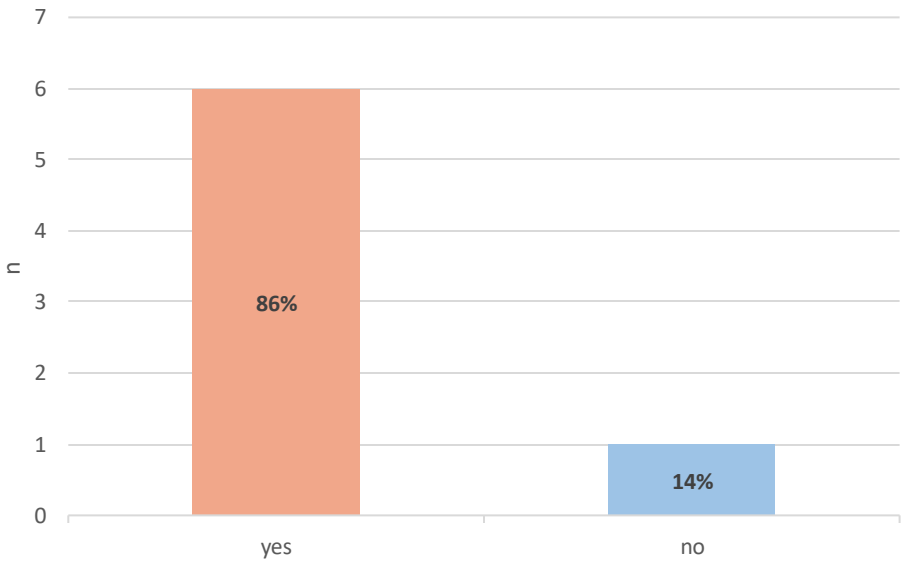

Figure 15. Portion of respondents that reported having used ubuntu or other Linux distribution before.

For artefact evaluation, the respondents were given eight statements. The respondents were asked to rate each statement on a scale from 1 to 5, depending on their position towards the statement. The answers are given on a 5-point Likert scale, where 1 is *completely disagree*,

2 is *somewhat disagree*, 3 is *no opinion*, 4 is *somewhat agree*, and 5 is *completely agree*. The statements S1-S8 and their average scores are shown in Table 13 and Figure 17. Generally, a higher score for a given statement is better, and this holds true for every statement except the statement S6, which, unlike other statements, is formulated as a "negative" statement instead of carrying a positive connotation. For this statement, inversely a lower score is preferred. Excluding statement S6, the statements hold a combined average score of 4.5. Out of all the statements, statement S3, which concerns VM installation and deployment, received the highest average score of 5, indicating that none of the respondents had issues with setting up the environment – most likely in part due to the installation and deployment instructions provided with the assignment. Additionally, the high combined score of 4.5 indicates that the tailored VM environment was very well received. This conclusion is also supported by the observed median values, which are less affected by outliers in the data than the mean. As mentioned, for statement S6 a lower score is generally better; For statement 6 the average score being 1.7 and the median being 1, it can be inferred that the relatively large file size of the VM, 6.69 GB, wasn't seen as much of an obstacle. Excluding outliers, the median of 1 is the lowest possible value on the given range, which indicates that the respondents completely disagreed with the given statement. This can be speculated to be due to the ever-improving internet speeds and infrastructure; especially for LUT students that are covered by the Lnet-network offered by LOAS student housing organization.

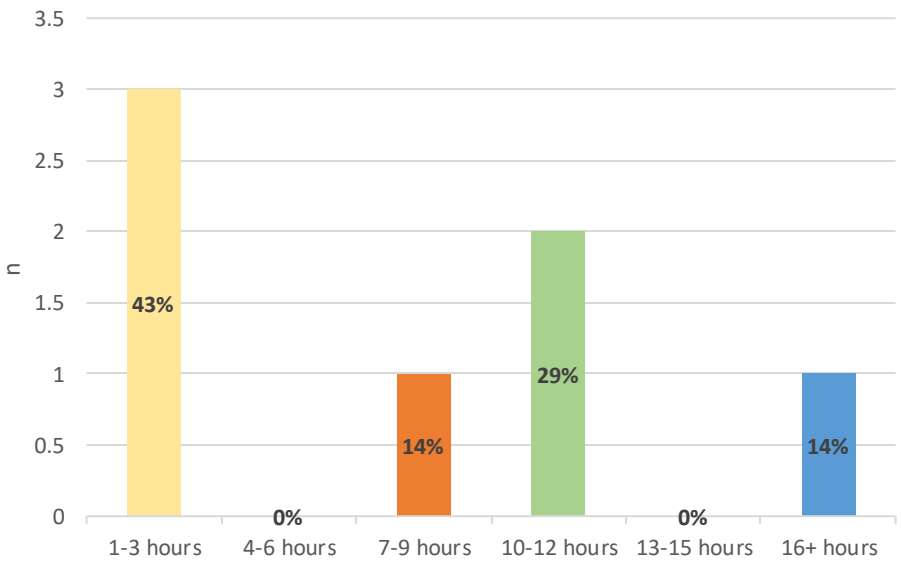

Figure 16. Reported total time invested in testing the VM.

Table 13. Survey statements S1-S8.

Statement	Statement description
S1	The VM aided me in my studies
S2	the VM provided necessary tools for the course tasks
S3	VM installation and deployment was easy
S4	VM usage and task completion was easy
S5	I believe using a tailored VM would benefit other university courses
S6	I found the large file size of the VM to be an inconvenience
S7	The navigation within the VM was easy
S8	Usage of the virtualization tool (Vmware Workstation Player) and the Linux VM was easy to learn

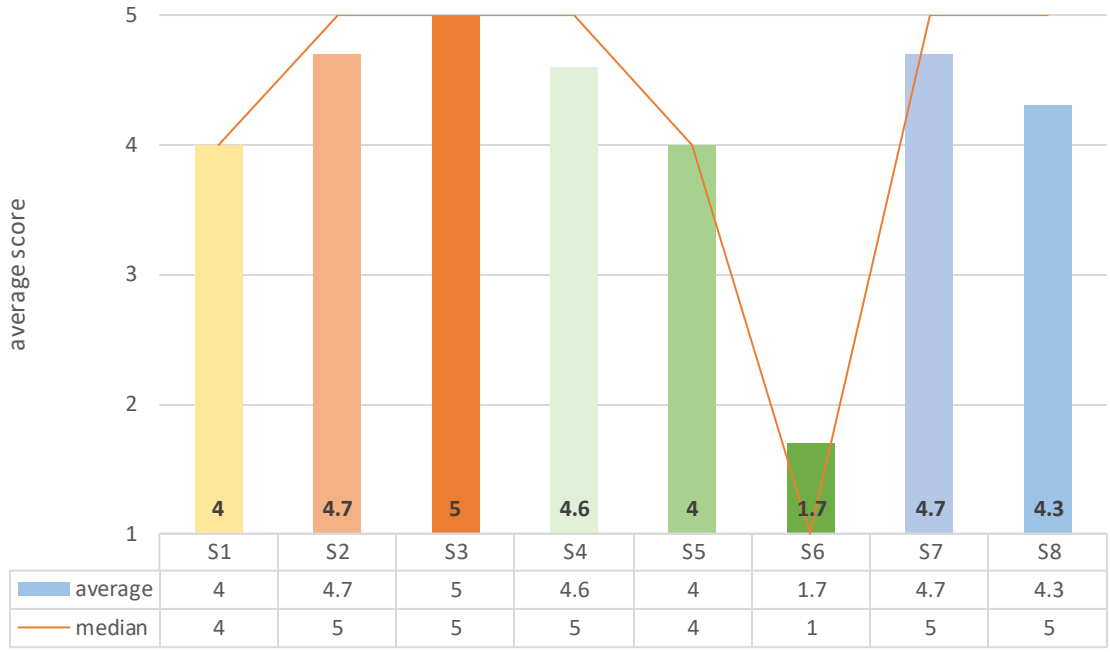

Figure 17. Respondents' evaluation of given statements regarding VM usage and potential.

6 DISCUSSION

According to the survey results, the students that chose testing the tailored VM as their course project and answered the survey, had an overall positive reception of the tailored VM as a tool to aid completion of the course exercises. While most of the respondents were indifferent towards the observed benefits of the VM, not a single respondent saw no benefits to a tailored VM environment. The respondents rated their experience with the installation, deployment, and usage of the VM and virtualization software highly with the lowest average score for a given statement being 4 on a 5-point (1-5) Likert scale. Even the relatively large file size of the VM image, 6.69GB, wasn't seen as a much of a barrier. These results suggest that a tailored VM has potential as a tool to aid remote education, especially in the case of programming-oriented courses, as long as the faculty is willing to invest into implementing course-specific tailored VMs. The survey results show that there is seemingly no link between prior experience, gender, education, or age, and the observed benefit, ease of use, and ease of deployment of the VM, as students of differing backgrounds had largely similar experiences with the VM environment. This suggests that tailored VMs could be used even outside the scope of software engineering, provided there's a need for it such as dependency on an OS or complicated software configurations.

While the results show that there is a clear demand for tailored learning environments, the results should be taken with some reservations; the survey sample being an opt-in-based nonprobability convenience sample, all respondents had prior experience in using VMs, majority had used a Linux distribution before, and every single respondent rated their IT capabilities as average or higher. Since those with prior experience and expertise with VMs are more likely to choose a VM related project, the survey results are likely to be skewed towards that specific demographic, i.e., the subset of students that already have pre-existing qualifications for said project. As the artefact was tested on a software engineering course, even with a completely random sample out of the course population, it would be impossible to completely eliminate sampling bias. For the sake of generalizability, future research should focus on applying the technology to wider variety of courses. That said, the benefits of tailored VMs are likely to be diminished outside the scope of software engineering as other disciplines aren't as reliant on specialized software and environment configurations. Besides bias, another factor affecting the statistical validity of the study is the sample size.

With a sample size of 26 respondents, out of which only 7 reported having tested the artifact, the results are unlikely to hold any statistical significance, especially when generalized to a larger group such as the entire student body. In addition to data collection methods, the future research should focus on building an infrastructure, content delivery system and architecture for higher efficiency and scalability, especially if the goal is to move towards cloud hosting of virtual machines. Cloud computing moves the burden of deployment and computing resources from the student to the faculty. While from the students' standpoint cloud alternatives are preferable due to the ease of access, this approach incurs additional costs in the form of not only building the infrastructure and ensuring sufficient resources, but also maintenance and availability. Therefore, cloud hosting should only be considered if the costs incurred are within one's budget.

Throughout this thesis it became apparent that image modification is challenging as one needs to make the changes inside the virtual machine itself and then export the image through a specialized tool. To aid in this process, a tool specialized in VM image modification could be introduced. While image modification tools weren't explored as a part of this thesis, it's something that should be considered in future research as a way to facilitate faculty adoption of the technology. A specialized tool has the capability to greatly facilitate the image modification process, especially if the implemented changes are small, e.g., adding a file to the image's file system. It should be noted that since the release of Windows 11, WSL 2 has WSLg enabled by default, which allows the execution of graphical Linux applications without VNC. At the time of this thesis, it was only possible to display GUI within a container using VNC, which was the primary reason for the rejection of containerization as a viable option. The possibilities and potential of WSL 2 enabled GUI should be further explored in future research as containers offer a lighter and, in many ways, more seamless virtualization experience.

7 CONCLUSION

The study shows that there is a definite demand for course tailored VM environments. In this thesis, the virtualization approach was selected based on a list of requirements elicited from relevant stakeholders. For this thesis' primary use case, i.e., creating a tailored workspace for course exercises, having a fully functional Linux kernel and GUI is imperative, and as such, at the time of this thesis traditional VMs were the only considerable alternative. While the students reported having faced some technical difficulties related to enabling virtualization and compatibility of the virtualization software, especially on macOS and older devices, the vast majority of the respondents would like to see tailored VMs being used on other university courses as well and saw clear benefits from tailored VMs as a tool to aid remote learning. While due to a small sample size and the survey sample being a nonprobability convenience sample, the results of the survey cannot be generalized to the whole student body, the results can still be used gauge the attitudes of software engineering students.

While traditional full system virtualization was used in this thesis, depending on the use case, cloud-hosting of VMs or usage of unikernels might be preferred over local VMs that are run directly on students' personal computers; Cloud-hosting shifts the burden of environment setup from the student to the faculty, which results in a better user experience. At the same time, cloud-hosting of VMs incurs additional costs for the faculty in the form of physical resources, maintenance, and ensuring availability, making it an unrealistic option for some, depending on the available budget. As traditional VMs run directly on the students' machines, the computational cost is transferred to the end-user, making traditional VMs a more cost-effective alternative from faculty standpoint. On the other hand, if security and lightness are the primary concerns, and running a single application is sufficient for the use case, unikernels stand above the rest. With the recent introduction of WSLg for Windows 11 it was made possible to run Linux GUI applications directly on top a Windows system, while still providing Linux kernel functionality. For this reason, containerization and WSL2 are recommended areas for future research, especially in the context of enabling a lightweight GUI supported Linux environment. Unikernels are another area of research with a lot of potential, however unless one's goal is to build single-threaded microservice-applications

with a minimal footprint, adoption of unikernels as a general-purpose virtualization tool is still too early.

REFERENCES

©UNIR, 2021. UNIR research. Universidad Internacional de la Rioja. URL http://research.unir.net/blog/project-fable-training-faculty-on-blended-learning/ (accessed 4.13.22).

Aggarwal, V., Thangaraju, B., 2020. Performance Analysis of Virtualisation Technologies in NFV and Edge Deployments, in: 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). Presented at the 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), pp. 1–5. https://doi.org/10.1109/CONECCT50063.2020.9198367

Ahmed, M., Sundaram, D., 2011. Design Science Research Methodology: An Artefact-Centric Creation and Evaluation Approach. ACIS 2011 Proc.

Alpine Linux Development Team, 2022a. Alpine Linux [WWW Document]. URL https://wiki.alpinelinux.org/wiki/Main_Page (accessed 2.16.22).

Alpine Linux Development Team, 2022b. Release Notes for Alpine 3.15.0 - Alpine Linux [WWW Document]. URL https://wiki.alpinelinux.org/wiki/Release_Notes_for_Alpine_3.15.0 (accessed 2.16.22).

Andreas Altmann, Bernd Ebersberger, Claudia Mössenlechner, Desiree Wieser, 2019. The Disruptive Power of Online Education : Challenges, Opportunities, Responses. Emerald Publishing Limited, Bingley.

Behravesh, R., Coronado, E., Riggio, R., 2019. Performance Evaluation on Virtualization Technologies for NFV Deployment in 5G Networks, in: 2019 IEEE Conference on Network Softwarization (NetSoft). Presented at the 2019 IEEE Conference on Network Softwarization (NetSoft), pp. 24–29. https://doi.org/10.1109/NETSOFT.2019.8806664

Bruce, J., 2010. Desktop Virtualization in K-12 Schools: Reducing Costs, Saving Time and Delivering Anytime, Anywhere Access for Students and Staff 8.

Butnaru, G.I., Niță, V., Anichiti, A., Brînză, G., 2021. The Effectiveness of Online Education during Covid 19 Pandemic—A Comparative Analysis between the Perceptions of Academic Students and High School Students from Romania. Sustainability 13, 5311. https://doi.org/10.3390/su13095311

Chen, G., 2020. CvLabs: A Container Based Interactive Virtual Lab for IT Education (Technical Report). Georgia Institute of Technology.

Chen, S., Zhou, M., 2021. Evolving Container to Unikernel for Edge Computing and Applications in Process Industry. Processes 9, 18. https://doi.org/10.3390/pr9020351

Childs, M., Soetanto, R., 2017. Online Learning for STEM Subjects: International Examples of Technologies and Pedagogies in Use. Taylor & Francis.

Cooley, S., Boher, S., Ross, E., Wilhite, C., Stanton, A., Gerend, J., Lohr, H., Nikolić, A., Mastrean, A., Lang, P., 2021. Linux Containers on Windows 10 [WWW Document]. URL https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/linux-containers (accessed 12.10.21).

Corbi, A., Burgos, D., 2017. Open Distribution of Virtual Containers as a Key Framework for Open Educational Resources and STEAM Subjects. Electron. J. E-Learn. EJEL 15, 126–136.

De Sordi, J.O., 2021. Design Science Research Methodology: Theory Development from Artifacts. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-82156-2

Docker, 2021. Docker Hub - Container Image Library | Docker [WWW Document]. URL https://www.docker.com/products/docker-hub (accessed 12.16.21).

Farace, D.J., Schöpfel, J. (Eds.), 2010. Grey Literature in Library and Information Studies. K. G. Saur.

Fink, A., 2003. The Survey Handbook. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States of America. https://doi.org/10.4135/9781412986328

Garousi, V., Rainer, A., Felderer, M., Mäntylä, M.V., 2022. Introduction to the Special Issue on: Grey Literature and Multivocal Literature Reviews (MLRs) in software engineering. Inf. Softw. Technol. 141, 106697. https://doi.org/10.1016/j.infsof.2021.106697

Griffin, T.F., Jourdan, Z., 2012. Educational use cases for virtual machines, in: Proceedings of the 50th Annual Southeast Regional Conference, ACM-SE '12. Association for Computing Machinery, New York, NY, USA, pp. 365–366. https://doi.org/10.1145/2184512.2184607

Gusenbauer, M., Haddaway, N.R., 2020. Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. Res. Synth. Methods 11, 181–217. https://doi.org/10.1002/jrsm.1378

Halsey, M., Bettany, A., 2015. Windows Software Compatibility and Hardware Troubleshooting. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-1061-1

Her, P., 2017. Students' Perceptions on Adoption of Desktop Virtualization in Higher Education Environment: A Multiple Case Study (Doctoral Dissertation Manuscript). Northcentral University.

Herselman, M., Botha, A., 2015. Evaluating an Artifact in Design Science Research Methodology as was implemented in a resource constrained environment. https://doi.org/10.1145/2815782.2815806

Hu, H., 2020. Application of Virtual Machine Technology in Computer Teaching. J. Phys. Conf. Ser. 1533. http://dx.doi.org.ezproxy.cc.lut.fi/10.1088/1742-6596/1533/3/032024

IBM Cloud, 2021. Containers vs. Virtual Machines (VMs): What's the Difference? [WWW Document]. URL https://www.ibm.com/cloud/blog/containers-vs-vms (accessed 11.3.21).

IBM Cloud Education, 2019. hypervisors [WWW Document]. URL https://www.ibm.com/cloud/learn/hypervisors (accessed 12.7.21).

Jacqueline Dohaney, Mairéad de Róiste, Rhian A. Salmon, Kathryn Sutherland, 2020. Benefits, barriers, and incentives for improved resilience to disruption in university teaching. Int. J. Disaster Risk Reduct. 50, 101691. https://doi.org/10.1016/j.ijdrr.2020.101691

Johnson, L., Becker, S.A., Cummins, M., Estrada, V., Freeman, A., Hall, C., 2016. NMC Horizon Report: 2016 Higher Education Edition. The New Media Consortium.

Koch, F., Assunção, M.D., Cardonha, C., Netto, M.A.S., 2016. Optimising resource costs of cloud computing for education. Future Gener. Comput. Syst. 55, 473–479. https://doi.org/10.1016/j.future.2015.03.013

Loewen, C., LeMoine, P., v-chmccl, Wojciakowski, M., v-surgos, 2022. FAQ's about Windows Subsystem for Linux [WWW Document]. URL https://docs.microsoft.com/en-us/windows/wsl/faq (accessed 2.16.22).

Loukides, M., Swoyer, S., 2020. Microservices Adoption in 2020 [WWW Document]. O'Reilly Media. URL https://www.oreilly.com/radar/microservices-adoption-in-2020/ (accessed 12.17.21).

Ma, Q., Wang, J., Wang, G., 2014. A Deployment Scheme of Virtual Machines for Campus Cloud Platform, in: 2014 International Conference on Service Sciences. Presented at the 2014 International Conference on Service Sciences, pp. 187–192. https://doi.org/10.1109/ICSS.2014.32

Madhavapeddy, A., Mortier, R., Rotsos, C., Scott, D., Singh, B., Gazagnaire, T., Smith, S., Hand, S., Crowcroft, J., 2013. Unikernels: library operating systems for the cloud. ACM SIGPLAN Not. 48, 461–472. https://doi.org/10.1145/2499368.2451167

Madhavapeddy, A., Scott, D.J., 2014. Unikernels: the rise of the virtual library operating system. Commun. ACM 57, 61–69. https://doi.org/10.1145/2541883.2541895

Maleh, Y., Sahid, A., Ezzati, A., Belaissaoui, M., 2017. Building open virtual cloud lab for advanced education in networks and security, in: 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM). Presented at the 2017 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 1–6. https://doi.org/10.1109/WINCOM.2017.8238172

Mavridis, I., Karatza, H., 2021. Orchestrated sandboxed containers, unikernels, and virtual machines for isolation-enhanced multitenant workloads and serverless computing in cloud. Concurr. Comput. Pract. Exp. n/a, e6365. https://doi.org/10.1002/cpe.6365

Mehringer, S., Barker, B., 2020. Using Containers to Create More Interactive Online Training and Education Materials, in: Practice and Experience in Advanced Research Computing, PEARC '20. Association for Computing Machinery, New York, NY, USA, pp. 246–251. https://doi.org/10.1145/3311790.3396641

Mohammadi, M., 2021. Current trends of blended and hybrid learning : case study: FABLE Project.

Nuere, S., de Miguel, L., 2020. The Digital/Technological Connection with COVID-19: An Unprecedented Challenge in University Teaching - ProQuest.

Oracle, 2012a. 1.1.1. Brief History of Virtualization [WWW Document]. URL https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1010.html (accessed 11.3.21).

Oracle, 2012b. 1.1.2. Hypervisor [WWW Document]. URL https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1011.html (accessed 11.3.21).

Oracle, 2012c. 1.2. Reasons to Use Virtualization [WWW Document]. URL https://docs.oracle.com/cd/E26996_01/E18549/html/BHCJAIHJ.html (accessed 11.3.21).

Pereira-Vale, A., Fernandez, E.B., Monge, R., Astudillo, H., Márquez, G., 2021. Security in microservice-based systems: A Multivocal literature review. Comput. Secur. 103, 102200. https://doi.org/10.1016/j.cose.2021.102200

Petersen, K., Vakkalanka, S., Kuzniarz, L., 2015. Guidelines for conducting systematic mapping studies in software engineering: An update. Inf. Softw. Technol. 64, 1–18. https://doi.org/10.1016/j.infsof.2015.03.007

phusion, 2021. phusion/baseimage - Docker Image | Docker Hub [WWW Document]. URL https://hub.docker.com/r/phusion/baseimage/ (accessed 3.2.22).

Plauth, M., Feinbube, L., Polze, A., 2017. A Performance Survey of Lightweight Virtualization Techniques, in: De Paoli, F., Schulte, S., Broch Johnsen, E. (Eds.), Service-Oriented and Cloud Computing, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 34–48. https://doi.org/10.1007/978-3-319-67262-5_3

Prat, N., Wattiau, I., Akoka, J., 2014. Artifact Evaluation in Information Systems Design Science Research ? A Holistic View. Proc. - Pac. Asia Conf. Inf. Syst. PACIS 2014.

Red Hat, 2020. What is a hypervisor? [WWW Document]. URL https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor (accessed 12.2.21).

Salvini, G., Hofstede, G.J., Verdouw, C.N., Rijswijk, K., Klerkx, L., 2020. Enhancing digital transformation towards virtual supply chains: a simulation game for Dutch floriculture. Prod. Plan. Control 0, 1–18. https://doi.org/10.1080/09537287.2020.1858361

Schonlau, M., Couper, M.P., 2017. Options for Conducting Web Surveys. Stat. Sci. 32, 279–292.

Schöpfel, J., 2010. Towards a Prague Definition of Grey Literature.

Singh, S., Singh, N., 2016. Containers & Docker: Emerging roles & future of Cloud technology, in: 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (ICATccT). Presented at the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), pp. 804–807. https://doi.org/10.1109/ICATCCT.2016.7912109

Sohrabi, C., Alsafi, Z., O'Neill, N., Khan, M., Kerwan, A., Al-Jabir, A., Iosifidis, C., Agha, R., 2020. World Health Organization declares global emergency: A review of the 2019 novel coronavirus (COVID-19). Int. J. Surg. 76, 71–76. https://doi.org/10.1016/j.ijsu.2020.02.034

Sue, V., Ritter, L., 2012. Conducting Online Surveys. SAGE Publications, Inc., 2455 Teller Road, Thousand Oaks California 91320 United States. https://doi.org/10.4135/9781506335186

Tomer, C., 2017. Cloud computing and virtual machines in LIS education: options and resources. Digit. Libr. Perspect. 33, 14–39. http://dx.doi.org.ezproxy.cc.lut.fi/10.1108/DLP-02-2016-0008

Ubuntu, 2022. Ubuntu 16.04 LTS (Xenial Xerus) [WWW Document]. Ubuntu. URL https://ubuntu.com/16-04 (accessed 3.2.22).

Uden, L., Liberona, D., Sanchez, G., Rodríguez-González, S., 2021. Learning Technology for Education Challenges.

VMware, 2008. SQL Server Workload Consolidation 14.

Vuorio, J., Ranta, M., Koskinen, K., Nevalainen-Sumkin, T., Helminen, J., Miettunen, A., 2021. Etäopetuksen tilannekuva koronapandemiassa vuonna 2020 132.

Watada, J., Roy, A., Kadikar, R., Pham, H., Xu, B., 2019. Emerging Trends, Techniques and Open Issues of Containerization: A Review. IEEE Access 7, 152443–152472. https://doi.org/10.1109/ACCESS.2019.2945930

Westphall, C., Lee, Y., Duncan, B., Olmsted, A., Vassilakopoulos, M., Lambrinoudakis, C., Katsikas, S., Ege, R., 2017. CLOUD COMPUTING 2017 - The Eighth International Conference on Cloud Computing, GRIDs, and Virtualization.

Williams, D., Koller, R., Lucina, M., Prakash, N., 2018. Unikernels as Processes, in: Proceedings of the ACM Symposium on Cloud Computing. Presented at the SoCC '18: ACM Symposium on Cloud Computing, ACM, Carlsbad CA USA, pp. 199–211. https://doi.org/10.1145/3267809.3267845

World Health Organization, 2021. Timeline: WHO's COVID-19 response [WWW Document]. URL https://www.who.int/emergencies/diseases/novel-coronavirus-2019/interactive-timeline (accessed 10.20.21).

Xu, L., Huang, D., Tsai, W.-T., 2012. V-lab: a cloud-based virtual laboratory platform for hands-on networking courses, in: Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE '12. Association for Computing Machinery, New York, NY, USA, pp. 256–261. https://doi.org/10.1145/2325296.2325357

APPENDIX 1. Online survey questions (English)

*Required questions are marked by an asterisk (*)*

Multiple-choice questions are marked by letter m ([m])

Open feedback is marked by letter f ([f])

Q1. Gender

Options: man / woman / other

Q2. Age

Options: under 20 / 20-25 / 26-30 / 31-35 / 36+

Q3. What is your highest attained degree?

Options: comprehensive school / vocational school / upper secondary school / university of applied sciences / bachelor's degree / master's degree / other, what?

Q4. How would you rate your own IT capabilities?

Options: nonexistent / weak / average / good / excellent

Q5. Did you complete the course project *Project 6: Portable Virtual Linux workstation*, or did you test the virtual environment listed under the project? *

Options: Yes / No

Q6. For you, what were the primary factors that led to the selection of another project? [m]

Options: project was too challenging / the topic doesn't interest me / the file size of the workstation was too large / my computer doesn't support virtualization / The system requirements aren't compatible with my personal computer / other

Q7. During the course, did you face challenges with the given tools or environments? If yes, what?

Options: yes [f] / no

Q8. Could you see the tailored workstation being beneficial for this or some other course? If yes, which course, and for which purpose? If no, why?

Options: yes [f] / no [f] / no opinion

Q9. How much previous experience do you have working with virtual machines?

Options: nonexistent / very low / fairly low / fairly high / very high

Q10. Have you used Ubuntu or some other Linux distribution previously?

Options: yes / no

Q11. Estimate the time you've used to test the virtual environment.

Options: 1-3 hours / 4-6 hours / 7-9 hours / 10-12 hours / 13-15 hours/ 16 or more hours

Q12. Choose the most relevant option related to the Linux workstation user experience.

Likert scale 1-5: (1) completely disagree / (2) somewhat disagree / (3) no opinion / (4) somewhat agree / (5) completely agree

S1. The workstation aided me in my studies
S2. The workstation provided me with necessary tools to complete the course tasks
S3. Installation and deployment of the workstation was easy
S4. Workstation usage and completion of the exercises was easy
S5. I can see the tailored workstation being beneficial to other university offered courses
S6. The file size of the workstation made the deployment of the workstation harder
S7. Navigation within the workstation was easy
S8. Usage of the virtualization tool (VMware Workstation Player) and Linux-workstation was easy to learn

Q13. Could you see the tailored workstation being beneficial for this or some other course? If yes, which course, and for which purpose? If no, why?

Options: yes [f] / no [f] / no opinion

Q14. Open feedback [f]

APPENDIX 2. Online survey questions (Finnish)

*Pakolliset kysymykset merkitty tähdellä (*)*

Monivalintakysymykset merkitty m-kirjaimella (m)

Avoin palaute merkitty f-kirjaimella (f)

Q1. Sukupuoli

Vaihtoehdot: mies / nainen / muu

Q2. Ikä

Vaihtoehdot: alle 20 / 20-25 / 26-30 / 31-35 / 36+

Q3. Mikä on korkein suorittamasi tutkinto?

Vaihtoehdot: peruskoulu / ammattikoulu / lukio / ammattikorkeakoulu / alempi korkeakoulututkinto / ylempi korkeakoulututkinto / Muu, mikä?

Q4. Miten arvioisit omia tietoteknisiä valmiuksiasi?

Vaihtoehdot: olematon / heikko / keskiverto / hyvä / erinomainen

Q5. Teitkö kurssilla harjoitustyöprojektin *Projekti 6: Virtuaalinen Linux-työasema kotiinkuljetuksella*, tai testasitko projektissa tarjottua virtuaaliympäristöä? *

Vaihtoehdot: Kyllä / Ei

Q6. Mitkä olivat osaltasi pääasialliset tekijät, jotka johtivat toisen harjoitystyöprojektin valintaan? m

Vaihtoehdot: projekti oli liian haastava / aihe ei kiinnosta / työaseman tiedostokoko on liian suuri / tietokoneeni ei tue virtualisaatiota / työaseman järjestelmävaatimukset eivät ole yhteensopivat tietokoneeni kanssa / muu

Q7. Kohtasitko kurssilla haasteita työkalujen tai ympäristön käyttöönotossa? Jos kyllä, niin mitä?

Vaihtoehdot: kyllä [f] / ei

Q8. Kokisitko kurssin tehtävien tekemiseen räätälöidystä työasemasta olevan hyötyä tällä tai jollakin muulla kurssilla? Jos kyllä, niin millä kurssilla ja mihin tarkoitukseen? Jos ei, niin miksi ei?

Vaihtoehdot: kyllä [f] / ei [f] / en osaa sanoa

Q9. Kuinka paljon sinulla on aikaisempaa kokemusta virtuaalikoneiden kanssa työskentelystä?

Vaihtoehdot: ei lainkaan / erittäin vähän / melko vähän / melko paljon / erittäin paljon

Q10. Oletko käyttänyt Ubuntua tai jotakin muuta Linux-jakelua aikaisemmin?

Vaihtoehdot: kyllä / ei

Q11. Kuinka paljon aikaa arvioisit käyttäneesi virtuaaliympäristön testaamiseen?

Vaihtoehdot: 1-3 tuntia / 4-6 tuntia / 7-9 tuntia / 10-12 tuntia / 13-15 tuntia / 16 tai enemmän

Q12. Valitse mielestäsi sopivin vaihtoehto liittyen virtuaalisen Linux-työaseman käyttökokemukseen

Likert scale 1-5: (1) Täysin eri mieltä / (2) Jokseenkin eri mieltä / (3) ei samaa eikä eri mieltä / (4) Jokseenkin samaa mieltä / (5) täysin samaa mieltä

S1. Työasema edisti oppimistani
S2. Työasema tarjosi tarvittavat työkalut kurssin tehtävien tekemiseksi
S3. Työaseman asennus ja käyttöönotto oli helppoa
S4. Työaseman käyttö ja tehtävien tekeminen oli helppoa
S5. Uskon kurssille räätälöidystä työasemasta olevan hyötyä muilla yliopiston tarjoamilla kursseilla
S6. Työaseman tiedostokoko hankaloitti ympäristön käyttöönottoa
S7. Työaseman sisäinen navigointi oli helppoa
S8. Virtualisointityökalun (VMware Workstation Player) ja Linux-työaseman käyttäminen oli nopeasti opittavissa

Q13. Kokisitko kurssin tehtävien tekemiseen räätälöidystä työasemasta olevan hyötyä jollakin muulla kurssilla? Jos kyllä, niin millä kurssilla ja mihin tarkoitukseen? Jos ei, niin miksi ei?

Vaihtoehdot: kyllä [f] / ei [f] / en osaa sanoa

Q14. Vapaa palaute [f]