Ollscoil na Gaillimhe
University of Galway

University of Galway
College of Science and Engineering, School of Computer Science
Department Head: Professor Michael Madden

Web Platform for Computing Principles Education

Master Thesis

Jakub Duras

Advisor: Dr Frank Glavin

This thesis is submitted for the qualification
MSc in Software Engineering and Database Technologies

August 2023

Final Thesis Submission

MSc in Software Engineering and Database Technologies

School of Computer Science
College of Science and Engineering
University of Galway

Student Name:	Jakub Duras
Telephone:	+421 917 432 974
E-mail:	jakub@duras.me
Date of Submission:	06-09-2023
Title of Submission:	Web Platform for Computing Principles Education
Supervisor Name:	Dr Frank Glavin

Certification of Authorship:

I hereby certify that I am the author of this document and that any assistance I received in its preparation is fully acknowledged and disclosed in the document. I have also cited all sources from which I obtained data, ideas or words that are copied directly or paraphrased in the document. Sources are properly credited according to accepted standards for professional publications. I also certify that this paper was prepared by me for the purpose of partial fulfilment of requirements for the Degree Programme.

Signature:  **Date:** 06-09-2023

Acknowledgements

WIP

First of all, I would like to thank my advisor Dr Frank Glavin and university staff who patiently answered my questions and provided me with helpful feedback. Secondly, I would like to thank my brother Jan Duras for the help with the graphic design and adaptation of Nand2Tetris learning material. Moreover, I would like to thank everybody who generously offered their time to participate in the comparative study. Last but not least, I would like to thank my family, friends, and collegues, who encouraged me and gave me the time and space needed to focus on this thesis.

Contents

List of Tables

List of Figures

List of Acronyms

ALU	Algorithmic Logic Unit.
API	Application Programming Interface.
AST	Abstract Syntax Tree.
BFS	Breadth First Search.
CAGR	Compound Annual Growth Rate.
CC	Creative Commons.
CS	Computer Science.
CST	Concrete Syntax Tree.
CUE	Components of User Experience.
EU	European Union.
HDL	Hardware Description Language.
HTML	HyperText Markup Language.
HTTP	Hypertext Transfer Protocol.
ICT	Information and Communication Technology.
IDE	Integrated Development Environment.
ISO	International Organisation for Standardisation.
JS	JavaScript.

KPI	Key Performance Indicator.
meCUE	Modular Evaluation of Key Components of User Experience.
MOOC	Massive Open Online Course.
OS	Operating System.
PSSUQ	Post-Study System Usability Questionnaire.
RAM	Random Access Memory.
REST	Representational State Transfer.
ROI	Return on Investment.
SUS	The Software Usability Scale.
UEQ-S	User Experience Questionnaire Short.
UMUX	Usability Metric for User Experience.
URL	Uniform Resource Locator.
UX	User Experience.
WAI	Web Accessibility Initiative.
WCAG	Web Content Accessibility Guidelines.
WCAG-EM	Website Accessibility Conformance Evaluation Methodology.
WYSIWYG	What You See Is What You Get.
YoY	Year over Year.

Abstract

To be done once everything else is finished. Max 300 words!

Possible line of thought: High-level concepts have received more attention on the web - online IDEs. Introduction to low-level principles not as much. Example - the most popular course on this topic on Coursera - nand2tetris - desktop tools. Shift in the devices to mobile and chromebooks. Universal design - higher market share of mobile devices - especially in emerging markets. Built-in accessibility with web tools, can be embedded into learning material. Outlines the process and considerations needed to migrate desktop educational tool to web. Mentions advantages of switching to web (percentage of covered devices, improved UX metrics - primary research).

1 Introduction

Please skip for now. Taken from Thesis Proposal with only minor adjustments and notes. Should be two pages that would outline content of each chapter and end with a thesis statement.

Understanding low-level computing principles can be beneficial for a wide variety of people - from the general public interested in computers, through practising software engineers, to Computer Science students. Rather popular material is the open-source licensed Nand2Tetris "taught at 400+ universities, high schools, and bootcamps" that explains how to build a computer from individual logic gates to high-level programming language (Shocken and Nisan 2017). The material is accompanied by a desktop software that is hard to access or completely inaccessible from a new class of devices unsuited for desktop Java applications: Chromebooks that are seeing massive growth and now account for the majority of the US K12 market (Boreham 2019) (IDC 2021), or mobile devices that account for 56% of web traffic, up from only 6% in 2011 (**StatCounter˙2021**). The recently created web-based alternative, WepSIM, is showing promising results but takes a more complex look focusing on the CPU and instruction processing (García-Carballeira *et al.* 2019).

Common problems with Massive Open Online Courses (MOOCs) leading to a high rate of dropouts include lack of time, problems adopting new systems, and bad past experience with technical problems on MOOC platforms (Onah *et al.* 2014). The use of MOOCs for software engineering education within higher education is argued to broaden student knowledge and is integrated by some universities in their courses and programmes (Stikkolorum

et al. 2014). However, MOOCs are also said to require significant time to both create and integrate (Stikkolorum *et al.* 2014).

This thesis proposes a new web platform for learning computing principles involving logic gates via a basic individually usable tool integrated into example content.

2 Literature Review

Outline will be added once I receive an approval on the breadth.

2.1 Human-System Interaction Ergonomics

The following section concentrates on the selected topics of Human-System Interaction Ergonomics: Accessibility, Usability, and User Experience (UX). The selected topics are explored in relation to computer software and, where possible, web services specifically. After introducing the concepts, this section focuses on non-obvious benefits and delves into the implementation and evaluation strategies. This section strives to introduce the topics and then dive deeper into pragmatic information relevant to this thesis.

2.1.1 Definition and Distinction

Keeping in mind there are incompatible definitions for Accessibility provided by International Organisation for Standardisation (ISO) (Wegge and Zimmermann 2007), the specific definition chosen for this thesis describes Accessibility as the "extent to which [a service] can be used by people from a population with the widest range of user needs, characteristics and capabilities to achieve identified goals [...]" (ISO 2018). ISO (2020) adds that Accessibility includes but does not apply exclusively to formally disabled people. That means Accessibility is concerned with the basic ability to utilise the software by the widest possible range of users, including disabled users (Wegge and Zimmermann 2007). Similar concepts include "design for all",

"inclusive design", or "universal design" (Sauer *et al.* 2020, p. 1210) with a viewpoint of designing services for all types of people, including the disabled. Wegge and Zimmermann (2007) point to existing confusion between the terms Accessibility and Usability. Although Wegge and Zimmermann (2007) admit these terms are related and have potential overlap, Wegge and Zimmermann (2007) stress the importance of their distinction. Importantly, Information and Communication Technology (ICT) is one of the fields where Accessibility, depending on the country and sector, is mandated by the law (Wegge and Zimmermann 2007; Sauer *et al.* 2020). For example, public sector services within European Union (EU) have to meet accessibility standards outlined in the Web Accessibility Directive, and there is an ongoing effort to extend this to the private sector (European Commission 2021).

Usability, on the other hand, is defined by ISO (2018) as the "extent to which [a service] can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction [...]". As Wegge and Zimmermann (2007) hint, compared to Accessibility, Usability deals with the success - "effectiveness, efficiency and satisfaction" (ISO 2018) - of software interactions which is hard to mandate the way Accessibility is. Therefore, there are few relevant laws and Usability is usually considered more as a competitive advantage that authors are trying to capitalise on (Wegge and Zimmermann 2007).

Similarly to confusion between Accessibility and Usability, researchers point to issues with the distinction between Usability and User Experience (Darin *et al.* 2019; Sauer *et al.* 2020). There are multiple views on the definition of UX and a great amount of disagreement on the topic (Sauer *et al.* 2020). ISO (2018) defines UX as "user's perceptions and responses that result from the use and/or anticipated use of [a service]". Compared to the sole satisfaction that was mentioned to be a part of Usability and Darin *et al.* (2019) argue is incorrectly used as, and often only, UX measurement, UX deals with "users' emotions, beliefs, preferences, perceptions, comfort, behaviours, and accomplishments" (ISO 2018). Additionally, UX is concerned with a broader timeline - the time spent performing the task *and* the time

before and after that (Sauer *et al.* 2020; ISO 2018). Considering the definition by ISO (2018) and in partial disagreement with Darin *et al.* (2019), we can look at UX as a concept that overlaps with Usability or a whole complex superset of Usability (Sauer *et al.* 2020).

In short, for the purposes of this thesis, Accessibility is the basic ability to use the software regardless of the user's various limitations, Usability is the extent to which it can be used to achieve delineated goals successfully, and UX is a more complex extension also concerned with impressions both during and outside of the use of the software. The following subsections always refer to these ideas as Accessibility, Usability, and UX, even if cited literature refers to them differently.

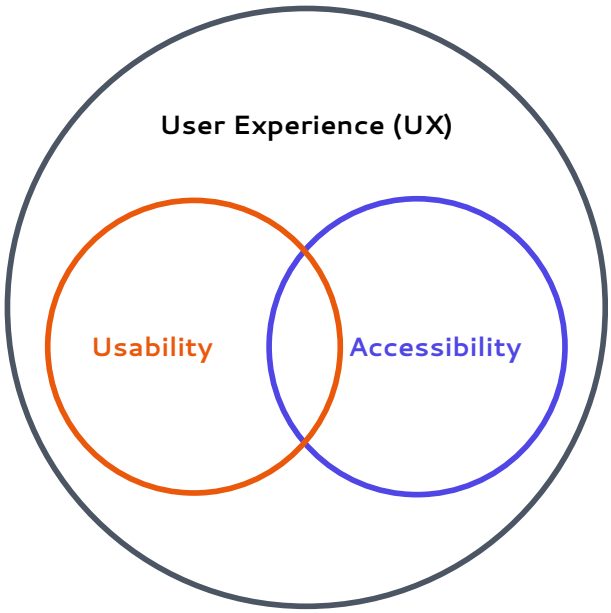

Figure 2.1: Relation between Accessibility, Usability, and UX.

2.1.2 Benefits

Sauer *et al.* (2020) point to their earlier research results that indicate implementing Accessibility on the web could provide benefits to users other than the usual primary target of Accessibility enhancements - disabled users (Schmutz *et al.* 2016; Schmutz *et al.* 2017; Schmutz *et al.* 2018). This notion

is seconded by Vanderheiden (2000), who mentions that others in similar challenging situations can benefit from Accessibility as well. Edyburn (2010) provides terminology identifying these two groups of users: primary and secondary beneficiaries. Specifically in the educational setting, Edyburn (2021) mentions that Accessibility can improve the ability to use the software for both primary beneficiaries - disabled students - and secondary beneficiaries - all other students.

Examples of secondary beneficiaries concerning the web are mentioned by Henry (2021b):

- people using different devices with smaller screens or "different input modes" (Henry 2021b),
- people challenged by limitations introduced by ageing,
- temporarily limited people - e.g. by a "broken arm or lost glasses [...or a] bright sunlight" (Henry 2021b), or conditions that do not allow audio playback,
- and people limited by their internet connection - i.e. latency or bandwidth.

However, Sauer *et al.* (2020) also mention that benefits from some specific concepts could be limited. For example, "using easy language" (Sauer *et al.* 2020, p. 1210) was shown to have limited benefits and considerable disadvantages like decreased enjoyment and higher required time to consume the content (Schmutz *et al.* 2019).

As for the Usability and UX benefits, as was already mentioned in Definition and Distinction, these can be used as a competitive advantage (Wegge and Zimmermann 2007), or, more specifically, to improve some metric like conversion rate, traffic numbers, user performance, or usage of key features (Nielsen 2008). Surveys summarised by Nielsen (2008) indicate a high, double-digit Return on Investment (ROI) even though this number is declining year on year, presumably due to the ever-improving starting state.

2.1.3 Implementation

Accessibility

According to Wegge and Zimmermann (2007, p. 296), there are three main approaches when implementing the Accessibility requirements:

- Universal Design - designing software to be usable without any modifications by the widest range of users.
- Adaptive Design - designing software to be adaptable to different types of users.
- Interoperability with Assistive Technology - designing software to work with existing assistive software.

Wegge and Zimmermann (2007) warn about the downsides of Universal Design: the ability to hinder the experience of the majority, stigmatisation, and implementation difficulties due to conflicting requirements. Sauer *et al.* (2020) reiterate this point and add that a compromise may need to be made to satisfy all groups. In contrast to that, implementation of Adaptive Design allows to opt-in to alternative, independent representations without influencing other groups of users (Wegge and Zimmermann 2007). Similarly, assuming the software follows the relevant standard Application Programming Interface (API), Interoperability with Assistive Technology allows selected users to use their existing tools, e.g. screen readers (Wegge and Zimmermann 2007). Edyburn (2021) mentions the usefulness of Assistive Technologies like speech-to-text and text-to-speech in relation the learning environment and argues they are already available on most platforms and can be targeted at both types of beneficiaries.

Sauer *et al.* (2020) mention that the web has received a significant amount of attention in regards to Accessibility thanks to its perceived importance. Web Accessibility Initiative (WAI) is recognised as the relevant source of information used to develop and verify the Accessibility of web services (Henry 2021b). Web Content Accessibility Guidelines (WCAG) by WAI are adopted

around the world, including by ISO (2012) and in the law (WAI 2018), an example of which is the EU's Web Accessibility Directive (European Commission 2021). The current stable version is WCAG 2.1. While it is version 2.0 that is standardised in ISO (2012) and is the source for many legally binding documents, any minor versions like WCAG 2.1 are backwards compatible as they contain verbatim all requirements from the previous version (WAI 2021b).

According to WAI (2016), there are two kinds of Accessibility requirements: technical, which are mostly fulfilled by properly utilising available APIs; and interaction/visual requirements implemented with the assistance of real users. Implementing both kinds of requirements should mean the service is both "technically and functionally usable" (WAI 2016). WAI (2016) warns that Accessibility is ideally implemented during the development as implementing it later can be problematic, but also admits addressing all Accessibility issues is challenging even for large projects.

While Accessibility for web services can be implemented following the mentioned industry-standard WCAG, Vanderheiden (2000) argues that implementing Usability and Accessibility, in general, can be overwhelming. Vanderheiden (2000) points to hundreds of strategies that one may end up choosing from without appropriate consideration. Therefore, Vanderheiden (2000) proposes prioritisation using four dimensions:

- Accessibility - does the given feature influence the basic Accessibility of the service. Not implementing the most important ones means the service is completely unusable for some groups, while skipping the less important ones only makes the product harder to use for some groups of users.

- Independence - should the user independently use the feature. Less used and more involved features can be usable and delegated only to more advanced users, while everyday tasks must be usable by all.

- Efficiency and Urgency - how much impact does the task have on the efficiency of the given type of user, and is there some time constraint.

Tasks performed very frequently and within some time constraints have a higher priority. Tasks that are not reversible and a failure can have significant consequences have a higher priority.

- Ease of Implementation - the estimated effort required to implement the Usability enhancement. Enhancements that are easier to implement have higher priority.

Usability and UX

Considering a narrower definition of Usability, Nielsen (1993) looks at Usability as a quality and mentions a service should be both usable and have utility (provide features users need). The relation between Usability, utility and other acceptability requirements is outlined in Figure 2.2. Assuming service is Accessible and has utility, Nielsen (1993) recognises the following five qualitative components:

- Learnability - how easy a service is to pick up for the first time.
- Efficiency - how quickly can tasks be successfully performed.
- Memorability - to what extent is the efficiency reproducible after a period of time not using a service.
- Errors - how many errors are being made, their significance, and recovery.
- Satisfaction - how nice the service feels to use.

The Components of User Experience (CUE) model introduced by Thüring and Mahlke (2007) differentiates between "instrumental" and "non-instrumental" qualities (Sauer *et al.* 2020, p. 1209). Similarly, Hassenzahl *et al.* (2008) distinguish between "pragmatic" and "hedonic" qualities (Sauer *et al.* 2020, p. 1209). Both former categories match the first four qualities mentioned by Nielsen (1993), while the latter are closer to Satisfaction and encompass, e.g., visual aesthetic, relatedness, or stimulation (Thüring and Mahlke 2007; Hassenzahl *et al.* 2008).

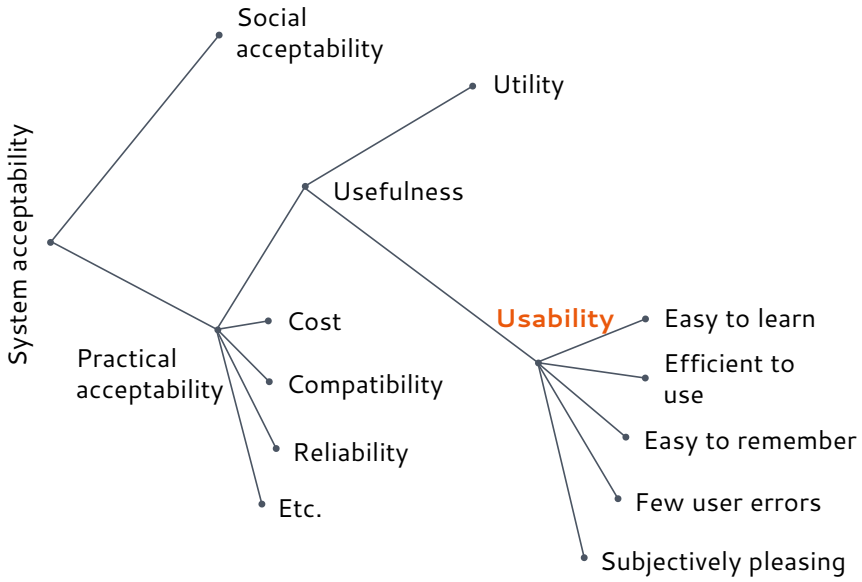

Figure 2.2: Usability in the context of System Acceptability (C. Wilson 2009).

2.1.4 Evaluation

Accessibility

Henry (2021a) acknowledges two ways to evaluate the Accessibility of web services akin to the two kinds of Accessibility requirements outlined in 2.1.3:

- using tools - software can automatically check a service against standards like WCAG or assist humans but cannot solely determine whether a web service is Accessible (WAI 2017),
- with the assistance of humans as target users - users without disabilities or with different disabilities can be involved in evaluating Accessibility but alone cannot determine whether a web service is Accessible (WAI 2020).

A comprehensive evaluation of conformance to WCAG can be performed using Website Accessibility Conformance Evaluation Methodology (WCAG-EM) (WAI 2021a). It can be used for self-assessment or assessment by a

third party and can involve both tools and users (WAI 2021a). WAI (2021a) provides the five steps that make up the WCAG-EM, and that can be executed with the help of WCAG-EM Report Tool[1]:

1. Determine the overall scope - define the evaluated subject, baseline, and objectives, including the mentioned WCAG version and conformance level - A, AA, or AAA - with AAA being the strictest one and AA being the recommended one.

2. Explore the subject - find what are the used technologies, e.g., HTML, WAI-ARIA, or MathML, and identify essential functionality or types of content, e.g., authentication, check-out, or blog pages.

3. Pick sample from the subject - pick specific (random) Uniform Resource Locators (URLs) representative of the findings from the previous step or all available URLs if possible.

4. Evaluate the sample - follow the WCAG to gather information about compliance with all of the requirements of the relevant conformance level for each identified URL.

5. Report the results - create a report, e.g. using the provided template, that includes an executive summary, background, scope, details about the person evaluating, review process approach and goal, and results and recommendations (Brewer 2018).

[1]WCAG-EM Report Tool is available online at `https://www.w3.org/WAI/eval/report-tool`.

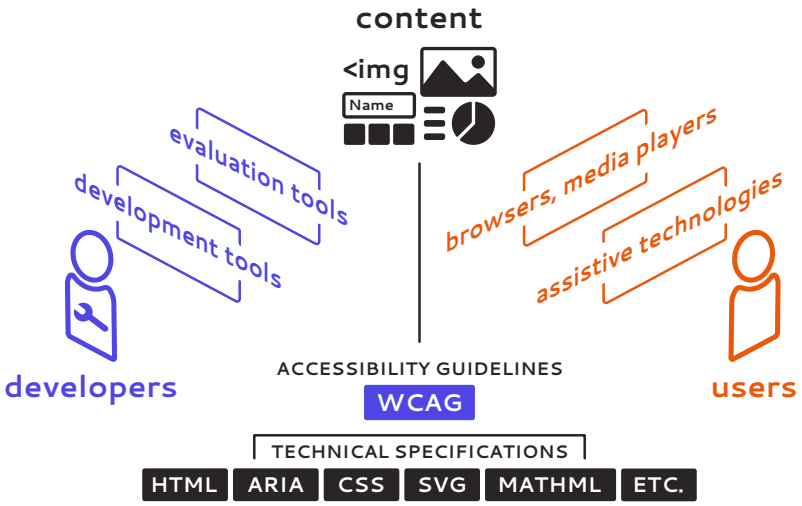

Figure 2.3: Web content relation to Accessibility guidelines, users, and developers (WAI 2021b).

WAI (2017) mentions a wide variety of (semi) automated evaluation tools are available to cover different guidelines, languages, or technologies and working with different environments, inputs, and outputs. Three examples of types of tools in regards to the mode of operation include APIs, browser plugins, and online services[2] (WAI 2017). Findings from Frazão and Duarte (2020), who reviewed the currently available most popular browser plugins, support the recommendation from WAI on the need to use multiple tools and the need to perform manual actions on top of the automated checks. For example, the most popular one, Google's Lighthouse, based on the aXe engine, checks for 19 criteria compared to TotalValidator's 50, and it seems to be more efficient in one kind of criteria while the others are more efficient at finding errors relevant for other criteria (Frazão and Duarte 2020). As far as evaluation with the assistance of users is concerned, WAI (2020) warns users having different backgrounds and representing different groups should

[2]A list of web accessibility evaluation tools maintained by WAI is available online at `https://www.w3.org/WAI/ER/tools`.

be ideally involved to capture a wide variety of real potential users and to eliminate subjectivity. Wegge and Zimmermann (2007) reiterate Accessibility evaluation involves users from different groups that can tell whether the service can be used with their disability, is compatible with their assistive technology, or if the service behaves the way they expect.

Usability and UX

Wegge and Zimmermann (2007) mention that Usability, in contrast to Accessibility, is not typically tested by adherence to standards but by testing and analysing the impact on the end-users. Edyburn (2021) adds that we can monitor subjective measures like decreased frustration or objective measures like increased productivity.

A distinction is pointed to by Sauer *et al.* (2020), who reiterate two kinds of Usability testing:

- formative - helps pinpoint and solve problems with service during the development - called "diagnostic usability" by Lewis (2014),
- summative - evaluates a service to compare it to other service or criteria - called "measurement-based usability" and told to have similarities to experimental psychology by Lewis (2014).

Sauer *et al.* (2020) also argue that the same thinking could be followed when evaluating UX as well, even though such explicit distinction has not been made yet in regards to UX. There is a large number of concrete methods that could be utilised, like questionnaires, observation, interviews, data logging, or user testing (Sauer *et al.* 2020). While expert-based methods are seen as more cost-effective, some argue they do not reflect real use the way user-based do, and user-based methods can generate a broader range of data (Sauer *et al.* 2020). Sauer *et al.* (2020) argue a "principal method" to evaluate the Usability of a service is user testing - users use artefact watched by observers that acquire objective quantitative data, e.g., on efficiency or errors. All while focusing on specific roles and tasks (Wegge and Zimmermann 2007; McCloskey 2014).

UX evaluation that takes into consideration emotion (i.e. not or not only Usability) is argued to be harder to evaluate since it is very individual (Sauer *et al.* 2020). Therefore, expert-based methods are replaced by user-based methods that can be based on more broad but still subjective established questionnaires like Modular Evaluation of Key Components of User Experience (meCUE) (Sauer *et al.* 2020). Considering another angle of subjective UX evaluation that includes Usability, Lewis (2014) asserts practitioners should use standardised Usability questionnaires which are proven to reliably assess perceived Usability and are referenced in national and international standards. For example, The Software Usability Scale (SUS) (Lewis 2018) or Post-Study System Usability Questionnaire (PSSUQ) (Lewis 1995; Lewis 2002). Both of these standardised questionnaires are available for free, should produce statistically significant data, and have a relatively low number of items - 10 for SUS and 16 for PSSUQ (Lewis 2014). An even shorter alternative that has only slightly lower reliability ($r > 0.8$ compared to $r > 0.9$) and seems to produce results correlated to SUS repeatedly is Usability Metric for User Experience (UMUX) with four items and its shorter variant UMUX-LITE with two items (Lewis 2014; Lewis 2018). Schrepp *et al.* (2023) suggest that based on their findings researchers can feel confident choosing SUS or UMUX-LITE if the main objective are Usability-focused aspects. That is mainly thanks to the focus on the Usability aspects that makes these variants lend themselves well to professional software (Schrepp *et al.* 2023). The same cannot be said about software where the software is used for leisure as Schrepp *et al.* (2023) mention only questionnaires like User Experience Questionnaire Short (UEQ-S) were able to capture hedonic qualities.

Some questionnaires like the mentioned SUS have a long history and so can offer useful suggestions based on the data from years of use (J. Brooke 2013). For example, J. Brooke (2013, p. 35) points out that the word "cumbersome" used in one of the questions is not easily understood by non-native English speakers. To address this problem, J. Brooke (2013, p. 35) suggests it can be replaced with the word "awkward", citing several supporting studies. In terms of interpreting the resulting SUS score, J. Brooke (2013, pp. 36-37) points out there are data-backed ways to assign a grade and an

adjective to resulting scores the way Bangor *et al.* (2009) and Sauro (2011) did.

TODO: Add adapted SUS score to adjective/grade here

.

In relation to e-learning software, in particular, Darin *et al.* (2019) also mention that commonly used instruments are questionnaires. That being said, Darin *et al.* (2019, p. 60) also point to "a rising trend [of capturing both] self-reported data [and] UX measurement, in quali-quantitative approaches". A more specific and objective alternative to capturing emotional aspects can be physiological data like body posture (Tan *et al.* 2013). However, these can come with privacy challenges and difficulties evaluating the data (Sauer *et al.* 2020).

In regards to the design of Usability testing, McCloskey (2014) mentions that user tasks should capture key user goals, prompt users to take action, and be engaging by providing context without giving out any clues. Remote Usability testing can, but does not have to, involve interactive sessions between user and facilitator and can be done using automated tools (Moran 2019). Formative Usability testing is likely to be qualitative, and even several users can identify most of the problems (Lewis 2014; Moran 2019). On the other hand, summative Usability testing is more likely to be quantitative and can be used to compare (Macefield 2009; Moran 2019).

Some of the most common quantitative metrics are task success rate and time on task (Macefield 2009; Moran 2019). Macefield (2009) mentions that these are suitable for analysis using statistical methods and can be used to make important decisions like go/no-go for new systems. When processing time on task data, Sauro and Lewis (2016a) suggest using geometric mean, rather than mean or median, as it seems to have a considerably lower error and bias on smaller sample sizes ($n < 25$). However, when comparing two systems, Sauro and Lewis (2016b) do not think it is worth it to use geometric mean as "two-tailed paired t-test is widely considered robust to violations of normality".

Lewis (2014) contends there is no universal number of participants for

summative studies, and one should try to utilise available tools to calculate the required number of participants. Macefield (2009) mentions that a comparative study is essentially a hypothesis test where the aim is to collect evidence a software A has better Usability metrics than software B. As such, it is important to ensure the results are statistically significant by having a low probability ($p \leq 0.1$, ideally $p \leq 0.05$) of the *null hypothesis* (Macefield 2009). Since increasing the number of participants should lower this probability, Macefield (2009) suggests one can run the study until statistical significance is reached or the test fails. Based on the previous data across the field, Macefield (2009) hints that a study group should not be smaller than eight participants, with ten to twelve participants being likely to result in statistically significant data. The same suggestion seems to apply specifically to questionnaires examining Usability on the web as well, with SUS achieving the "correct" conclusion 100% of the time with twelve and more participants on data with a large effect size (Tullis and Stetson 2004). The relation between the sample size and the probability a "correct" conclusion is reached involving different questionnaires can be seen in Figure 2.4.

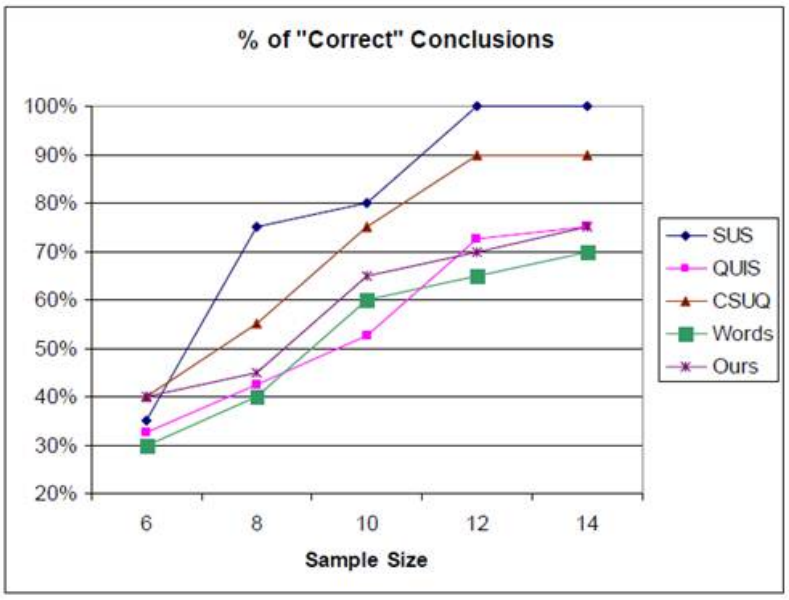

Figure 2.4: Sample size vs "correct" conclusions (Tullis and Stetson 2004).

Macefield (2009) mentions that the size of the effect has a strong influence on the required number of participants, and *power analysis* can be done at different stages to verify the sample size required for statistically significant data or significance in general. However, Dziak *et al.* (2020) argue that using power analysis to decide significance during or after the study (post hoc) should never be done. Alternatives suggested by Dziak *et al.* (2020) for this purpose are *confidence intervals* and *Bayesian analysis*. That being said, Dziak *et al.* (2020) admits Bayesian analysis is not yet very commonly understood. Sauro and Lewis (2016b) concur with the use of confidence invervals when comparing SUS scores or time on task times. If a between-subjects comparison is performed, Sauro and Lewis (2016b) suggest the use of two-sample t-test to take into consideration both variation within the group and between the groups. The relevant formula can be seen in Figure 2.5, where $\hat{x}_1$ and $\hat{x}_2$ are means of data for system 1 and 2, s_1 and s_2 are respective standard deviations, and n_1 and n_2 are respective sample sizes.

$$t = \frac{\hat{x}_1 - \hat{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Figure 2.5: Two-sample t-test formula (Sauro and Lewis 2016b).

The confidence interval can then be calculated using formule in Figure 2.6, where t_a is "the critical value from the t-distribution for the specified level of confidence and degrees of freedom" (Sauro and Lewis 2016b).

$$(\hat{x}_1 - \hat{x}_2) \pm t_a \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

Figure 2.6: Confidence interval formula (Sauro and Lewis 2016b).

2.2 Learning Computing Principles

The goal of the following section is to explore the state of remote learning and Massive Open Online Courses (MOOCs) in general concentrating on the problems or potential challenges, the most popular content available for learning low-level computing principles, and its supporting software. Learning low-level computing principles, in this case, means introduction to topics like logic gates, chip design, computer architecture, or low-level code. The assumed demographic is a senior secondary or postsecondary student, computing professionals without a formal background in computer science, or a general public interested in computers.

2.2.1 State of Remote Learning and MOOCs

Data gathered by the U.S. Department of Education (2021) show roughly 60% of U.S. postsecondary students took at least some of their classes online in 2021. While this number is down from the COVID-19 pandemic level of 75% in 2020, it shows a clear upward trend compared to 36% in 2019 and 25% in 2012 (U.S. Department of Education 2021). More broadly, the global remote learning market was valued at about 300 to 400 billion U.S. dollars in 2022 and is expected to grow at about 10% Compound Annual Growth Rate (CAGR) by some market research companies (Global Industry Analysts (GIA) 2023; Global Market Insights (GMI) 2023). One of the largest MOOC providers, Coursera, reported the total number of registered users grew 61% Year over Year (YoY) during the pandemic year of 2020 and 30% the year after that (Yu *et al.* 2021).

The effects of the accelerated widespread shift to remote learning were especially apparent during the onset of the pandemic that, at one point, UNESCO (2022) estimates caused more than a billion children ($> 70\%$) to be out of the classroom globally. European Commission (2023) shows disadvantaged children and children in countries with lower adoption of ICT were more likely to be negatively affected. Taking into consideration research

from Tadesse and Muluye (2020), these challenges were especially evident in developing countries. Tadesse and Muluye (2020) point out the lack of stable internet connection and devices suitable for remote learning as some of the challenges. In order to support all learners, Ali (2020) mentions, among else, it is necessary the content is "available on a wide variety of devices and mobile friendly" and accessible despite limited bandwidth or even offline.

Although remote learning has seen a major uptick in students, MOOC dropout rates can be up to 90% (Goopio and Cheung 2021). Common problems with MOOCs leading to the high rate of dropouts include lack of time, difficulty, and bad past experience (including technical problems) with MOOC platforms (Onah *et al.* 2014). Goopio and Cheung (2021) mention several software-related recommendations based on their systematic review of research focused on dropouts:

- Increase interactivity by providing activities like quizzes, games, or video interactions.
- Improve course design by using smaller chunks of content with visualization of abstract content and real examples.
- Use various media formats and ensure accessibility from mobile devices and with limited internet connectivity.

From the viewpoint of teaching institutions, MOOCs are said to require significant time to both create and integrate (Stikkolorum *et al.* 2014). That being said, Stikkolorum *et al.* (2014) mention the use of MOOCs for software engineering education within higher education is argued to broaden student knowledge and is successfully integrated by some universities into their courses and programmes. Furthermore, data from the U.S. Department of Education (2021) show that, at least in the U.S., online classes serve a considerably higher number of racially diverse students compared to conventional classes.

2.2.2 Low-Level Computing Principles

One of the most popular, if not the most popular, learning materials for learning computing principles is the Nand2Tetris "taught at 400+ universities, high schools, and bootcamps", which explains how to build a computer from individual logic gates to a working virtual computer (Shocken and Nisan 2017). In the process, Nisan and Schocken (2021) cover a wide variety of computing principles via practical projects:

1. Construction of simple Boolean gates.
2. Design of more advanced 16-bit chips.
3. Incremental construction of an Algorithmic Logic Unit (ALU) starting with simpler chips like incrementer or half and full adder.
4. Introduction of a clock to chips and construction of registers and Random Access Memory (RAM).
5. Programming using simplified assembly.
6. Construction of the final computer utilizing previously built parts.
7. Writing of an assembler that translates custom assembly language into binary runnable on the built computer.

The learning material is offered as a book written by Nisan and Schocken (2021), as a MOOC[3], and in a PDF format[4]. The latter is available freely under a Creative Common Attribution-NonCommercial-ShareAlike 3.0 Unported License. However, it does not contain additional material that builds on the mentioned projects and adds more software-related topics revolving around a Jave-like Virtual Machine, a custom high-level programming language and a standard library for a custom Operating System (OS). Regardless of which form of learning material learners choose, they are expected to use the accompanying desktop Java software[5].

[3]Available at `https://www.coursera.org/learn/build-a-computer`
[4]Available at `https://www.nand2tetris.org`
[5]Available at `https://www.nand2tetris.org/software`

Edybrun (2021) argues the industry now focuses on web-based curricula as the web already has essential accessibility tools, and frameworks like Depth of Knowledge (DOK) lend themselves better to the web environment. More specifically, Edybrun (2021) takes note of the development of more interactive experiences than just "text on a screen" adapted from textbooks. For example, web-based curricula can contain interactive "embedded supports", and Edybrun (2021) mentions these should be context-sensitive. However, the only kind of examples Edybrun (2021) provides are a tool that provides a breakdown and planning of subtasks and "virtual pedagogical agents" similar to virtual assistants like Alexa or Siri but specific to the pedagogical environment.

The recently created web-based WepSIM (García-Carballeira *et al.* 2019) provides microdesign, microprogramming and assembly language educational simulator and was implemented by authors as an interactive way to practice learned concepts. Compared to Nand2Tetris, which takes a more broad approach and goes through many concepts throughout several layers of abstraction, WepSIM is focused only on a thorough simulation of the CPU and instruction processing. García-Carballeira *et al.* (2019) argue the introduction of WepSIM increased the percentage of students taking the final exam and improved grades in assembly and microprogramming exercises. García-Carballeira *et al.* (2019) also add that WepSIM is usable not only as a supplementary tool but also as a standalone learning tool thanks to the bundled examples and help material. Notably, García-Carballeira *et al.* (2019) mention students were able to use WepSim on a wide variety of devices and operating systems, from Windows-NT, Linux, and macOS to Android, iOS, and Windows-Phone.

2.3 Device Type Usage

The following section focuses on the current device type usage and possible trends both in developed and developing countries. The aim is to look at statistics both among the general population and in the education sector.

Web clients are of particular interest due to the focus of this thesis. This section is broken down into two sub-sections for developed and developing markets, as the gathered data are considerably different.

Globally, mobile devices account for around 56% of web traffic, up from only 11% in 2012 (StatCounter 2023e). The shift to mobile devices slowed significantly around the year 2017, when they reached a market share of 52%, and there has been no apparent significant change in data since then (StatCounter 2023e). Tablet devices, on the other hand, seemed to have reached a peak of almost 7% in 2014, and their market share has been steadily declining since (StatCounter 2023e).

In terms of OS and screen resolution, the market is now also very fragmented (StatCounter 2023f; StatCounter 2023g). While in the past, just 10 years ago, Windows had a commanding position, now no OS has majority of the market share and the top four makes up the same market share as Windows alone did in 2009 (StatCounter 2023f). StatCounter (2023g) also shows how there is no major screen resolution with a market share larger than 10% the way it used to be and how the variety of screen resolutions has been growing over time.

2.3.1 Developed Countries

Contrary to the global web traffic data, data for Europe and North America still show a gradual increase in the mobile device market share with parity reached just within the last few years (StatCounter 2023a; StatCounter 2023b). Notably, Chromebooks haven seen a considerable growth in North America and Western Europe and accounted for the majority of the upcoming learners from US K12 market in years 2020 and 2021 (Boreham 2019; IDC 2021). However, tablet and Chromebook shipments have since slowed down noticeably (IDC 2022).

In terms of internet smartphone dependency, about 15% of U.S. adults - up from 8% in 2013 - report a mobile device is the only way they access the internet (Pew Research Center 2021). Importantly, this group is made up

mostly of the 18-29 age group and that is the only age group where we can see a clear trend of increasing smartphone dependency (Pew Research Center 2021). That being said, looking for example at data from Chen *et al.* (2022) collected at University of Central Florida show smartphone use for learning has not increased during the years 2016-2021 with laptops still dominating.

2.3.2 Developing Countries

In sharp contrast to Europe and North America discussed before, developing markets like Africa and Asia have a much higher market share of mobile devices at about 70% (StatCounter 2023c; StatCounter 2023d). Additionally, mobile devices overtook desktop devices much sooner, around the year 2015 (StatCounter 2023c; StatCounter 2023d). Apart from that, as far as tablet devices go, there is no considerable difference we can observe with developing markets (StatCounter 2023c; StatCounter 2023d).

Similarly, while developed countries were seeing noticeable growth in Chromebook and tablet shipments, developing countries did not (Boreham 2019). Internet smartphone dependency is very high in emerging economies, with only about 34% of households having access to a desktop, laptop, or tablet device (Silver *et al.* 2019). Importantly, this number varies greatly between individual developing countries, with India being at the lower end with just 11% and Lebanon being at the high end with 57% (Silver *et al.* 2019). That being said, smartphone ownership is prevalent mainly in the age group of 18-29, with typically less than 40% of people aged 50+ owning a smartphone in developing countries (Silver *et al.* 2019). As far as use in education goes, data seem harder to find, but surveys say the majority of people think smartphones have a positive impact on education and educated people are more likely to own smartphones (Silver *et al.* 2019).

2.4 Text and Code

This section starts with a look at the text as a learning material that humans produce and consume, continues with a production of a source code from the viewpoint of a learner, and ends with a short subsection on how the code is processed by a machine.

2.4.1 Producing Learning Material

For the purposes of this thesis, we can simplify the choice of a way to produce learning material by considering two approaches:

1. What You See Is What You Get (WYSIWYG) editors like Microsoft Word and PowerPoint or Adobe Dreamweaver and Captivate.
2. Plain text formats like LaTeX, HTML, or Markdown and the tools that are necessary to make them usable.

The first group of tools is commonly mentioned when presenting authoring tools for learning content (Khademi *et al.* 2011). While most of these tools are said to be easy to use and often concentrate on creating courses, the majority of them are also paid, and there are concerns about interoperability (Khademi *et al.* 2011; Shieber 2014). One particular concern mentioned by Shieber (2014) is the lack of output consistency when different users use the large range of functionality of these tools differently.

Compared to the problems with the input of WYSIWYG editors mentioned above, plain text input is by nature backwards compatible and interoperable as it is largely a UTF-8 encoded text (Shieber 2014). However, the complexity of plain text formats can differ considerably. LaTeX, which is used extensively within the scientific community, is commonly understood to be more powerful and complex, carries larger overhead on the input, and can potentially take longer to write (Baramidze 2013; Knauff and Nejasmic 2014; Shieber 2014). One of the most common specifications of Markdown, Com-

monmark, features a two-page long self-explanatory reference guide[6] while LaTeX distributes a 31-page long user guide[7]. Moreover, Markdown is said to have a lower overhead than both LaTeX and HyperText Markup Language (HTML) (Shieber 2014) while still permitting extensibility with LaTeX-like features (Shieber 2014) and allowing for potentially better accessibility (Voegler *et al.* 2014). The output format LaTeX and Markdown focus on seems clear from the respective websites. While the LaTeX project website[8] commonly mentions PDF as the output format, Commonmark specification[9] directly mentions HTML as the expected output. HTML output is possible with LaTeX as well, but the efficiency of the most commonly used tool[10] is limited as the produced HTML is said to be often substandard (Voegler *et al.* 2014).

2.4.2 Writing Code

According to a survey conducted by Stack Overflow (2023), more complex Integrated Development Environments (IDEs) like Visual Studio Code, Visual Studio, or IntelliJ IDEA are used by more developers than simpler text editors like Notepad++ or Vim. The most popular IDE used by more than 70% of professionals and learners is Visual Studio Code (Stack Overflow 2023).

Research focusing on the ease of use of sophisticated IDEs, especially among learners, is inconclusive. While a survey from Owoseni and Akanji (2016) suggests learners find sophisticated IDEs difficult to use when learning programming, Vihavainen *et al.* (2014) use session traces to back the opinion there is no evidence learning to use an IDE is hard.

As far as a web-based IDE goes, Valez *et al.* (2020) explored the idea of using a pre-configured custom IDE that is a low-threshold alternative to

[6] Available at `https://commonmark.org/help/`.

[7] Available at `https://www.latex-project.org/help/documentation/usrguide.pdf`.

[8] Available at `https://www.latex-project.org`.

[9] Available at `https://spec.commonmark.org/0.30/`.

[10] LaTeX2HTML available at `https://www.latex2html.org/`.

common IDEs. While most participants found the web IDE useful and found the fact that it was web-based and did not require installation useful, it was also found to often lack the functionality learners were looking for and was used mostly by more junior students (Valez *et al.* 2020).

2.4.3 Parsing Code

This subsection starts with a mention a subset of basic graph terminology relevant for this thesis and some basic algorithms and graph properties. After a brief introduction to graph theory, it is possible to talk about tokenizers and parsers from a practical point of view of a software developer.

Graph Theory

A graph is a data structure that consists of a non-empty finite set of *vertices* and a possibly empty finite set of *edges* (R. J. Wilson 2009). Each edge connects exactly two vertices, and if the edges have a direction, we call the graph a *directed* graph (R. J. Wilson 2009).

There are many different ways how to traverse a graph. One such way is to use the Breadth First Search (BFS) algorithm. BFS follows the following logic (R. J. Wilson 2009):

1. Take a vertex v.
2. Visit all vertices that have an incoming edge from v.
3. Repeat step 1. with the least recently visited v from which we did not yet visit other vertices.

If we can find a path through the graph that starts and ends with the same vertex, it is said it contains a *cycle* (R. J. Wilson 2009). In case some vertex has no associated edges, it is an *isolated vertex*, and if it is not possible to visit all vertices via edges from each vertex, the graph is called *disconnected* (R. J. Wilson 2009). Figure 2.7 shows an example graph that is disconnected

as there is no edge connecting J with the rest of the graph. Additionally, it is cyclic, as vertex DF creates a cycle that could be removed by removing one of the edges DE, EF, or DF.

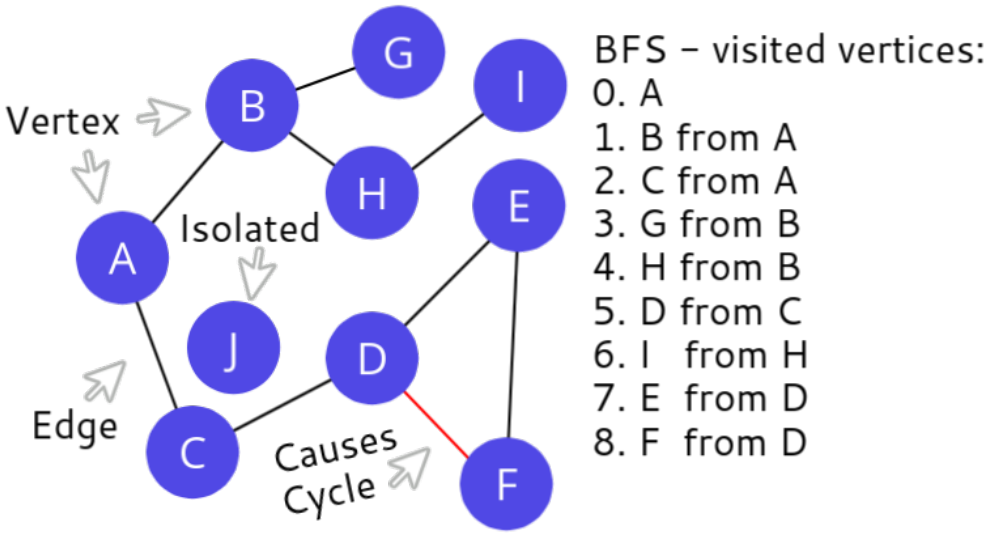

Figure 2.7: Example disconnected graph showing different concepts.

A graph is considered a *tree* if it has exactly one path between any of the vertices, does not contain cycles, and is not disconnected (R. J. Wilson 2009). A tree is called *rooted* if exactly one vertex is chosen as the root and all edges are "directed away from the root" (Baek and Sugihara 2015). Figure 2.8 shows an example of such a tree. Vertices that have at least one outgoing edge are called *internal vertices*, while vertices that have no outgoing edge are called. *Children* of an internal vertex are all vertices that have an incoming edge from that internal vertex (Baek and Sugihara 2015).

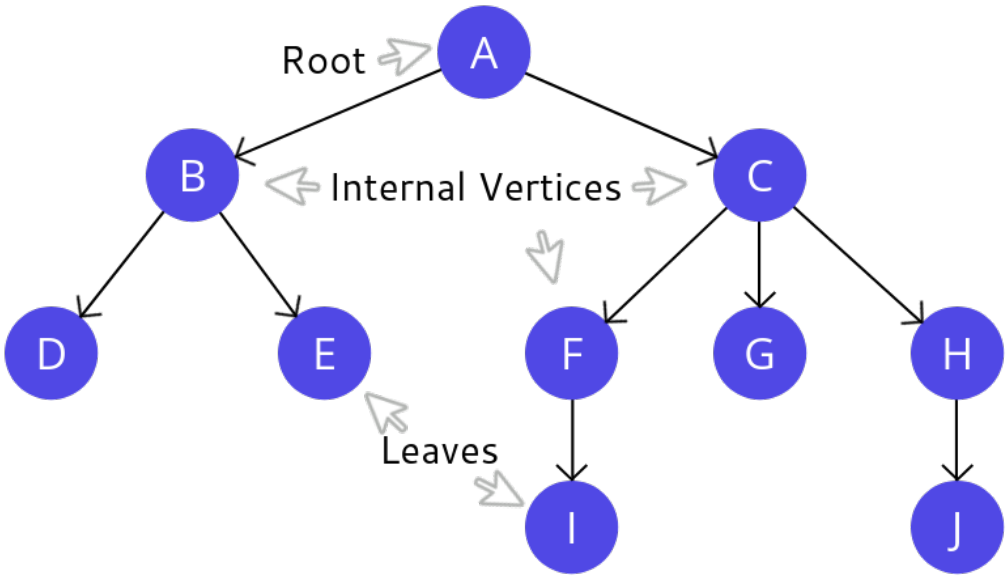

Figure 2.8: Example directed rooted tree.

Tokenizers and Parsers

Is referencing fine here? How to cite sources if I've only loosely used (Tomassetti 2017; Kjolstad 2023) as most of the literature is too theoretical and "unwieldy"?

Parsing is a process of turning a text in a given *language*, like code, into some meaningful object, for example, a program that can be executed (Tomassetti 2017; Kjolstad 2023). A naive approach to parsing is to go through the program character by character using a loop. However, the logic of our language, called *grammar*, would have to be captured within the procedural code, which would make it rather verbose, inflexible, and relatively hard to understand. Instead, the grammar of the language is commonly defined separately and serves as an input for a *parser generator*, which creates the parser. The parser can operate on individual characters, in which case it is said to be *scannerless* as it lacks a *scanner* that would pre-process the text. The scanner groups individual characters into more meaningful tokens and,

for the purposes of this thesis, is also referred to as *lexer* or *tokenizer*.

Therefore, we can say processing of the text on the input goes through three distinct consecutive phases:

1. **Lexing**: A string of characters on the input is broken down using a lexer into a string of tokens.

2. **Parsing**: A string of tokens on the input is turned into a Concrete Syntax Tree (CST) using a parser - if the input is valid.

3. **Processing**: A valid CST, which is mostly just a set of tokens represented using a tree, is simplified into an Abstract Syntax Tree (AST), which is a convenient logical representation of the described object.

From a practical point of view, lexing is typically done using regular expressions as it is the simplest sufficient way to define a regular language used for tokens. Moreover, the processing of parsed tokens can usually be done during the parsing, rather than after, skipping the extra tree traversal. Therefore, parser generators can take: 1. one or more regular expressions defining tokens, 2. grammar, and 3. processing rules on the input and produce encapsulated parser abstraction capable of performing all three mentioned steps. As such, the produced parser takes text on the input and provides the AST on the output.

Since the job of a lexer and parser generator is rather generic, these are abstracted away by a library or a set of libraries. The specification of grammar is left to a human, and the supported features differ between different types of parser generators. A commonly used feature is recursion within the grammar rule. A special case is a left and right recursion when the rule refers to itself on the leftmost or rightmost part of the rule.

Whether a given grammar can be written concisely and the performance of the generated parser is sufficient are typical deciding factors when choosing a parser generator. Conciseness allowed by using features like left and right recursion may require a higher complexity that can mean a worse performance depending on the use case. Simpler and faster parsers called LL parse

from left to right and produce a tree from the largest logical blocks to the smallest logical blocks called *terminals*. More flexible but potentially slower are parsers using the *Earley* algorithm. Earley algorithm uses top-down dynamic programming and enables support of both left and right recursion, and allows for ambiguous grammar. While the average performance of an Ealery parse is $\theta(n^3)$, where n is the number of tokens on the input, there are multiple ways to achieve better performance (and complexity class): by using left recursion, by using unambiguous grammar, or by using an implementation that features optimizations from Leo (1991) if using right recursion in an LL grammar.

2.5 Creating Open Web Application

Put together my notes from Fogel (2017) but put it in contrast with modern software engineering view of Sommerville (2019).

Start with the definition "free vs open-source", state and motivation.

First steps of OSS and soft eng.

Technical aspects - includes servers etc. that support my implementation parts.

Development and release process.

Legal matters - especially CC licenses I will use from Nand2Tetris material, and copyleft/permissive with citation to my undegraduate thesis so I don't plagiarise myself.

Security - including Denial of wallet `https://www.sciencedirect.com/science/article/pii/S221421262100079X` with ref to serverless from tech aspects.

2.5.1 Testing

Google's testing pyramid.

Sensible code coverage `https://research.google/pubs/pub48413/` and `https://www.researchgate.net/publication/324959836_Mythical_Unit_Test_Coverage`.

3 Methodology

The main paradigm used for this thesis is Design Science Research (DSR), as currently outlined by Brocke *et al.* (2020). The author used existing theories, frameworks, and models to produce and evaluate a new innovative artifact - instantiation, as categorised by Hevner *et al.* (2004).

The created artifact is a web application - referred to as "Proposed Software" for the rest of this chapter. The most popular existing software Proposed Software is based on is the Nand2Tetris Java desktop application - referred to as "Existing Software" for the rest of this chapter.

This thesis aims to: show it is feasible for Proposed Software to implement the same functionality on the client-side web; analyse the efficiency improvements compared to Existing Software; and explore the link between usability and efficiency. As such, it starts with a description of the methods used to design Proposed Software. After that, it covers the evaluation of Proposed Software consists of the following:

- **Functional Testing** to verify Proposed Software can perform the selected functions Existing Software can (feature parity of selected parts) and that it works as expected (correctness).
- **Comparative Study** to collect and analyse mainly quantitative and partially qualitative data on the efficiency and usability of Proposed Software and Existing Software while performing the same set of tasks under controlled conditions.

3.1 Design

This section will contain just a few paragraphs. In literature review I explain options/background, methodology here explains how I made choices, design & implementation chapter shows what exactly I have chosen and how I incorporated it.

TODO: What I considered when making decisions around architecture and features.

TODO: How I made the choice custom vs library and what I considered when choosing a library.

3.2 Functional Testing

Multiple types of automated tests were utilized to verify the functionality of Proposed Software.

Unit Tests were used for the verification of granular parts of the software with clear inputs and outputs. That includes all proposed parsers that accept a text on the input and provide an AST on the output; a factory that accepts an AST on the input and produces chips on the output; elementary built-in and simple custom chips that have a clear interface and a definite number of combinations of inputs and outputs; and Representational State Transfer (REST) API that accepts an Hypertext Transfer Protocol (HTTP) request represented as a JavaScript (JS) object on the input and provides a HTTP response represented as a JS object on the output. The exact unit test framework chosen for this purpose was Vitest[1], as it is well-integrated with the code bundler Vite[2] the chosen SvelteKit framework[3] is based on. The expected coverage on all files providing the mentioned functionality should exceed 80%, which corresponds to the median coverage on projects at Google

[1] Available at `https://vitest.dev/`.
[2] Available at `https://vitejs.dev/`.
[3] Available at `https://kit.svelte.dev/`.

and is set to prevent an exponential increase in effort required to increase coverage further - see Testing.

End to End UI Tests verified the software from the viewpoint of a user. In contrast to the Unit Tests, that meant designing automated tests that covered interaction between multiple components and even multiple screens. Examples include navigation through Proposed Software with checks for the existence of essential information, interaction with the IDE, or error handling. The tool of choice for this purpose was Cypress[4]. No automated coverage as a percentage was calculated for this group of automated tests, as many lines of code would be marked as covered (given the nature of these tests) even though they were not actively tested. Instead, all major paths through the software (use cases) were identified and captured in an automated test unless the return on the effort to do so would not be considered worth it. The reasoning was captured individually for each use case that was not covered.

Accessibility Tests checked automatically for common errors with incorrectly implemented APIs - aiming for "technical accessibility" as mentioned in Accessibility. The chosen conformance target was "generally accepted and recommended" WCAG Level AA (WAI 2014) and WCAG version 2.1 as it is the latest supported version within the WCAG-EM Report Tool as of the time of writing. To test for possible violations continually and in more scenarios than can be realistically achieved using one-off evaluation, checks were run on multiple screens in multiple states entered during the execution of Cypress tests. This was accomplished using the JS package cypress-axe that integrates aXe - an accessibility testing engine that powers Google's Lighthouse as mentioned in Accessibility. Considering the limited amount of time and resources, the human testing involved was limited to testing done by the author to produce the WCAG-EM report.

The actual extent and output from all mentioned types of tests, including additional information, where relevant, was captured in Functional Testing.

[4]Available at `https://www.cypress.io/`.

3.3 Comparative Study

The central hypothesis is that using Proposed Software instead of Existing Software will increase efficiency. Therefore, the main objective of the comparative study was to perform summative testing of Existing Software and Proposed Software in a way that produces comparable data. To do that, the author conducted a remote moderated usability testing and performed a between-subjects comparison.

To achieve conditions that would lead to the collection of comparable data, the chosen methodology considers three kinds of variables:

- **Controlled** variables: for example, presented learning content, participant's profile, and actions performed outside of the measured software.
- **Independent** variable: the used software - Existing Software or Proposed Software.
- **Dependent** variables: mainly the time spent on the tasks and the perceived usability.

Since the only independent variable is the used software and there are two options, participants were split into two groups:

- **Group A** using Proposed Software.
- **Group B** using Existing Software.

Target users of the software are expected to come from different backgrounds corresponding to three target demographics mentioned in the Introduction: Computer Science (CS) students, practitioners with/without formal education, and people interested, but not directly involved, in Computer Science. Participants were recruited from the social circle of the author, and they were subsequently asked if they knew any other potentially interested

participants. To control for external motivation and help with the recruitment, one half of participants within both groups were paid to participate for a rate of 10 euro per hour, as suggested by Prolific[5]. However, Prolific or a similar crowdsourcing method was not used to recruit participants. All participants were based in an advanced economy country as defined by International Monetary Fund[6]. Almost all participants were non-native English speakers.

Participants were pre-screened to make sure they:

- Are capable of reading English text of an academic nature.
- Are at least somehow interested in ICT and either want to pursue a career in the field or are already active in the field.
- Have a suitable device to run desktop software.
- Can join a Google Meet voice call and share their screen.

3.3.1 Controlled Variables

Both groups used the same simplified content based on the Creative Commons-licensed (CC) Nand2Tetris content from Nisan and Schocken (2021). After several rounds of pilot studies, the content and assigned tasks were simplified significantly to reduce the time needed to complete the session and to control for the differences in the ability to learn new concepts. As a result, the tasks were mostly mechanical in nature and could be performed just by using the same set of steps that were shown in an example. In the end, participants were expected to spend more time with the software and less time learning and thinking about the presented problems.

The learning material consisted of the same short introduction to Boolean algebra and Boolean gate logic and design. As part of the content, there was

[5]Archived Prolific calculator: `https://web.archive.org/web/20221221162903/https://app.prolific.co/calculator.html`

[6]Available at `https://www.imf.org/external/pubs/ft/weo/2022/01/weodata/groups.htm`.

a video showing how to solve the first example that differed only in the necessary parts that involved the tested software.

The study was conducted by the author using the same online meeting software Google Meet. Each participant took the comparative study individually so that the author could fully focus on only one person, and other people did not influence the results.

All participants used their own devices, with only laptops and desktop computers allowed due to the limitations of Existing Software - a desktop Java software. The use of the participant's device aims to reflect how target users are likely to practice, work on assignments, or explore the content on their own.

To lower the chances the participant would associate the content presented with the author that conducted the study:

- All mentions of the author's name and website header with logo and navigation was removed from the modified version.
- Both videos featured a voice-over from the same professional voice actor with a neutral American accent, slow rate of speech, and clear pronunciation.

The communication between the participant and the author was kept to a minimum and consisted of the following:

1. The participant is greeted and asked whether they are aware of the author's research or if they have heard any details about the comparative study from other participants.
2. The author explains the participant will read a short text, watch a video, and will be expected to complete all presented tasks.
3. The participant is asked to share their screen, and they are informed the screen recording will be kept only until all data are captured and that it will not be shared with any third party.

4. The participant is asked to work on their own and to ask questions only if necessary.

5. The author answers any questions the participant has after the learning content has loaded but before the participant dedicates their attention to the content.

6. The author is strictly muted anytime they are not needed. If necessary, the author answers questions or offers help while the participant solves assigned tasks. This is typically noted in the captured data, as explained when describing dependent variables.

7. After the participant finishes all of the assigned tasks, the author reminds the participant to use the link to open the administered SUS questionnaire. The participant is asked to be as objective as possible, reminding them there are no correct answers and that since the participant does not know which group they belong to, they should not try to knowingly influence the results either way.

Since it is impossible to keep the profile of the participant the same if different groups of target users are to be represented, the author captured the following variables forming the participant's profile:

- **Prior Experience** with the relevant topics on a scale of 0 to 2:

  0 - No prior experience.

  1 - Came into contact with the mentioned topics for a short period of time.

  2 - Has either relevant professional experience or was exposed to the topics for a long period of time.

- **Education** in the field of Computer Science or Computing on a scale of 0 to 2:

  0 - No relevant education.

1 - Taken modules included Boolean logic or programming in general, but not logic gates or chip design.

2 - Learnt about Boolean gates and chip design specifically.

- **Relevancy of Occupation** performed by the participant currently or recently on a scale of 0 to 2:

  0 - No relevant occupation.

  1 - Works as a part-time hardware designer, software developer or similar.

  2 - Works as a full-time hardware designer, software developer or similar.

- **Intrinsically Motivated** to participate, as observed by the author on a scale of 0 to 2:

  0: Shows no significant interest in the topic. Is hesitant to start reading.

  1: Shows some interest in the topic, appears mostly focused.

  2: Is visibly engaged and explicitly confirms their interest in the topic.

- **Paid** to participate:

  1: Yes.

  0: No.

- **Age** on a scale of 18 to 65+:

  18-29

  30-49

  50-64

  65+

- **Platform**:

Windows: One of the commonly used desktop Windows NT versions - Windows 7, Windows 8, Windows 10, or Windows 11.

Linux: One of the currently supported versions of any commonly used distribution that uses a floating window manager like X11 or Wayland.

macOS: One of the currently supported versions, regardless of the CPU architecture - x86 or arm64.

An effort was made to distribute participants among Group A and Group B so that their profiles are distributed equally. Any correlations between these profile variables and dependent variables were analysed and revealed during results analysis to hint at possible data skew.

3.3.2 Independent Variable

The link shared with Group A was side-a.chipsandcode.pages.dev[7] and the link shared with Group B side-b.chipsandcode.pages.dev[8].

The use of Proposed Software by Group A and Existing Software by Group B was reflected mainly in the way tasks were assigned. Group A had Proposed Software embedded within the same page with the assignment preloaded, as can be seen in Figure 3.1. Group B, on the other hand, considering the limitations of Existing Software, had to start by copying three files representing the same assignment, as can be seen in Figure 3.2.

[7]Archived at `https://web.archive.org/web/20230810031115/https://side-a.chipsandcode.pages.dev/`

[8]Archived at `https://web.archive.org/web/20230810031153/https://side-b.chipsandcode.pages.dev/`

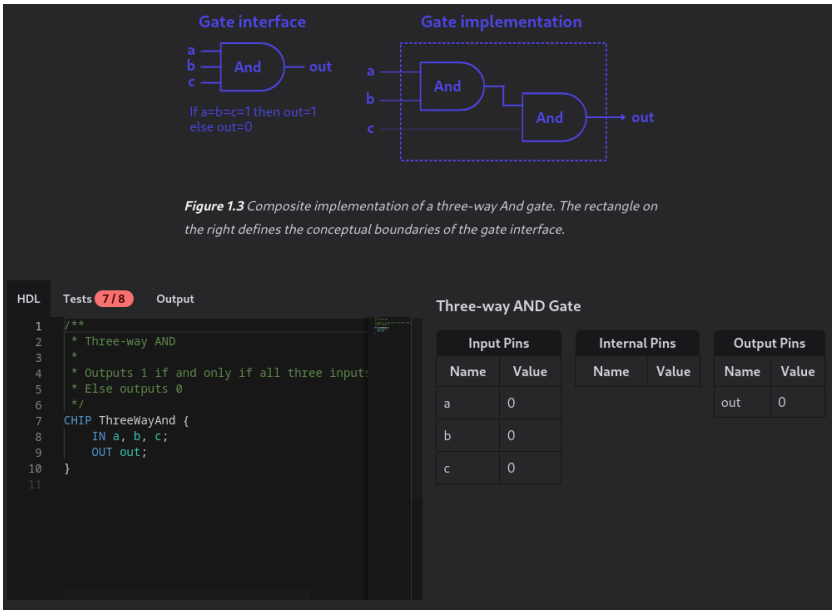

Figure 3.1: Screenshot of Proposed Software within content for Group A.

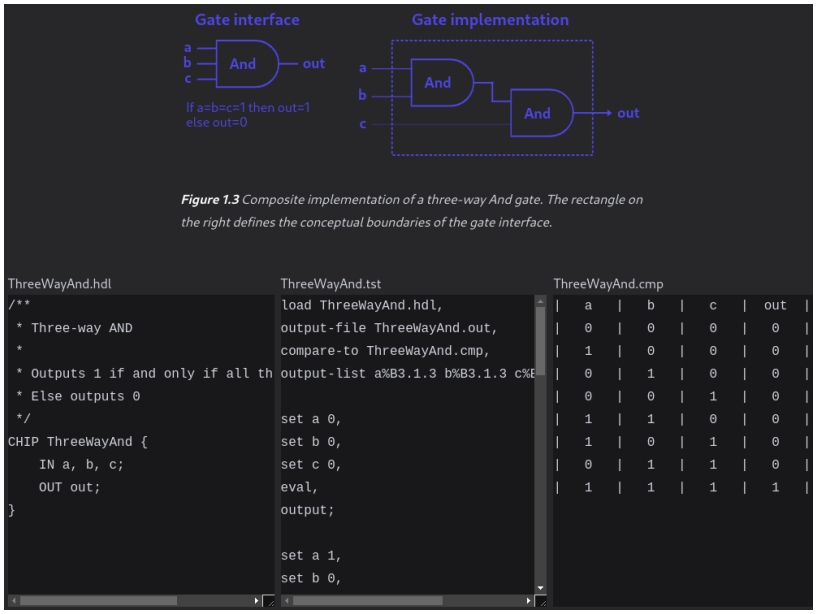

Figure 3.2: Screenshot of Input for Existing Software within content for Group B.

3.3.3 Dependent Variables

The following metrics were measured:

- **Success** - Did the participant complete all assigned tasks - 1/0.
- **Preparation Time** - Time required to start working with the hardware simulator in seconds. This includes the download and installation time in the case of Existing Software.
- **Time for 1st Task** - Time required to finish the implementation of a three-way And Boolean gate in seconds.
- **Time for 2nd Task** - Time required to finish the implementation of a Nand Boolean gate in seconds.
- **Time for 3rd Task** - Time required to finish the implementation of a Nor Boolean gate in seconds.
- **Time for 4th Task** - Time required to finish the implementation of a three-way Or Boolean gate in seconds.
- **Confusions** - Number of times:
  (a) The participant asked for help because they were confused.
  (b) The author offered help to the participant because the participant had been stuck on the same problem for more than about a minute without any progress.
  (c) The participant did not perform the needed action in the user interface due to a misunderstanding and could not immediately correct themselves.

The time it took the participant to initially go through the learning material and the embedded video was not measured and was excluded from the metrics mentioned above. The time to perform a task was measured from the moment the participant started focusing on the part of the website that

contains the task to the time they left the hardware simulator after all comparisons and optional manual testing was done. These times were manually extracted by writing down the timestamp ranges from the screen recording after the session ended, an example of which can be see in Table 3.1.

Table 3.1: Partial timestamp data collected from one of the sessions.

Prep		
From	**To**	**Time**
0:10:45	0:12:25	0:01:40
0:13:32	0:15:06	0:01:34
	Total:	**194**
First Task		
From	**To**	**Time**
0:23:13	0:25:11	0:01:58
0:25:35	0:25:55	0:00:20
0:29:10	0:30:33	0:01:23
	Total:	**221**

While there was no hard time limit, the participant was informed the session would take one hour and was given the option to quit once the allocated time had passed. The chosen approach to time limit aims to represent the way learners at home either have an expectation of how long the task will take or have allocated some time to the task but are willing to go past it if they feel like they are making progress and are close to finishing.

In addition to the mentioned metrics, participants were also asked to fill out the SUS (j. Brooke 1996) questionnaire using a 5-point Likert scale from Strongly agree to Strongly disagree. This questionnaire was chosen based on the amount of supporting data combined with the relatively short length of 10 questions and the relevance of questions to the tested software. The word "cumbersome" was replaced with the word "awkward" to help non-native speakers, see SUS evaluation.

In the process, unstructured qualitative data were collected, including:

- explanations of confusing situations experienced by participants,

- overall description of the experience of participants using the system,
- various feedback.

While the comparison did not primarily concentrate on qualitative data, it was summarised in the chapter Discussion.

3.3.4 Data Analysis

To decrease the likelihod of manual errors when evaluating SUS scores, the author obtained the SUS Guide & Calculator Package[9]. Based on the Usability and UX literature review, it was suggested at least 12 participants per group would need to participate in the study if the assumption is the effect size is relatively large.

The measured efficiency is represented by calculating arithmetic mean of the measured times - individually for each task and added together, and the mean number of confusions. The subjective usability is analysed as a mean of SUS scores for each group. Analysed data are assumed to follow standard distribution and all relevant calculations are done with two-tailed tests.

To determine the stastitical power of the gathered data, the results were evaluated using the two-sample t-test and include:

- p-value, with expected $p \leq 0.1$, ideally $p \leq 0.05$,
- and confidence intervals.

Visually, results are presented using:

- a linear chart with the time each group needed to perform subsequent tasks,
- a stacked bar chart showing the difference in measured efficiency,
- a box plot and a position on a percentile curve for SUS scores inspired by visualisations from Blattgerste *et al.* (2022),

[9]Available at: `https://measuringu.com/product/suspack/`.

- a scatter plot showing the relation between the measured efficiency and SUS score, including color-coding representing the group to reveal clusters,
- and scatter plots that show possible correlations between the various data points in participant profiles and measured efficiency.

3.3.5 Limitations

The comparative study has several design limitations the author is aware of:

- **Sampling Bias**: The study consists primarily of the extended social circle of the author and only includes participants who agreed to participate (self-selection bias).
- **Additional Step**: While Existing Software bundle does include a starting point for projects, they do not include the simple exercises chosen for this study. That means Group B participants had to take one extra step to create starting files.
- **Moderator Bias**: The study was moderated by the author rather than a blind data collector.

4 Design and Implementation

Contains only brief notes.

Outline considerations when migrating desktop education tool to accessible, usable, extendable, secure, and reproducible web application.

Consider mentioned hurdles with MOOC development.

4.1 Accessibility and Usability

Designed to be usable with built-in accessibility tools shown to be used by primary and secondary beneficiaries.

Main points: semantics, ARIA. Figures are SVGs with system-stack fonts and not curves.

Include https://www.w3.org/WAI/about/accessibility-statement/ and https://www.w3.org/WAI/meta/customize/.

Performance/size related:

Bandwith/internet connection (Accessibility benefits backreference)

.

Time to first interaction.

Minimal build size - Svelte via SvelteKit and Tailwind with three shaking, chunking, server side rendering (static build).

Used bandwith and required internet connection.

Compare size and time to download the Nand2Tetris desktop (with and without JRE) app and the highly-optimised web app.

Contrary to expectations - web app does not have to be downloaded again - can be used offline as PWA.

So faster to load than both nand2tetris and wepsim and is loaded only once/works offline like nand2teris and unlike wepsim.

Client device supported:

Usable from all platforms - percentage? Compare to Nand2Tetris.

Progressive enhancement.

IDE - Monaco for accessibility and powerful capabilities. Link back to Literature Review - Monaco at the hearth of the most popular VSCode.

4.2 Extensibility

Everything (including learning materials) control versioned on Github.

Learning materials processed from Markdown files using Markdoc.

4.3 Acceleration of Development

Use of available third party libraries to accelerate development.

Nearley with Moo for simple and powerful parser. Still performant - regex, linear complexity right recursion.

Use of freely available scalable services - Github, Cloudflare - CDN, CI/CD, key/value.

4.4 Security

Infrastructure and deployment: DNSSEC, HTTPS, automation - CI/CD with signed commits - open source considerations!, “Denial of Wallet”.

Anonymous sessions via secure-only http-only cookies.

Parsers and interpreters: No eval or access to arbitrary JS APIs in interpreter.

4.5 Reproducibility

Again - everything is versioned - infrastructure, documentation, content, code.

Made for easy hosting by anybody even without the backend.

Backend optional - can be swapped for other provider.

5 Evaluation

Outline of the covered topics will be added when finishing the chapter.

5.1 Accessibility Evaluation

Automated and manual test results - wcag template report.

5.2 Functional Testing

Unit tests - brief description of lcov results and exceptions. Example unit tests.

List of UI tests based on use cases and showcase of some selected ones. Description of exceptions.

5.3 Comparative Usability Study

The total number of participants who joined the scheduled comparative study session was $n = 26$. Out of that, the data of two participants had to be removed - Participant F, who experienced technical difficulties unrelated to the software and Participant J, who mentioned they tried the Proposed Software in the past. The resulting sample size that was analysed was $n = 24$, with an equal distribution of $n_1 = 12$ and $n_2 = 12$ among both groups.

The list of participants and their profiles within Group A can be seen in Table 5.1, while Group B participant profiles can be seen in Table 5.2.

Table 5.1: Group A participant profiles.

Person	Prior Exp.	Education	Occupation	Age Group	Paid	Intr. Motiv.	Platform
M	1	2	2	30-49	0	0	Ubuntu Linux desktop
LU	0	0	0	18-29	1	1	Windows 10 laptop
Y	1	0	0	30-49	0	2	Windows 10 laptop
MI	0	1	0	30-49	0	0	Ubuntu Linux laptop
MX	1	2	1	30-49	0	1	Windows 10 desktop
R	0	2	0	18-29	1	0	Windows 10 desktop
MK	0	1	2	30-49	0	1	Windows 11 laptop
L	0	2	2	18-29	0	1	macOS 13 arm64 laptop
MA	0	0	0	18-29	1	1	Windows 10 laptop
KY	0	1	1	18-29	1	1	Windows 10 laptop
KA	0	0	0	18-29	1	0	Windows 7 laptop
E	0	0	2	30-49	1	2	Windows 10 desktop

Table 5.2: Group B participant profiles.

Person	Prior Exp.	Education	Occupation	Age Group	Paid	Intr. Motiv.	Platform
P	0	1	1	18-29	0	2	Windows 10 laptop
S	0	1	0	30-49	0	1	Windows 10 desktop
MM	1	1	2	18-29	0	1	Ubuntu Linux laptop
SA	0	1	2	18-29	0	0	Windows 11 laptop
JO	0	0	0	18-29	1	0	Windows 11 desktop
PE	0	1	0	18-29	1	1	Windows 11 laptop
MS	0	2	2	18-29	0	1	Windows 11 laptop
B	1	2	2	18-29	0	1	macOS 13 arm64 laptop
V	0	2	2	18-29	1	1	Windows 10 laptop
MH	1	2	2	30-49	1	1	macOS 13 x86 laptop
LB	0	2	2	18-29	1	1	Windows 10 laptop
MT	0	2	0	18-29	1	2	Windows 10 desktop

As we can see, the profiles were not completely equal between the groups.

To better capture the differences, Figure shows the cumulative “Prior Experience”, “Education”, “Relevant Occupation”, and “Intrinsic Motivation” between the two groups.

Additionally, Figure shows the age make-up of the groups and Figure platforms used by participants of each group.

Linear chart with task times.

Stacked chart with task times.

Box plot for the number of confusions.

RESULTS on all observed metrics, incl. statistical significance.

Appendices with results - questionnaires and raw data - Google spreadsheet printed as PDF.

Box plot for SUS with adjectives and grade + Sauro (2011) percentile rank.

Scatter plot task time vs SUS group-wise color coding.

Correlation between profiles and performance.

6 Discussion

To be done.

7 Conclusions

To be done once everything else is done.

Bibliography

Ali, W. (2020) "Online and Remote Learning in Higher Education Institutes: A Necessity in Light of COVID-19 Pandemic". In: *Higher Education Studies* 10.3, pp. 16–25. ISSN: 1925-4741. DOI: 10.5539/hes.v10n3p16.

Baek, K. and Sugihara, K. (2015) *Discrete Mathematics II ICS 241 11.1 Introduction to Trees - Lecture notes*. Available http://courses.ics.hawaii.edu/ReviewICS241/morea/trees/Trees-QA.pdf [accessed 20th Aug. 2023]. Archived https://web.archive.org/web/20221101000000*/http://courses.ics.hawaii.edu/ReviewICS241/morea/trees/Trees-QA.pdf.

Bangor, A., Kortum, P. and Miller, J. (2009) "Determining what individual SUS scores mean: adding an adjective rating scale". In: *Journal of Usability Studies* 4.3, pp. 114–123. Available https://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/.

Baramidze, V. (2013) "LaTeX for Technical Writing". In: *Journal of Technical Science and Technologies*, pp. 45–48. DOI: 10.31578/.v2i2.63. Available https://jtst.ibsu.edu.ge/jms/index.php/jtst/article/view/63 [accessed 1st Aug. 2023].

Blattgerste, J., Behrends, J. and Pfeiffer, T. (2022) "A Web-Based Analysis Toolkit for the System Usability Scale". In: *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*. PETRA '22. Corfu, Greece: Association for Computing Machinery, pp. 237–246. ISBN: 9781450396318. DOI: 10.1145/3529190.3529216. Available https://doi.org/10.1145/3529190.3529216.

Boreham, M. (2019) *Global K-12 Mobile PC Education Market Continues Growth Momentum*. Tech. rep. Hertfordshire, United Kingdom: Futuresource Consulting. Available https://www.futuresource-consulting.com/insights/new-k-12-mobile-pc-report-confirms-2018-growth-and-upbeat-future/ [accessed 10th Oct. 2021].

Brewer, J. (2018) *Template for Accessibility Evaluation Reports*. Available https://www.w3.org/WAI/test-evaluate/report-template/ [accessed 27th Dec. 2021]. Archived https://web.archive.org/web/20210810181815/https://www.w3.org/WAI/test-evaluate/report-template/.

Brocke, J. v., Hevner, A. and Maedche, A. (2020) "Introduction to Design Science Research". In: pp. 1–13. ISBN: 978-3-030-46780-7. DOI: 10.1007/978-3-030-46781-4_1.

Brooke, J. (2013) "SUS: a retrospective". In: *Journal of Usability Studies* 8.2, pp. 29–40. Available https://uxpajournal.org/sus-a-retrospective/.

Brooke, j. (1996) "SUS: A 'Quick and Dirty' Usability Scale". In: *Usability Evaluation In Industry*. CRC Press. ISBN: 9780429157011.

Chen, B., deNoyelles, A., Brown, T. and Seilhamer, R. (2022) *Internet/Broadband Fact Sheet*. Available https://er.educause.edu/articles/2023/1/the-evolving-landscape-of-students-mobile-learning-practices-in-higher-education.

Darin, T., Coelho, B. and Borges, B. (2019) "Which Instrument Should I Use? Supporting Decision-Making About the Evaluation of User Experience". In: *Design, User Experience, and Usability. Practice and Case Studies*. Ed. by A. Marcus and W. Wang. Cham: Springer International Publishing, pp. 49–67. ISBN: 978-3-030-23535-2. DOI: 10.1007/978-3-030-23535-2_4.

Dziak, J. J., Dierker, L. C. and Abar, B. (2020) "The interpretation of statistical power after the data have been gathered". In: *Current Psychology* 39.3, pp. 870–877. ISSN: 1936-4733. DOI: 10.1007/s12144-018-0018-1.

Edyburn, D. L. (2010) "Would You Recognize Universal Design for Learning if You Saw it? Ten Propositions for New Directions for the Second Decade

of UDL". In: *Learning Disability Quarterly* 33.1, pp. 33–41. ISSN: 0731-9487. DOI: 10.1177/073194871003300103.

Edyburn, D. L. (2021) "Universal Usability and Universal Design for Learning". In: *Intervention in School and Clinic* 56.5, pp. 310–315. ISSN: 1053-4512. DOI: 10.1177/1053451220963082.

European Commission (2021) *Web Accessibility*. Available https://digital-strategy.ec.europa.eu/en/policies/web-accessibility [accessed 4th Dec. 2021].

European Commission (2023) *Covid-19 learning deficits in Europe: analysis and practical recommendations : analytical report*. Publications Office. Available https://data.europa.eu/doi/10.2766/881143.

Fogel, K. (2017) *Producing Open Source Software: How to Run a Successful Free Software Project*. Second edition. O'Reilly Media. Available http://www.producingoss.com/.

Frazão, T. and Duarte, C. (2020) "Comparing accessibility evaluation plugins". In: *Proceedings of the 17th International Web for All Conference*. W4A '20. Association for Computing Machinery, pp. 1–11. ISBN: 9781450370561. DOI: 10.1145/3371300.3383346.

García-Carballeira, F., Calderón-Mateos, A., Alonso-Monsalve, S. and Prieto-Cepeda, J. (2019) "Wepsim: an online interactive educational simulator integrating microdesign, microprogramming, and assembly language programming". In: *IEEE Transactions on Learning Technologies* 13.1, pp. 211–218. DOI: 10.1109/TLT.2019.2903714.

Global Industry Analysts (GIA) (2023) *Global E-Learning Industry*. Tech. rep. San Jose, California: Global Industry Analysts. Available https://www.strategyr.com/market-report-e-learning-forecasts-global-industry-analysts-inc.asp.

Global Market Insights (GMI) (2023) *E-learning Market, 2023-2032*. Tech. rep. Selbyville, Delaware: Global Market Insights. Available https://www.gminsights.com/industry-analysis/elearning-market-size.

Goopio, J. and Cheung, C. (2021) "The MOOC dropout phenomenon and retention strategies". In: *Journal of Teaching in Travel & Tourism* 21.2, pp. 177–197. ISSN: 1531-3220, 1531-3239. DOI: 10.1080/15313220.2020.

1809050. Available https://www.tandfonline.com/doi/full/10.1080/15313220.2020.1809050.

Hassenzahl, M., Koller, F. and Burmester, M. (2008) "Der User Experience (UX) auf der Spur: Zum Einsatz von www. attrakdiff. de". In: *Tagungsband UP08*.

Henry, S. L. (2021a) *Evaluating Web Accessibility Overview*. Available https://www.w3.org/WAI/test-evaluate/ [accessed 4th Dec. 2021]. Archived https://web.archive.org/web/20211130145209/https://www.w3.org/WAI/test-evaluate//.

Henry, S. L. (2021b) *Introduction to Web Accessibility*. Available https://www.w3.org/WAI/fundamentals/accessibility-intro/ [accessed 4th Dec. 2021]. Archived https://web.archive.org/web/20211127115925/https://www.w3.org/WAI/fundamentals/accessibility-intro/.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004) "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1, pp. 75–105. ISSN: 02767783. DOI: 10.2307/25148625.

IDC (2021) *Chromebook and Tablet Growth Continued in the Second Quarter Despite On-Going Supply Concerns*. Tech. rep. Needham, Massachusetts: International Data Corporation. Available https://www.idc.com/getdoc.jsp?containerId=prUS48120621 [accessed 10th Oct. 2021]. Archived https://web.archive.org/web/20230126230623/https://www.idc.com/getdoc.jsp?containerId=prUS48120621.

IDC (2022) *Tablet and Chromebook Shipments Continued to Decline in Q3 Amidst Ongoing Market Headwinds, According to IDC Tracker*. Tech. rep. Needham, Massachusetts: International Data Corporation. Available https://www.idc.com/getdoc.jsp?containerId=prUS49812222 [accessed 10th July 2023]. Archived https://web.archive.org/web/20230330020313/https://www.idc.com/getdoc.jsp?containerId=prUS49812222.

International Standards Organisation (2012) *ISO/IEC 40500:2012 Information technology — W3C Web Content Accessibility Guidelines (WCAG) 2.0*. ISO/IEC 40500:2012. Available https://www.iso.org/standard/58625.html.

International Standards Organisation (2018) *ISO 9241-11:2018 Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts*. ISO 9241-11:2018. Available https://www.iso.org/standard/63500.html.

International Standards Organisation (2020) *ISO/IEC 10779:2020 Information technology — Office equipment — Accessibility guidelines for older persons and persons with disabilities*. ISO/IEC 10779:2020. Available https://www.iso.org/obp/ui/fr/#iso:std:iso-iec:10779:dis:ed-2:v1:en.

Khademi, M., Haghshenas, M. and Kabir, H. (2011) "A Review On Authoring Tools". In: *5th International Conference on Distance Learning and Education*. IACSIT Press. Available https://www.semanticscholar.org/paper/A-Review-On-Authoring-Tools-Khademi-Haghshenas/fac59b388f822adac8bf3338fbb98b4a0690629b [accessed 21st Aug. 2023].

Kjolstad, F. (2023) *Introduction to Parsing CS143 Lecture 5 - Lecture notes*. Available https://web.stanford.edu/class/cs143/lectures/lecture05.pdf [accessed 10th Aug. 2023]. Archived https://web.archive.org/web/20230327182646/https://web.stanford.edu/class/cs143/lectures/lecture05.pdf.

Knauff, M. and Nejasmic, J. (2014) "An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development". In: *PLOS ONE* 9.12, e115069. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0115069. Available https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0115069 [accessed 1st Aug. 2023].

Leo, J. M. I. M. (1991) "A general context-free parsing algorithm running in linear time on every LR(k) grammar without using lookahead". In: *Theoretical Computer Science* 82.1, pp. 165–176. ISSN: 0304-3975. DOI: 10.1016/0304-3975(91)90180-A. Available https://www.sciencedirect.com/science/article/pii/030439759190180A [accessed 21st Aug. 2023].

Lewis, J. R. (2014) "Usability: Lessons Learned ... and Yet to Be Learned". In: *International Journal of Human-Computer Interaction* 30.9, pp. 663–684. ISSN: 1044-7318. DOI: 10.1080/10447318.2014.930311.

Lewis, J. R. (1995) "IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use". In: *International Journal of Human-Computer Interaction* 7.1, pp. 57–78. ISSN: 1044-7318, 1532-7590. DOI: 10.1080/10447319509526110.

Lewis, J. R. (2002) "Psychometric Evaluation of the PSSUQ Using Data from Five Years of Usability Studies". In: *International Journal of Human-Computer Interaction* 14.3–4, pp. 463–488. ISSN: 1044-7318, 1532-7590. DOI: 10.1080/10447318.2002.9669130. Available http://www.tandfonline.com/doi/abs/10.1080/10447318.2002.9669130.

Lewis, J. R. (2018) "The System Usability Scale: Past, Present, and Future". In: *International Journal of Human–Computer Interaction* 34.7, pp. 577–590. ISSN: 1044-7318, 1532-7590. DOI: 10.1080/10447318.2018.1455307.

Macefield, R. (2009) "How to specify the participant group size for usability studies: a practitioner's guide". In: *Journal of Usability Studies* 5.1, pp. 34–45. DOI: 10.5555/2835425.2835429.

McCloskey, M. (2014) *Task Scenarios for Usability Testing*. Available https://www.nngroup.com/articles/task-scenarios-usability-testing/.

Moran, K. (2019) *Usability Testing 101*. Available https://www.nngroup.com/articles/usability-testing-101/.

Nielsen, J. (1993) "Chapter 2 - What Is Usability?" In: *Usability Engineering*. Morgan Kaufmann, pp. 23–48. ISBN: 9780125184069. DOI: 10.1016/B978-0-08-052029-2.50005-X. Available https://www.sciencedirect.com/science/article/pii/B978008052029250005X.

Nielsen, J. (2008) *Usability ROI Declining, But Still Strong*. Available https://www.nngroup.com/articles/usability-roi-declining-but-still-strong/ [accessed 21st Dec. 2021].

Nisan, N. and Schocken, S. (2021) *The elements of computing systems: building a modern computer from first principles*. Second edition. Cambridge, Massachusetts: The MIT Press. ISBN: 9780262539807.

Onah, D. F., Sinclair, J. and Boyatt, R. (2014) "Dropout rates of massive open online courses: behavioural patterns". In: *EDULEARN14 proceedings* 1, pp. 5825–5834. Available http://wrap.warwick.ac.uk/65543/.

Owoseni, A. and Akanji, S. A. (2016) "Survey on Adverse Effect of Sophisticated Integrated Development Environments on Beginning Programmers' Skillfulness". In: *International Journal of Modern Education and Computer Science* 8.9, pp. 28–34. ISSN: 20750161, 2075017X. DOI: 10.5815/ijmecs.2016.09.04. Available http://www.mecs-press.org/ijmecs/ijmecs-v8-n9/v8n9-4.html [accessed 1st Aug. 2023].

Pew Research Center (2021) *Internet/Broadband Fact Sheet*. Tech. rep. Washington, D.C.: Pew Research Center. Available https://about.coursera.org/press/wp-content/uploads/2021/11/2021-Coursera-Impact-Report.pdf.

Sauer, J., Sonderegger, A. and Schmutz, S. (2020) "Usability, user experience and accessibility: towards an integrative model". In: *Ergonomics* 63.10. PMID: 32450782, pp. 1207–1220. DOI: 10.1080/00140139.2020.1774080.

Sauro, J. (2011) *A practical guide to the system usability scale: background, benchmarks & best practices*. Denver, CO: Measuring Usability LLC. ISBN: 9781461062707.

Sauro, J. and Lewis, J. R. (2016a) "Chapter 3 - How precise are our estimates? Confidence intervals". In: *Quantifying the User Experience (Second Edition)*. Boston: Morgan Kaufmann, pp. 19–38. ISBN: 9780128023082. DOI: 10.1016/B978-0-12-802308-2.00003-5. Available https://www.sciencedirect.com/science/article/pii/B9780128023082000035 [accessed 15th Aug. 2023].

Sauro, J. and Lewis, J. R. (2016b) "Chapter 5 - Is There a Statistical Difference between Designs". In: *Quantifying the User Experience (Second Edition)*. Boston: Morgan Kaufmann, pp. 61–102. ISBN: 9780128023082. DOI: 10.1016/B978-0-12-802308-2.00003-5. Available https://www.sciencedirect.com/science/article/pii/B9780128023082000035 [accessed 15th Aug. 2023].

Schmutz, S., Sonderegger, A. and Sauer, J. (2016) "Implementing Recommendations From Web Accessibility Guidelines: Would They Also Provide Benefits to Nondisabled Users". In: *Human Factors* 58.4, pp. 611–629. ISSN: 0018-7208. DOI: 10.1177/0018720816640962.

Schmutz, S., Sonderegger, A. and Sauer, J. (2017) "Implementing Recommendations From Web Accessibility Guidelines: A Comparative Study of Nondisabled Users and Users With Visual Impairments". In: *Human Factors* 59.6, pp. 956–972. ISSN: 0018-7208. DOI: 10.1177/0018720817708397.

Schmutz, S., Sonderegger, A. and Sauer, J. (2018) "Effects of accessible website design on nondisabled users: age and device as moderating factors". In: *Ergonomics* 61.5, pp. 697–709. ISSN: 0014-0139. DOI: 10.1080/00140139.2017.1405080.

Schmutz, S., Sonderegger, A. and Sauer, J. (2019) "Easy-to-read language in disability-friendly web sites: Effects on nondisabled users". In: *Applied Ergonomics* 74, pp. 97–106. ISSN: 0003-6870. DOI: 10.1016/j.apergo.2018.08.013.

Schrepp, M., Kollmorgen, J. and Thomaschewski, J. (2023) "A Comparison of SUS, UMUX-LITE, and UEQ-S". In: *Journal of User Experience* 18.2, pp. 86–104. ISSN: 1931-3357.

Shieber, S. (2014) "Why scholars should write in Markdown". In: available https://www.semanticscholar.org/paper/Why-scholars-should-write-in-Markdown-Shieber/08cfff45a757343c9fc3fe42cef253b11c30a727 [accessed 1st Aug. 2023].

Shocken, S. and Nisan, N. (2017) *From Nand to Tetris - Building a Modern Computer From First Principles*. Available https://www.nand2tetris.org/ [accessed 24th Aug. 2021].

Silver, L., Smith, A., Johnson, C., Jiang, J., Anderson, M. and Rainie, L. (2019) *Mobile Connectivity in Emerging Economies*. Tech. rep. Washington, D.C.: Pew Research Center. Available https://www.pewresearch.org/internet/2019/03/07/mobile-connectivity-in-emerging-economies/.

Sommerville, I. (2019) *Engineering software products*. First edition. Pearson. ISBN: 9780135210642.

Stack Overflow (2023) *Stack Overflow Developer Survey 2023*. Available https://survey.stackoverflow.co/2023 [accessed 1st Aug. 2023].

StatCounter (2023a) *Desktop vs Mobile Market Share Europe*. Available https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/europe/#monthly-201208-202307 [accessed 1st Aug. 2023].

StatCounter (2023b) *Desktop vs Mobile Market Share North America*. Available https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/north-america#monthly-201208-202307 [accessed 1st Aug. 2023].

StatCounter (2023c) *Desktop vs Mobile Market Share North America*. Available https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/africa#monthly-201208-202307 [accessed 1st Aug. 2023].

StatCounter (2023d) *Desktop vs Mobile Market Share North America*. Available https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/asia#monthly-201208-202307 [accessed 1st Aug. 2023].

StatCounter (2023e) *Desktop vs Mobile Market Share Worldwide*. Available https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide/#monthly-200907-202307 [accessed 1st Aug. 2023].

StatCounter (2023f) *Operating System Market Share Worldwide*. Available https://gs.statcounter.com/os-market-share#monthly-200907-202307 [accessed 1st Aug. 2023].

StatCounter (2023g) *Screen Resolution Stats Worldwide*. Available https://gs.statcounter.com/screen-resolution-stats#monthly-200907-202307 [accessed 1st Aug. 2023].

Stikkolorum, D. R., Demuth, B., Zaytsev, V., Boulanger, F. and Gray, J. (2014) "The MOOC Hype: Can We Ignore It? Reflections on the Current Use of Massive Open Online Courses in Software Modeling Education." In: *EduSymp 2014 : MODELS Educators Symposium 2014: proceedings of the MODELS Educators Symposium, co-located with the ACM/IEEE 17th International Conference on Model Driven Engineering Languages and Systems (MODELS 2014)*. Valencia, Spain, pp. 75–86. Available https:

//dare.uva.nl/personal/pure/en/publications/the-mooc-hype-can-we-ignore-it(b0bff0f2-7e02-446e-afd3-33b18ad155cf).html.

Tadesse, S. and Muluye, W. (2020) "The Impact of COVID-19 Pandemic on Education System in Developing Countries: A Review". In: *Open Journal of Social Sciences* 08.10, pp. 159–170. ISSN: 2327-5952, 2327-5960. DOI: 10.4236/jss.2020.810011. Available https://www.scirp.org/journal/doi.aspx?doi=10.4236/jss.2020.810011.

Tan, C. S. S., Schöning, J., Luyten, K. and Coninx, K. (2013) "Informing intelligent user interfaces by inferring affective states from body postures in ubiquitous computing environments". In: *Proceedings of the 2013 international conference on Intelligent user interfaces - IUI '13*. ACM Press, p. 235. ISBN: 9781450319652. DOI: 10.1145/2449396.2449427. Available http://dl.acm.org/citation.cfm?doid=2449396.2449427.

Thüring, M. and Mahlke, S. (2007) "Usability, aesthetics and emotions in human–technology interaction". In: *International Journal of Psychology* 42.4, pp. 253–264. ISSN: 1464-066X. DOI: 10.1080/00207590701396674.

Tomassetti, G. (2017) *A Guide To Parsing: Algorithms And Terminology*. Available https://tomassetti.me/guide-parsing-algorithms-terminology/.

Tullis, T. and Stetson, J. N. (2004) "A Comparison of Questionnaires for Assessing Website Usability". In: available https://www.semanticscholar.org/paper/A-Comparison-of-Questionnaires-for-Assessing-Tullis-Stetson/9d621a71a9c9e24f689f5263ed5b74e53535374a.

U.S. Department of Education (2021) *Digest of Education Statistics*. Washingnton, D.C.: National Center for Education Statistics.

UNESCO (2022) *Education: From disruption to recovery*. Available https://en.unesco.org/covid19/educationresponse. Archived https://web.archive.org/web/20220107074613/https://en.unesco.org/covid19/educationresponse.

Valez, M., Yen, M., Le, M., Su, Z. and Alipour, M. A. (2020) "Student Adoption and Perceptions of a Web Integrated Development Environment: An Experience Report". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, pp. 1172–1178.

ISBN: 9781450367936. DOI: 10.1145/3328778.3366949. Available https://dl.acm.org/doi/10.1145/3328778.3366949 [accessed 1st Aug. 2023].

Vanderheiden, G. (2000) "Fundamental principles and priority setting for universal usability". In: *Proceedings on the 2000 conference on Universal Usability*. CUU '00. Association for Computing Machinery, pp. 32–37. ISBN: 9781581133141. DOI: 10.1145/355460.355469.

Vihavainen, A., Helminen, J. and Ihantola, P. (2014) "How novices tackle their first lines of code in an IDE: analysis of programming session traces". In: *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*. ACM, pp. 109–116. ISBN: 9781450330657. DOI: 10.1145/2674683.2674692. Available https://dl.acm.org/doi/10.1145/2674683.2674692.

Voegler, J., Bornschein, J. and Weber, G. (2014) "Markdown – A Simple Syntax for Transcription of Accessible Study Materials". en. In: *Computers Helping People with Special Needs*. Ed. by K. Miesenberger, D. Fels, D. Archambault, P. Peňáz and W. Zagler. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 545–548. ISBN: 9783319085968. DOI: 10.1007/978-3-319-08596-8_85.

Web Accessibility Initiative (2014) *Involving Users in Evaluating Web Accessibility*. Available https://www.w3.org/TR/2014/NOTE-WCAG-EM-20140710/ [accessed 11th May 2023]. Archived http://web.archive.org/web/20230511002641/https://www.w3.org/TR/2014/NOTE-WCAG-EM-20140710/.

Web Accessibility Initiative (2016) *Accessibility, Usability, and Inclusion*. Available https://www.w3.org/WAI/fundamentals/accessibility-usability-inclusion/ [accessed 4th Dec. 2021]. Archived https://web.archive.org/web/20211204150932/https://www.w3.org/WAI/fundamentals/accessibility-usability-inclusion/.

Web Accessibility Initiative (2017) *Selecting Web Accessibility Evaluation Tools*. Available https://www.w3.org/WAI/test-evaluate/tools/selecting/ [accessed 4th Dec. 2021]. Archived https://web.archive.

org/web/20211120123630/https://www.w3.org/WAI/test-evaluate/tools/selecting/.

Web Accessibility Initiative (2018) *Web Accessibility Laws & Policies*. Available https://www.w3.org/WAI/policies/ [accessed 4th Dec. 2021]. Archived https://web.archive.org/web/20211201202600/https://www.w3.org/WAI/policies/.

Web Accessibility Initiative (2020) *Involving Users in Evaluating Web Accessibility*. Available https://www.w3.org/WAI/test-evaluate/involving-users/ [accessed 4th Dec. 2021]. Archived https://web.archive.org/web/20210513020157/https://w3c.github.io/wai-website/test-evaluate/involving-users/.

Web Accessibility Initiative (2021a) *Involving Users in Evaluating Web Accessibility*. Available https://www.w3.org/WAI/test-evaluate/conformance/wcag-em/ [accessed 4th Dec. 2021]. Archived https://web.archive.org/web/20210101134807/https://w3c.github.io/wai-website/test-evaluate/conformance/wcag-em/.

Web Accessibility Initiative (2021b) *Web Content Accessibility Guidelines (WCAG) Overview*. Available https://www.w3.org/WAI/standards-guidelines/wcag/ [accessed 27th Dec. 2021]. Archived https://web.archive.org/web/20211227183535/https://www.w3.org/WAI/standards-guidelines/wcag/.

Wegge, K. P. and Zimmermann, D. (2007) "Accessibility, Usability, Safety, Ergonomics: Concepts, Models, and Differences". In: ed. by C. Stephanidis. Lecture Notes in Computer Science. Springer, pp. 294–301. ISBN: 9783540732792. DOI: 10.1007/978-3-540-73279-2_33.

Wilson, C. (2009) *User Experience Re-Mastered: Your Guide to Getting the Right Design*. Morgan Kaufmann Publishers Inc. ISBN: 9780123751140. DOI: 10.1016/C2009-0-20682-9.

Wilson, R. J. (2009) *Introduction to graph theory*. 4. ed., [Nachdr.] Harlow Munich: Prentice Hall. ISBN: 9780582249936.

Yu, X., Karsten, E., Kenney, D., Sinha, A. and Brunamonti, C. (2021) *2021 Coursera Impact Report*. Tech. rep. Mountain View, California: Coursera.

Available https://about.coursera.org/press/wp-content/uploads/2021/11/2021-Coursera-Impact-Report.pdf.