PSSOH 2021, DOI: https://doi.org/10.5281/zenodo.5554570

AN ENGINEERING TEXTBOOK TYPESETTING USING SPHINX DOCUMENTATION GENERATOR

Vladimir Milovanović

Faculty of Engineering, University of Kragujevac, 34000 Kragujevac, Serbia
e-mail: vlada@kg.ac.rs

Abstract / Резиме

One of the things that the ongoing Information age has recently brought is reading books on various handheld and portable electronic devices. Novel e-book formats enable inclusion of more interactive contents that is not only generally convenient but is also suitable for engineering and scientific textbooks. Besides the established portable document format (PDF) that presents de facto standard for physical book printing and publishing, e-reader and the Web book editions are becoming more popular. This paper describes a typesetting workflow which is based on Sphinx documentation generator that can produce different output formats out of a single plain text input source code. It is elaborated how the text can be formatted and how different book elements such as figures, diagrams and video clips can be built-in with a special Sphinx directives. However, the accent was placed on embedding source code into the textbook itself and giving readers the ability of copying and executing it with ease. The presented concept is practically demonstrated for an example case of a programming textbook which is made available in three different formats and open under a permissive license. An automated continuous integration and new release delivery is supported and implemented to allow and facilitate quick and widespread book adaption with a hope that electrical engineering and computer science students would be intrigued and that their learning process would be more efficient as well as more productive.

Једна од ствари коју је текуће информатичко доба донело јесте читање књига на разним џепним и преносивим електронским уређајима. Нови формати електронских или е-књига омогућавају укључивање интерактивних садржаја који нису погодни само у општем смислу, већ су надасве прикладни и за уџбенике из инжењерских, научних и техничких дисциплина. Поред такозваног PDF формата који представља дефакто стандард за објављивање штампаних књига, интернет издања, баш као и издања за читаче е-књига постају све популарнија. Овај рад описује поступак слагања текста заснованог на Sphinx генератору документације који из јединственог чисто текстуалног улазног изворног кода може да произведе и врати различите излазне формате. Изложено је како се текст може форматирати и како се различити књишки елементи, као што су слике и дијаграми могу уградити користећи се посебним Sphinx директивама. Међутим, акценат је ипак стављен на уграђивање изворног кода унутар самог уџбеника и на давање читаоцима могућности лаког копирања и извршавања истог. Представљени концепти су демонстрирани на практичном примеру уџбеника програмирања који је направљен тако да буде доступан у три различита формата под допусном лиценцом. Аутоматизован начин за непрекидну интеграцију и испоруку нових издања је подржан и имплементиран како би се омогућила и олакшала широка употреба књиге уз наду да ће студенти електротехнике и рачунарства бити заинтересовани за изложене концепте и да ће самим тим и њихов процес учења бити ефикаснији и продуктивнији.

Keywords: documentation generator, e-book formats, source code, textbook, typesetting.

Кључне речи: генератор документације, е-књига, изворни код, слагање текста, уџбеник.

1 Introduction

The world has changed drastically over the past few decades. Living an everyday life people often take technological gadgets for granted, seldom digressing to observe the long-term impact that the so-called Digital Revolution has brought. The digital, or also by many considered the Third Industrial Revolution which started in the second half of the 20th century simultaneously marked the beginning and has brought the society to the Information Age. Central to the aforementioned revolution is the mass production of integrated and solid-state circuits, themselves based on transistors, and the derived products and technologies such as computers, cellular/mobile and smartphones, as well as the Internet.

Although the Information age, also known as the New Media Age [1], is enabled by the development of semiconductor devices, it is foremost characterized by the onset of use of information technology. This period has also been marked by the expansion of communications with a widespread use of mobile phones and the Internet, which have caused fundamental changes in individuals' personal lives and the whole human society. Generation Z are the first social group that has grown up with access to portable digital technology and the Internet from an infancy and childhood thus commonly being dubbed as "digital natives". Most members of Generation Z are not only children of, but are also being taught in schools and universities, trained in sports activities, and supervised in general by Generation X.

In spite of profound transformations in the overall surroundings between generations X and Z, a rare medium that has not become obsolete and that is still resisting an essential modification is certainly a book. As one of the cornerstones of civilization, books have been around for millennia adapting slowly from clay and wax tables, over papyrus scrolls, up to present day e-books (short for electronic book) which are keeping pace with the current media age. Book selling and collection storing points such as bookstores and libraries also exist, even though website-based versions of both are omnipresent.

Contentwise, common book separation is in fiction and non-fiction ones. While many literary book forms such as novels, poetry and even comic books published today fall in the former broad category, all other material can be included under the umbrella of the latter one. Fiction books today dominantly consist of plain text and optional figures and images whose use ranges from occasional to prevalent such as in comics and graphical novels. Constituting elements of the non-fictional books can generally be more diverse. As opposed to fictional books, it is not seldom that non-fiction books are not intended to be read or studied from cover to cover but can rather serve as a reference, e.g., like a dictionary.

Nearly all academic literature is non-fiction. Besides plain text and images, schoolbooks and textbooks can contain many variations of each, such as mathematical equations and formulas and programming code snippets, or diagrams, figures, (photo)graphs, maps and plots, just to name a few. Since more and more books are published (also) in electronic format, hyperlink referencing that enables convenient and quick referencing and navigation started to accompany and to replace traditional indexes which are becoming less preferred to standard text search. With the prevalence of computers (in their desktop and portable form) and the Internet, some traditional book forms, e.g., like telephone directory, ceased to exist in hard paper version, while other tend to coexist in electronic and traditional formats.

The peculiarities of engineering and scientific textbooks are two-fold. Firstly, besides paragraphs of standard text they usually contain equations, physical formulas as well as an exemplary code snippets that describe actual calculation, method execution or implementation in a command-line interface (CLI) or a (domain-specific) programming language of the tool of choice that is presented. Secondly, these kind of textbooks are often published in smaller volumes (relative to the general public fictional books such as *belles-lettres*) and are hence frequently written and typeset by their authors themselves who are by the rule skillful professionals knowledgable in the typesetting tools of choice, or at least are in possession of above average word processing proficiency levels as compared to general population.

This paper describes a relatively recent documentation generator tool named *Sphinx* and how it can be utilized by the authors to typeset and produce a modern engineering and scientific textbooks.

1.1 Engineering and Scientific Textbook Specifics

Dominance of handheld and pocket-size portable communication and computing devices (with permanent Internet connection) inevitably led to preference of electronic book versions versus old-fashioned physical book entities. There are a number of arguments that speak in favour of e-books. Namely, they are present always, since a personal portable computer or a smartphone is always with the person owning it. Just like other software they do not wear out nor there is any naturally associated copying and multiplication difficulty. If legally allowed, they can be shared among individuals by a single click or a tap. Furthermore, electronic content can be listed and searched much faster. Finally, additional material, appendices, revisions, errata and updates can be much more easily distributed and published in an electronic format. Nevertheless, for many the joy of physical book reading will perhaps never disappear even if conveniences such as battery and power outlet free “operation modes” are put aside.

A prerequisite for any engineering and scientific discipline to attain a high community acceptance is that it must lay on solid theoretical foundations and to be well-understood. These ideas are naturally conveyed using formulas which represent equations, theorems, etc., and common for it is that the underlining fundaments do not change (at least not on a short-term basis). This is one of the reasons why some classical master pieces in engineering and science, e.g., physics [2] or programming, do not lose audience nor popularity with time. Another requirement, especially for applied methods and practical disciplines to reach the full impact is to be supported by well-maintained and mature software libraries and tools which are meant to accelerate, automate and simplify common tasks. Therefore, all textbooks apart from purely theoretical ones by the rule also include exemplar code that should be easy for practitioners to apply, extend, modify and tailor to suit their own needs. In addition to previous specific elements, there will always be unavoidable general illustrations and plain text descriptions.

An all-inclusive engineering and scientific textbook tends to be comprehensive and to incorporate both the theoretical and the practical materials that are often intertwined. While the former part by the rule does not change over the average lifetime of a university textbook, keeping the latter one up-to-date with the latest developments in the field requires major reworks that appear in new book editions and revisions. Actually, documenting the practical tool details, especially when it comes to programming code leans more towards writing a software documentation for which there already exist well-established tools and workflows. Finally, as permissive licenses are gaining in popularity [3] it is not uncommon any longer that someone else but the author himself updates the material. Naturally, the more convenient way for something like that to be accomplished, the more likely it will happen.

The intention to teach the computational and critical thinking skills necessary to formulate problems, the mathematics to solve them, and the tools to practically implement those solutions all in one place presents a formidable challenge. The aim of an engineering and scientific textbook should be to present a unified resource to bring students up to speed and prepare them for smooth industry acceptance or a career in research. Nowadays, there are many excellent textbooks but only a very few of them [4] that are up to date and engaging with the very latest hands-on tutorials. Commonly, such tutorials and code examples can be found elsewhere on the Internet, but are scattered across various blog posts or repositories thus tedious to look for and non-trivial to find. Moreover, such examples typically focus on *how* to implement a given approach, but leave out the discussion of *why* certain algorithmic decisions are made. Also, many resources are hidden behind the paywalls of commercial providers.

To summarize, the goal of a far-reaching engineering or scientific textbook resource should be to (i) be freely available for everyone; (ii) offer engineering and/or sufficient technical depth to provide a good and solid starting point; (iii) include runnable code examples thus showing readers how to solve problems in practice using appropriate tools for the field; (iv) allow and be ready for rapid updates, both by the authors themselves and also by the community at large; and (v) be optionally complemented by some sort of a forum for interactive discussion of technical details and to answer questions. This paper tries to describe technical means centered around Sphinx tool [5] to achieve the above set goals.

The paper is organized as follows: Section 2 presents motivation for use of different book formats and presents limits of the currently available conversion tools, while sections 3 and 4 respectively give a brief workflow description of the Sphinx documentation generator and example case of one textbook.

2 Multiple Textbook Formats and Conversion Tool Limitations

The goals set at the end of the previous section are often conflicting. Namely, from the distribution perspective, textbooks are normally available in the so-called Portable Document Format (PDF), an open file format originally developed by Adobe but now accepted as an international standard (ISO 32000). Vast majority of physical books are sent for printing in PDF and are also stored on computers and computing devices independent of their hardware, operating system or application software that is used to view or edit such documents. Almost all text and word processing software has the ability either to export or directly save its outputs as a PDF file. However, equations, theorems, and citations are best managed and laid out in LaTeX [6] which is a typesetting software system for document preparation widely used in academia [7] for the communication and publication of scientific documents (research preprints, textbook drafts, conference proceedings, journal articles, etc.) in many fields [8], including but not limited to mathematics and statistics, computer science, engineering, physics, and more recently also economics, linguistics, quantitative psychology, philosophy, and political science.

On the other hand the Internet web pages are native in the HyperText Markup Language (HTML) and JavaScript code used for dynamic behavior. Besides the printed book material also downloadable as a PDF, to reach a wider audience as well as to make the material more comfortable to digest from computers and handheld devices, a website textbook version is highly beneficial. Namely, if a book contains any kind of an executable source code it is much easier to copy it than to retype it. Additionally, if possible a direct on-click code execution might be supported especially if an in-browser JavaScript compiler or interpreter exists and if the free and open-source software (FOSS) tools are being exploited. Even though it is certainly not impossible to achieve on-site execution of the code written for some proprietary software, it is much seldomly straightforward to do so due to technical and legal difficulties since non-free closed-source tools are often subject to restrictive licensing terms.

Besides the HTML and the PDF versions, it would be desirable to also support some of e-book file formats that are desirable for e-reader devices and gadgets. E-readers are preferable [9] since they can hold many books limited only by their memory while most of them use e-ink display technology that is not back-illuminated and therefore seem to cause no more eye strain [10] than a traditional book and certainly less eye strain than LCD screens, simultaneously with a substantially longer battery life.

The first idea that naturally comes would be to target either a PDF, i.e., a LaTeX more precisely, or an HTML textbook version and then to perform an automatic conversion to the other formats. No matter how attractive this approach sounds, in reality it is difficult to obtain a high-quality result across all targeted outputs. Specifically, whoever has at least once tried to print a web page can predict the PDF quality and usefulness that is the outcome of the HTML-to-PDF conversion process. The other way around, conversely PDF-to-HTML works even worse, especially for diagrams and math which generally speaking render to be useless. To be honest, using PDF as an intermediate format when converting from LaTeX to HTML is not helpful since in one such conversion much of the structural information is irreversibly lost and cannot be successfully recovered. It is worthwhile considering a direct LaTeX-to-HTML conversion as both are structural markup languages used to describe the document structure, e.g., sections, emphasize, formulas, etc. Perhaps the most comprehensive and universal tools for such tasks are LaTeX2HTML [11], LaTeXML [12], TeX4ht [13] which are all amazing conversion tools but each with its own set of limitations that are associated either with inherent use of device independent (DVI) file format as another type of intermediary, or use of rasterized bitmaps to convert mathematical symbols, diagrams and other “difficult” elements. Generation of easily executable and runnable source code samples are not among the features that can be expected out of any conversion tool.

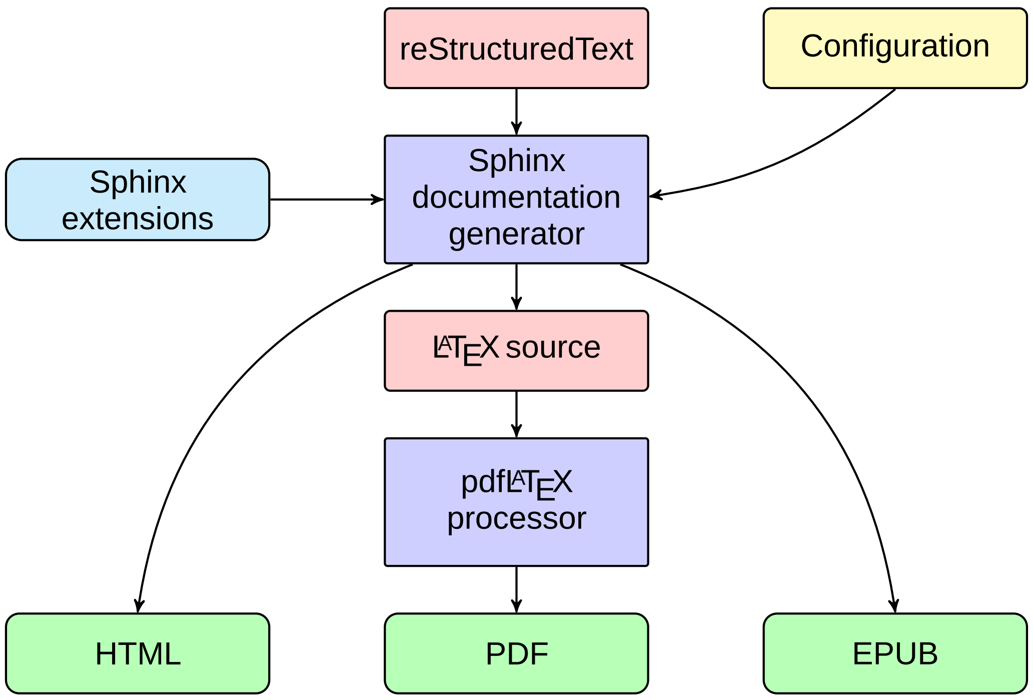

Figure 1: A typical Sphinx workflow as used in the example case for producing three output formats.

3 Sphinx — Python Documentation Generator (to rule them all)

Instead of relying on conversion tools, it would be much simpler, and for that sake much better, to use a single documentation generator to produce every type of aimed output formats. There exist a vast number of documentation generators but only a few that besides HTML can concurrently yield additional outputs such as LaTeX and indirectly PDF, and hardly any with the native e-book support.

One of the most popular documentation generators is Sphinx [5], a tool that makes it easy to create documentation principally for Python and C/C++ projects, but also any other documents consisting of multiple reStructuredText sources. Being part of the Docutils project [14] that was initially made to extract comments and information from Python programs, as well as to format them into various forms of program documentation, reStructuredText [15] is a lightweight markup language (very similar to somewhat younger and more popular cousin Markdown [16]) designed to be easily readable by human programmers and at the same time processable by adequate automatic parsers and translators.

The philosophy of reStructuredText (reST) and its key design goal was the readability, that is, it should be publishable as-is, as plain text, without looking like it has been marked up with tags or formatting instructions, unlike HTML, LaTeX or Rich Text Format (RTF), which for that matter all have obvious tags. Its main inspiration comes from unofficial conventions for marking up plain text e-mails.

Some well-known and already established software projects, like the Linux kernel, transitioned to reST, that has been a core component of Sphinx toolchain, for documentation generation and publishing.

Although Sphinx has been originally developed in 2008 for the Python project documentation, since then it has seen a wide adoption not confined to Python. In its essence, Sphinx converts input reST files into HTML websites and other formats like PDF (via LaTeX) and ePub, a popular vendor-independent XML-based open electronic publication standard. The EPUB format is widely used on electrophoretic display (electronic paper) readers. In addition to reST sources that encompass the actual documentation content, a typical Sphinx workflow accepts a configuration file as well, and just as shown in Figure 1 diagram produces the outputs. Along with the three mentioned outputs of interest, i.e., HTML, LaTeX (for printable PDF versions) and ePub, it can also yield Texinfo, manual pages and plain text.

The inclusion of programming code snippets, notes, figures, images, and other graphical or visual effects and elements is achieved through the use of special directives embedded directly into the reST.

A feature that separates Sphinx is that it comes with a natural code support in a sense that any kind of a programming source code can be included, highlighted and manipulated naturally. For the case of interpreted languages like Python, a built-in prompt and output hiding and showing (in HTML) and automatic output tests and validation through external modules such as `doctest` [17] is also supported.

With a help of external components, also known as (third-party) Sphinx extensions, such as **activecode** from the `runestone.academy` [18], an interactively enhanced electronic textbooks can be created.

There exist more than a dozen built-in and more than a hundred unofficial third-party extensions to Sphinx with which various kinds of special features can be accomplished. Without an ambition to cover extensions in depth and thoroughly, but rather just to give a feeling what can be realized, for example, with a built-in Graphviz [19] extension diagrams of graphs and networks can be directly coded into reST source. Similarly, general figure drawing can be done with a third-party extension that incorporates PGF/Ti*k*Z [20], a widely used pair of languages for producing vector graphic drawings.

Perhaps the nicest feature of all, at least from a perspective of an engineer or a scientist, is that all Sphinx inputs are purely textual and all execution goes directly from a command line. This allows to track changes and to collaborate with others with ease using a (remote) Git repository, as well as to build upon it by assembling a full automation server for workflow execution in a continuous integration (CI) and continuous deployment (CD) manner. The key takeaway is that even if at present day there exists no workflow perfectly suited to address all of the demands placed upon modern engineering and scientific textbooks, the software tools centered around Sphinx as a rendering engine are available, moreover they are FOSS and effortlessly extensible with the built-in, third-party, or self-made add-ons.

4 An Example Case of a Modernized and Pythonic Version of the Classical "Wizard Book" a.k.a. SICP in Serbian Language

One of the best and most influential computer science textbooks of all time is certainly the MIT Press' classic Structure and Interpretation of Computer Programs (SICP) [21] written by Harold Abelson and Gerald Jay Sussman with Julie Sussman, professors of the Massachusetts Institute of Technology. The book teaches fundamental principles of computer programming which are exposed using Scheme [22], a dialect of Lisp. However, due to various reasons there was a shift in the original MIT course 6.001 and it is replaced by a new class which uses Python instead. Despite being the book of programming (as a paradigm) and not of a particular programming language, SICP is not as attractive to Generation Z as it was for previous generations. Since to the best of the author's knowledge, neither a direct SICP translation, nor books that derive from it were available in Serbian language, the author has decided to write a "pythonic" one (and also to fulfill one of the mandatory conditions for academic advancement).

A mitigating circumstance was that the original SICP was legally accessible under the Creative Commons license and that a lot of quality reworked materials were attainable, some [23] even for Python.

Prior to that, the author was teaching fundamentals of programming to electrical engineering and computer science freshmen and was aware of an average university student habits that are also associated with handheld devices, such as constant lack of time, patience, and long-term focus. Therefore, to create a book that would be interesting for a student to scroll through it quickly, grasp as much as possible in the limited interval by either copying example code or executing it inside the browser, thus instantly obtaining the result was a logical decision. Besides HTML that would be residing on the Web [24], both an e-book version and a printed physical book was published [25] for everyone who wanted to have it in a regular textbook format. Naturally, Sphinx was used to produce all the outputs out of a single reST source [26], and the appropriate CI/CD pipeline is made to facilitate future updates.

More specifically, the continuous integration proved to be fairly useful particularly for the Python doctest [17] examples that are automatically executed to verify that they work exactly as shown. Such a system managed to catch some subtle modifications and lack of backward compatibility between different Python versions. On the other hand, the continuous delivery and deployment on every commit trigger a series of actions to execute Sphinx workflow of Figure 1 and produce the three outputs available for download. Major releases, for example, the first edition, are handled with git [27] tags.

Not only the complete textbook source code is available on GitHub [26], but also the workflow for the so-called GitHub Actions which allows building continuous integration and continuous deployment pipelines for testing, releasing and deploying software without the use of third-party platforms and that is, at the moment of writing this paper, free of charge for all public open source projects.

Thus, everyone can fork the repository, change something, for example fix a typo, or add another example or even a complete chapter, and send the appropriate pull request for committing of the made changes into the main branch. Not that the expectation exists or that it is likely, but the ability is there.

Finally, the main beneficiaries of such a textbook should be students themselves. Not only that they should be learning the fundamental principles of computer programming, like recursion, abstraction, modularity, and programming language design and implementation, and not only that they would have the ability to quickly see and exercise practical code samples and snippets, but they could be intrigued by the flow itself. They might get curious how everything is set, and then start learning about version control systems like git, or how reST source code which is available to them gets translated to PDF, HTML, and EPUB, maybe they delve more into LaTeX and thereby raise their overall computer skills.

With the hope that something along the previous lines of thought will actually happen, this textbook is devoted to all of the author's students, past, current and future ones, and also dedicated to his teachers and professors who have enabled him to reach and see this far by "standing on *their* shoulders" [28].

5 Conclusions

One of the characteristics of the ongoing Information age is the exponential publishing growth and new media that has transformed the traditional book industry. New electronic formats have been introduced and book content can be consumed on-line. Most books are published by a small number of very large book publishers, but thousands of authors self-publish their own works. Academic textbooks are no exception. To keep pace with the current developments and stay up-to-date a publication should be available across different platforms (both, electronic and physical) to satisfy the demanding needs.

There are a number of advantages of different electronic book formats which can interleave plain text with formulas, figures and (executable) code, especially in engineering and scientific disciplines. If a single author aim is to support all of them using traditional workflows it would either take prohibitively long time or would not yield acceptable results. Therefore, a sort of an agile software development [29] flow that is centered around Sphinx documentation generator is introduced. With an adequate content description in reStructuredText, one can simultaneously produce: HTML suitable for on-line version, as well as PDF and EPUB which are *de facto* standards for physical book printing and open e-reader format, respectively. Writing books in a software coding style yields a plethora of additional benefits.

The previously mentioned Sphinx-based workflow is exercised on an example case of writing an electrical engineering and computer science textbook. Besides manual workflow execution the process is automated in a CI/CD fashion hence enabling quick releases and snapshots prepared after every change/fix. Additionally, other teachers and professors can more easily adapt (by forking followed by adding new, or discarding and cutting existing material) the manuscript and tailor it to suit their own needs which is enabled by the permissive license. To the best of the author's knowledge, this is the first textbook published in Serbian language that uses such a flow and is provided in the three formats.

Acknowledgement

The author would like to thank professors Nadica Miljković and Predrag Pejović from the University of Belgrade for cordially inviting him to give a talk on this topic at the annual PSSOH 2021 conference.

References

[1] M. Castells, *The Information Age: Economy, Society and Culture*. Oxford: Blackwell, 1996.
[2] R. P. Feynman, R. B. Leighton, and M. L. Sands, *The Feynman lectures on physics*. Addison-Wesley, 1989. [Online]. Available: https://feynmanlectures.caltech.edu
[3] P. Stacey and S. Hinchliff, *Made With Creative Commons*. Ctrl+Alt+Delete Books, 2017. [Online]. Available: https://creativecommons.org/made-with-cc
[4] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.
[5] G. Brandl. (2008) Sphinx — python documentation generator. [Online]. Available: https://www.sphinx-doc.org
[6] L. Lamport, *LATEX: A Document Preparation System*. Addison-Wesley, 1986.
[7] D. E. Knuth. (1992) What are TEX and its friends? [Online]. Available: https://www.ctan.org/tex
[8] A. Gaudeul, "Do open source developers respond to competition? The LATEX case study," *Review of Network Economics*, vol. 6, no. 2, 2007. [Online]. Available: www.degruyter.com/document/doi/10.2202/1446-9022.1119
[9] A. Maxim and A. Maxim, "The role of e-books in reshaping the publishing industry," *Procedia - Social and Behavioral Sciences*, vol. 62, pp. 1046–1050, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042812036191
[10] S. Benedetto, V. Drai-Zerbib, M. Pedrotti, G. Tissier, and T. Baccino, "E-readers and visual fatigue," *PLoS One*, vol. 8, no. 12, 2013. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3873942
[11] N. Drakos, M. Rouchal, J. Lippmann, and R. Moore. (2016) The LaTeX2HTML translator. [Online]. Available: https://www.latex2html.org
[12] B. Miller. (2004) LaTeXML – a LATEX to XML/HTML/MathML converter. [Online]. Available: https://dlmf.nist.gov/LaTeXML
[13] E. Gurari. (2008) TeX4ht – a system for converting documents written in (La)TEX to HTML. [Online]. Available: https://www.tug.org/tex4ht
[14] D. Goodger. (2002) Docutils: Documentation utilities. [Online]. Available: https://docutils.sourceforge.io
[15] ——. (2002) PEP 287 — restructuredtext docstring format. [Online]. Available: www.python.org/dev/peps/pep-0287
[16] J. Gruber. (2004) Markdown — a text-to-html formatting syntax for web writers. [Online]. Available: https://daringfireball.net/projects/markdown
[17] T. Peters. (2001) Python `doctest` module — test interactive Python examples. [Online]. Available: https://docs.python.org/3/library/doctest.html
[18] B. N. Miller and D. L. Ranum, "Beyond PDF and ePub: Toward an interactive textbook," in *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '12. ACM, 2012, pp. 150–155. [Online]. Available: https://dl.acm.org/doi/10.1145/2325296.2325335
[19] J. Ellson, E. Gansner, Y. Hu, and S. North. (1991) Graphviz – graph visualization software. [Online]. Available: https://www.graphviz.org
[20] T. Tantau and C. Feuersänger. (2005) The t*i*k*z* and PGF packages. [Online]. Available: https://pgf-tikz.github.io/pgf/pgfmanual.pdf
[21] H. Abelson and G. J. Sussman, *Structure and Interpretation of Computer Programs*, 2nd ed. MIT Press, 1996. [Online]. Available: http://mitpress.mit.edu/sicp
[22] G. L. Steele and G. J. Sussman. (1975) The Scheme programming language. [Online]. Available: http://www.scheme-reports.org
[23] J. DeNero. (2014) Composing programs. [Online]. Available: https://www.composingprograms.com
[24] V. Milovanović. (2021) Komponovanje računarskih programa. [Online]. Available: https://milovanovic.github.io/krp
[25] ——, *Komponovanje Računarskih Programa*. Fakultet Inženjerskih Nauka, 2021. [Online]. Available: https://github.com/milovanovic/krp/releases
[26] ——. (2021) Komponovanje računarskih programa. [Online]. Available: https://github.com/milovanovic/krp
[27] S. Chacon and B. Straub, *Pro Git*, 2nd ed. Apress, 2014. [Online]. Available: https://git-scm.com/book/en/v2
[28] I. Newton and H. W. Turnbull, *The Correspondence of Isaac Newton: 1661-1675*. Cambridge University Press for the Royal Society, 1959, vol. 1.
[29] K. Beck, J. Grenning, R. C. Martin, M. Beedle, J. Highsmith, S. Mellor, A. van Bennekum, A. Hunt, K. Schwaber, A. Cockburn, R. Jeffries, J. Sutherland, W. Cunningham, J. Kern, D. Thomas, M. Fowler, and B. Marick. (2001) Manifesto for agile software development. [Online]. Available: http://agilemanifesto.org