

Some thoughts about how to write a thesis in the field of Computer Science (to be continued)

Prof. Dr. C. Schommer
Dept. of Computer Science and Communication
University of Luxembourg

August 30, 2011

I have often been asked how to write an academic thesis and what exactly should be in. And I have often become aware that most of the students are not capable of writing sentences in a logical and comprehensive way. It seems that many students lack the simple rules of how to put an idea to paper. In the last resort, things are copied - sometimes 1:1 - or translated from their origin language to English. How awful!

The following lines are therefore an attempt to direct the attention to some simple rules that should be considered. In this regard I must say that this is not a guideline on how to do it - the situations (academic vs. industrial, practical work vs. technical work vs. theoretical work) are too diverse. These lines are more to give some more general hints, which are - in my opinion - applicable to all theses, being a Bachelor, Master, or a PhD thesis. Please note that the following text is my opinion only, so other professors could handle this in a different way!

A piece of work

A thesis is a final piece of work. It typically summarizes the work you have performed in the recent past. Depending on the project, the academic approach, its relationship to theory and practice, this piece of work may be either more scientific or industrial oriented.

The thesis documentation reflects a work that contains a leitmotif. This may be a mathematical hypothesis, a conclusive assumption, or simply a circumstance of software engineering. In the first case, this hypothesis is typically existing a-priori, it is not elaborated but much more a consequence of an existing mathematical problem. Many theses base on that. In the second case, the pros and cons of a specific target are discussed, which then mainly point to existing disadvantages of these approaches. Here, the leitmotif is then to argue that the own approach is somehow 'better'. For the third case, no real problem exists but more the lack of a software solution. This is exclusively occurring at industrial places.

Independent from the work you have done, please notice that for the (independent) reader - who is less familiar with your work - the presentation of the underlying problem, the leitmotif, and the logical explanation plan is an indispensable necessity! As an example, you may think on the the

'w'-questions, but especially "why?". Examples: What is the message of your thesis? Why do you propose your way and not another one? Who are the current key-players in these fields and what do they propose? What are advantages of your system?

Please also note that - depending on the level - the writing of a thesis takes up to 6 months and more! The writing is an evolutionary process, structure and contents change hourly, daily, weekly, et cetera! It is therefore not insignificant to keep a *scientific diary*, where you note important milestones of your work, thoughts, anecdotes, et cetera. It is clever to have this in a written format than to write the minutes from memory!

A proposed framework

The following contains - in note form - a basic structure how I see it. A thesis typically has 6-7 chapters, an index register, and a list of references. Software code should not put in but eventually copied to a CD or placed in paper form on a separate document. This avoids that the actual focus is deflected.

- First, there should be a *frame* with the title of the thesis, the name, the affiliation, the identity, and the date. On one of the next pages do not forget to sign the declaration that you have performed this thesis by you own!
- An acknowledgement is a suited way to express your gratitude to those who have supported you. This is the only place to give subjective comments, it is a place to recognise of and to give gratification. Do not miss it!
- The first chapter concerns with the *general motivation* of your work. Give a brief introduction to the field of work by considering the following questions: what is your thesis about? why is your work interesting? why is it worth to meet the targeted problem? how should an experienced person / ordinary person read your thesis? Do not forget - and that is true for all other chapters - that the chosen language style must be clear and proper!
- The second chapter concerns with the *conditions of understanding*. Are there any specific that must be known by the reader? Are there any penetrative things that must be known beforehand - just in order to understand your contribution? The chapter is interesting above all regarding an interdisciplinary thesis: even in this case, definitions, buzzwords, and "slogans" of the application field (which is typically not Computer Science) must be made. Just to give an example, I am currently advising a doctoral thesis, which concerns the detection of anomalies for satellites. But since I am from Machine Learning but neither an engineer nor someone from telecommunication, I requested some more information, above all a dictionary. ;-)
- The third chapter concerns (more detailed) *existing approaches* that target the same problem? What is the state of the art in the field? What are challenging approaches? If your contribution is part of a superordinate project, please introduce it by a short summary.

We are now leaving the part of the thesis, which concerned the *outside* of the thesis - the work of others- and turn our attention to the *inside*, which is typically your work. From now on, there are some strategies on how to explain the design, implementation, and validation. I am recommending a 'Top-Down'-approach, which yields on the overall architecture first and which is then followed by

the realisation (implementation), the test arrangements, and the presentation of the results as well as (possibly) open problems follow. This sounds a little bit more conceptional than a “mixed-up” or a “bottom-up” strategy.

- The fourth chapter fosters the *architecture of your work*; it introduces the design of your system, approach, and/or theory. This is purely your contribution! Generally, the chapter keeps more abstract, with (almost) no comments about the implementation or the usage of external software/hardware. Eventually, an superordinate project (if it exists, see the comments in chapter 3) can be placed here instead.
- The *implementation details* are *summarised* in chapter five. Here, you explain the major contribution you have made, the core implementation aspects, the reasons why you have chosen which soft- and hardware architecture. Please note: this is not the place of explaining details on ‘how you have made it’ but more the place of presenting the key implementation results!
- Regarding the costs of your implementation, can you give some comments about the costs someone has to invest? Is your software working correctly? Is it complete? Which data structures have you chosen? Which programming language? Which major test results have you got? What was the test arrangement? How did you elaborate the test arrangement? How ‘independent’ are your test results? Can your work been used in other systems/environments?

Now you have presented your work, and it is time to bring the results to light!

- The *Validation and Testing* chapter is sometimes close to the implementation chapter. But just in case, it is different, for example that an evaluation, interpretation, and analysis of the results must be made. This is important, and I recommend you to take some time to think over. Avoid snapshots!
- The *conclusions and future work* chapter concerns with a brief summary of what you have said before (exclusively your main contributions), a critical review of it (disadvantages and advantages, pros and cons), and open problems that still might be there. What can be improved? What are challenges and visions do you see?
- The work optionally closes with an appendix, for example a table of contents (can be placed also to the beginning), a table of figures (dito), an index, glossary (optional), references (mandatory!), et cetera

Some further remarks

- For larger documentations, it is rather more appropriate to select a documentation software like L^AT_EX or T_EX than Microsoft Word, Pages (Mac), or others.
- Always write your thesis in an objective way and avoid personal (subjective) comments! The only place you can do this is in the acknowledgement. As an example, sentences like “the software test could not be performed because the hardware was so awful and the I could not find a suitable software support” must be avoided! A sentence like for example “Because of technical problems, the test could not be performed.” should be used instead.

- Regarding the acknowledgement, you may put in some nice stories/anecdotes from the past. Try to be precise, concrete, and avoid sentences like "...she was very nice because she was there whenever I needed him...": this sounds strange and lets the person be as someone who seldom gave support and advise.
- Regarding the writing style, I always recommend short sentences. If relative clauses must be done, then correctly: directly after the word!
- The usage of figures is highly welcome - if it supports the reader in his understanding. Please note that the pictures should be serious and neither represent "fun" nor "entertainment". Figures in academic documents are typically in a black-white format, but color must be used in case that colored figures are needed, of course.
- Concerning the writing style: there is often the question of using the 'I have shown that', 'we have shown that', or a passive ('it was shown that') way to transfer the information/knowledge to the user. This is hardly to say, I personally prefer the passive version but it is also possible to use these alternatives (however: the usage of 'we' may be misunderstand with respect to the declaration at the beginning of the work - see above).
- In concern of the number of pages: I am often asked for this ("Professor, how many pages do you expect?"); it should be clear that the one and only trigger is the topic of the thesis, and depending on the motivation, conditions, et cetera, the number of pages change, of course. A work of 50 pages can be "better" than a work of 200 pages (and vice versa). Some experience values for students from Computer Science are: Bachelor thesis: approx. 30-60 pages, Master/Diploma approx. 60-100 pages, PhD approx. 80-300 pages.
- In concern of the language select the language by yourself (if it is allowed!) where you are most familiar in or where you think to be familiar in. English is generally the best choice if you want to broadcast your contribution to others. But please take also into account that the thesis must be reviewed by different persons of different language skills!
- Regarding the English syntax and style, consider the advise of a native speaker! British English should be preferred. Avoid abbreviations if possible.
- Do not copy from external sources unless you cite the reference! Always use official sources, which are for example text from academic web-sites, scientific literature in general, especially scientific journals and scientific papers. References from the "red-top press" must be avoided.
- Use the recommendations of your mentor/advisor/supervisor, since he or she has already gone through this process!
- It is recommended to read at least 1-2 similar theses in order to get an idea of a thesis. Maybe, you read some parts of a thesis and ask the writer why he wrote it in this way (if necessary).
- Plan the *last steps* of the project right in time: when to finish (soft and hard deadline), where/when to print, where/when to submit. Is it necessary to have an executive summary in other languages? What is to do afterwards (e.g. presentation, publication, etc.) For PhD: How to publish the thesis at the end? Are there some guidelines?
- Please note that it often takes some time to "switch" from reading to programming to writing and vice versa. Mostly, students are not able to switch from one to the other in a shorter period of time. Often, it takes a while (days, a week), so do not be surprised!

- It is wise to define ‘milestones’ and to make a plan, what to do as overnext. Also, consider breaks to get some distance to the current topic! Check your reading and find the delta (= what could be improved, what must be inserted, is the style good enough, does the reader understand, etc.)

Selected References

Books

- Bui Yvonne N. Bui: “How to Write a Master’s Thesis?”. Sage Publications, 2009.
- Murray Rowena Murray: How to write a thesis? Open University press. 2011.
- Teitlebaum Harry Teitlebaum: “How to Write a Thesis: A Guide to the Research Paper”. Hungry Minds. 1994.
- Wagenen Keith Van Wagenen: “Writing a Thesis: Substance and Style”. Prentice Hall. 1991.

WWW-links

- Brecht Prof. Tim Brecht, R. Cheriton School of Computer Science, University of Waterloo, CA: “A simple approach to thesis writing”. See <http://www.cs.uwaterloo.ca/brecht/thesis-hints.html> (August 2011).
- Covington Prof. Covington, Dept. of Computational Linguistics, University of Georgia: “How to write a term paper or thesis?”. See <http://www.ai.uga.edu/mc/> (August 2011).
- Shoaff Prof. William D. Shoaff, Department of Computer Sciences, Florida Institute of Technology, Melbourne, Florida 32901: “How to Write a Master’s Thesis in Computer Science?”. See <http://cs.fit.edu/wds/guides/howto/> (August 2011).
- Wolfe Prof. Joe Wolfe, School of Physics, University of New South Wales. “How to write a PhD thesis”. See <http://www.phys.unsw.edu.au/jw/thesis.html> (August 2011).
- Horn Werner Hor, Institut für Medizinische Kybernetik und Artificial Intelligence, Universität Wien, Freyung 6, A-1010 Wien, Austria. (August 2011): “Wie schreibe ich eine wissenschaftliche Arbeit?”. See www.leischner.inf.fh-bonn-rhein-sieg.de/PDF/wiss-arbeiten.pdf. (August 2011).
- Fischer Prof. Fischer, Technical University Hamburg-Harburg, Dept. of Operations Research and Information Systems. “Thesis Writing Guidelines”. See www.oris.tu-harburg.de/dateien/oeffentlich/thesiswritingguidelinesoris.pdf (August 2011).