B.Sc. / M.Sc. Workflow, Hints, and Information

Benjamin Schiller
schiller@cs.tu-darmstadt.de

Thanks for part of the content to Stefanie Roos & Thorsten Strufe

August 2, 2012

Abstract

This documents gives a rough overview over the workflow when doing a Bachelor or Master thesis with us. Also, it includes hints on how to find and read papers, how to structure your thesis, and how to properly write down your accomplishments. Furthermore, it contains some information about the tools we use, the accounts you need to create, what information we need from you, and how to proceed when handing in your thesis.

1 Rough timeline of a thesis

A thesis (very) roughly consists of four main parts: reading, solving a problem, implementation, and evaluation. While they build upon each other, there is not always a clear cut between them. Writing your thesis is an ongoing process which you should pursue from the first day, starting by annotating the first paper you read. Giving exact durations to these phases is quite impossible as it highly depends on each thesis and topic. This is why thinking about a rough timetable is the last question of your exposée as described in Section 2.

1^{st} phase - Reading In the first part of your thesis, you get an understanding of its topic and contents by reading and researching the related work, i.e., what other have already done in your field. This gives you a better understanding of the problem we are attempting to tackle and hopefully solve in this thesis. After you have finished this part, you should have a good understanding of the area we are working on. It should then be easy to answer all the questions required for the exposée (cf. Section 2). After finalizing the exposée, we register your thesis at the Prüfungssekreteriat and the official time of 3 months (B.Sc.) or 6 months (M.Sc.) starts.

2^{nd} phase - Solving the problem After understanding the problem we are about to tackle, we start thinking about a solution. This means figuring out how to solve the problem we are investigating or at least parts of it (in order to make the world a better place).

3^{rd} phase - Implementation In most cases, we need to implement the solution or a way to evaluate our solution / test our hypothesis, etc. After figuring out a solution to our problem / a way how to solve it or at least parts of it, you will start to implement means to show that our solution is valid.

4^{th} phase - Evaluation The last phase of your thesis will be the evaluation of our solution. For example, evaluating multiple simulations runs of a system under investigation and drawing conclusions about the investigated system, its parameters, out hypothesis, etc.

Handing in your thesis When handing in the final version of your thesis, we need four printed copies. In two of them, a CD/DVD containing source code, thesis sources, result data sets, etc., must be included. If you want to save some money, you can print the two other copies in black/white.

The deadline for handing in the thesis is determined when registering the thesis.

Defense After handing in your thesis, you will give a presentation (defense) where you describe the contents of your thesis, what you did, what the results are, etc. This should be done roughly four to five weeks after handing in the thesis.

2 The exposée

After the first phase of your thesis, i.e., reading and understanding about your topic and the related work, you are expected to write a short exposée. It is supposed to answer the following seven questions. Answering these questions ensures that you understand the problem we are tackling and that we are on the same page on what is supposed to be done as part of your thesis.

Problem statement The problem (what /exactly/ is the problem tackled in this thesis)

Relevance / motivation Some support for it's relevance (why is it really a problem and why will solving it change the world and make it a better place?)

Background A rough overview over background (explain the basics) and related work (what do others solve and what is missing) (it can help to already roughly sketch the requirements for the system, as well as metrics that can be used to show if a solution is good or bad =¿ especially the metrics that will show that your solution is better than the state of the art)

Our approach A first sketch of an idea how the problem could be solved in the thesis – here the proposal will indicate which ideas the student wants to follow and how they will be implemented

Solving the problem A very short rationale why this solution will solve the problem (and how)

Evaluation An indication how the solution will be evaluated (at best: together with the expected outcome) – here it will indicate how the student will show the superiority (or characteristics) of the solution, will most probably contain two of (mathematical analysis & proofs—simulation—prototypical implementation and measurement study). Keep in mind that you will have to compare your solution to a "baseline", iow. the best existing solution, and that you will somehow need to show this in your thesis (evaluating /both/!) – unless you are the first one solve a new problem and there is no prior art (in which case it is still a sign of quality to come up with a baseline like, e.g., an upper bound (the optimal solution), or a trivial solution).

Timeplan An initial time plan: what are the major steps (and milestones) and roughly during which time will they be dealt with. Should contain "literature review", "initial design phase", "implementation", "implementing the evaluation", "evaluation", "writing", and "final revision and correction". Allow at least 2 weeks for the revision and corrections and at least 4 weeks for the "writing" (but make no mistake: state writing early, you will need at least the four weeks to make a story out of you text snippets, to rearrange content avoiding redundancy and streamlining the logic, etc.

3 Finding and reading papers

In this Section, we give a brief overview over the different publication formats for scientific work. Then, we describe how additional sources for your respective topic can be found and give some tips on how to read papers efficiently.

3.1 Publication formats

The following list gives an overview over the different ways that scientific results are published. This is important because the relevance and impact of publications often depends on their respective format.

- **books / book chapter** - detailed information, mostly very good quality, not necessarily up-to-date

- **journal article** - strictly reviewed, long duration until publication, potentially outdated
- **conference paper** - quality depends on the conference
- **conference poster** - very new but potentially uncertain results / rough ideas
- **workshop paper** - often work-in-progress, quality depends on associated conference
- **technical report** - detailed description of a problem and its solution, not peer-reviewed
- **webpage** - not peer-reviewed, can only be quick notes

3.2 Finding additional sources

In essence, there are three ways to find publications on a given topic:

1. Searching by keywords
2. Backwards referencing
3. Forward referencing

The easiest way is to use the search functionalities of a publication database like *Google Scholar* [1], *ACM* [2], or the proceedings of relevant conferences. Always keep in mind that not everyone uses the same terminology and keywords even though working on the same topic. Hence, always think about different keywords that might point to publications from the area you are working on. The following list includes the most important portals for finding papers:

- ACM - `http://portal.acm.org/portal.cfm`
- IEEE Xplore - `http://elib.tu-darmstadt.de/ieee/Xplore/dynhome.jsp`
- ULB - `http://www.ulb.tu-darmstadt.de`
- Libraries - `http://wwwwbs.cs.tu-berlin.de/bibliotheken`
- SpringerLink - `http://www.springerlink.com`
- Science Direct - `http://www.sciencedirect.com`
- CS Bibliographies - `http://liinwww.ira.uka.de/bibliography`
- DBLP - `http://dblp.mpi-inf.mpg.de/dblp-mirror/index.php`
- Google Scholar - `http://scholar.google.com`
- CiteseerX - `http://citeseerx.ist.psu.edu`
- Zentralblatt Math - `http://www.zentralblatt-math.org/zmath/en/`
- Arxiv - `http://arxiv.org`
- Conference webpages ... [3]
  - always prefer the good conferences: INFOCOM, SIGCOMM, CCS, S&P (Oakland), WWW, Usenix, PETs, STOC, FOCS, ...

When you have found a paper that is relevant for your work, you should also look at its references. Since most paper contain information about related work that has been done before, the bibliography should contain the most important works of the same field. Therefore, it is always a good idea to simply look at the bibliography of relevant papers and investigate the papers that have been referenced often.

The other way around would be to look at the papers that reference (potentially older) publications that are relevant to your work. Most portals provide a list of publications that cite a specific paper, e.g., in Google Scholar under the link *Cited by*.

Very often, you will find a paper with a title that sound very promising and like a perfect fit to your topic. Since not all of these paper are in fact relevant for your work, before printing and potentially reading a paper completely, skim over the abstract to check if the content of the paper really contains what the title (in your context) seems to promise. In case the abstract sound promising as well, read the summary and conclusion next before including a paper into your reading queue.

[1] `http://scholar.google.com`
[2] `http://portal.acm.org/portal.cfm`
[3] `http://en.wikipedia.org/wiki/List_of_computer_science_conferences`

3.3 Reading papers

Reading papers:

- decide on a set of questions you want to be answered by this paper
- write down/mark the answers to those questions + other interesting stuff you find
- relate the paper to others, categorization
- you can (but don't have to) use tools for organizing literature (e.g. Mendeley `http://www.mendeley.com/`)

Writing the final version:

- decide on structure:
  - abstract, introduction and conclusion are necessary
  - possibly a background section introducing concepts necessary to understand the actual content: notation, mathematical concepts and theorems, basic terms
  - the remaining structure should classify the papers you read in some way
  - most cases: one general section introducing the different categories and one (sub)section per category
  - alternative: discuss analytical ideas in one section, practical evaluation in a second section
  - alternative: discuss ideas in one section and results in a second
- write each section down in bullet points usually starting with the 'content sections', organise bullet points into groups, so that each group contains one basic thought (which will later be one paragraph)
- write down the full text
- several iterations of corrections:
  1. structure, overall content (ideally before writing down the full text!)
  2. content, sense, understanding
  3. grammar and spelling

LaTeX:

- compile regularly, at latest after each paragraph, after each (long) mathematical equation and whenever you try something new
- behavior varies between different operating systems, LaTeX distributions and Editors, so make sure you know your system setting when googling for help
- Guide: `http://en.wikibooks.org/wiki/LaTeX`
- Forum: `http://www.latex-community.org/forum/`

4 Outline of a thesis

4.1 Outline for a computer science paper

From a structural point of view, B.Sc. and M.Sc. thesis are very similar to published papers. The following list gives an overview over a good outline for a computer science paper (according to Al Bundy) [4].

Title - ideally the title should state the hypothesis of the paper

Abstract - state hypothesis and summarise the evidence that supports or refutes it

Introduction - motivate the contribution!

Literature Survey - broad and shallow account of the field, rival approaches, drawbacks of each, major outstanding problems

Background - states previous work in more detail, where this is necessary for understanding

Theory - underlying theory, definitions, theorems etc.

[4] `http://homepages.inf.ed.ac.uk/bundy/how-tos/resbible.html`

Specification - requirements and specs of implementation

Implementation Evaluation Related Work - narrow but deep comparison with main rivals

Further Work Conclusion - summarize research, discuss significance, restate hypothesis and the evidence for and against it, - recapitulate original motivation, reassess the state of the field in the light of this new contribution

Appendices

4.2 Plagiarism

- `https://www.informatik.tu-darmstadt.de/de/sonstiges/plagiarismus/`
- `http://www.williamstallings.com/Extras/Writing_Guide.html`

4.3 Abstract

The abstract of a thesis consists of the following four points (one paragraph each):

1. Problem statement: What is the problem we are trying to solve? (One problem is more than sufficient for a thesis!)
2. Relevance: Why is this problem really a problem?
3. Response: What is our solution to the problem?
4. Confidence: how do we show in this thesis, that our solution is good?

4.4 Introduction

- Broad Topic, potentially little broad background
  - topic, special problem we're looking at, motivation
  - probably more background for our problem (why is it actually hard?)
  - broad background, general definitions
- our goal and our claims (what are we solving in this work?)
  - Our goal, research question, motivation and relevance (Why is it a problem the reader should care about? Why is it hard?)
- Requirements for our solution
  - Requirements for a good solution
- Which metrics can we use to show the quality/quantity of our solution?
  - probably rough definition of metrics
  - Metrics to measure how good a solution is
- If space missing the related work may be presented in a paragraph here
- Summary of our solution
  - Overview of our solution and first confidence (how do we show that it's good?)
- Our contributions in this paper
- outline of the paper / reader's digest
- Related work (can be done together with literature survey)
  - STATE HOW THE RELATED WORK RELATES TO YOUR WORK!! (how is it similar, how is it different?)
  - Related work is not your enemy, but gives you "the shoulders of giants" you can stand on
  -
- –
-
- reader's digest

4.5 Background

- Literature Survey and Background
-
-
-

4.6 Theory (probably)

-
-
-
-

4.7 Specifications

-
-
-
-

4.8 Implementation

-
-
-
-

4.9 Evaluation

-
-
-
-

4.10 Summary & conclusion

-
-
-
-

5 Some tips on writing

5.1 General tips on scientific writing

- `http://homepages.inf.ed.ac.uk/bundy/how-tos/writingGuide.html`
- `http://www-net.cs.umass.edu/kurose/writing/`
- `http://www.cs.columbia.edu/~hgs/etc/writing-style.html`
- Book: *Zen - or the art of motorcycle maintenance, Robert M. Pirsig*[5]
- Book: *The craft or research, Wayne C. Booth et al.* [6]

[5] `http://en.wikipedia.org/wiki/Zen_and_the_Art_of_Motorcycle_Maintenance`
[6] `http://en.wikipedia.org/wiki/The_Craft_of_Research`

5.2 Structural hints

The following lists gives a rough overview on how to structure sections, paragraphs, and sentences and how to deal with different levels of abstraction.

Sentences A sentence presents a single statement, e.g., $x = y$, *red is not blue*, $z < 1 \Rightarrow x = y$.

Paragraphs A paragraph contains all statements belonging to the same idea or concept. For example, $x > 2$ and $y = 7 \Rightarrow x > 10$ should be grouped in a single paragraph. Separate paragraph on the other hand should be used for statements like *"the weather is nice"* and *"elephants are huge"*.

Sections A section combines all statements that belong to the same idea / line of thoughts. In most cases, this implies that they are on the same level of abstraction. Subsections should logically divide the content of a section and build upon each other, e.g., "preliminaries for X", "approaches for X", "new ideas".

Abstraction levels Inside a paragraph, always stay on the same level ob abstraction, e.g., design, specification, implementation. In case you change the level of abstraction inside a section, do not jump between these levels. Either move from a high level of abstraction to a low level or the other way around (e.g., design → specification → implementation or implementation → specification → design).

Reader's digest The first part of every section summarizes what the content of this section is, e.g., "In this Section, we motivate our work, describe the problem space, and solve the problem". This allow the reader to quickly get an idea about the content of a section.

6 Writing proper English

In general, it is up to you to choose the language of your thesis. Since basically everything in computer science is done in English and there are many technical terms that do not have German translations, I strongly recommend to write in English.

A very good and short book with basic and very useful rules and hints is *The Elements of Style* by William Strunk Jr. and E.B. White [7] (cf. Figure 1). All basic rules are available in various online resources [8]. But for a price of less than 10 Euro, I consider it a good investment [9].

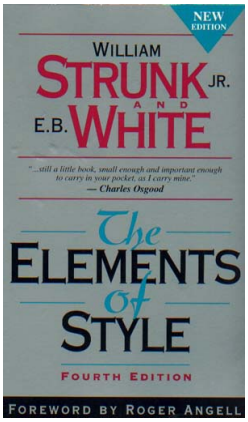

Figure 1: The Elements of Style

[7] http://en.wikipedia.org/wiki/The_Elements_of_Style
[8] http://www.cs.vu.nl/~jms/doc/elos.pdf
[9] http://www.amazon.de/The-Elements-Style-E-White/dp/020530902X/ref=sr_1_1?ie=UTF8&qid=1341214949&sr=8-1

6.1 Some rules

The following list contains some important rules (some of them included in Strunk's basic rules) which are disregarded quite often.

- *my father's dog*, *the inhabitants' opinion*, *Hans's car*
- *it's* is an abbreviation for *it is* (should not be used!) and differs from *its*, the possessive pronoun
- *a, b, and c / a, b, or c* in clear contrast to the German *a, b und c / a, b oder c*
- *i.e.* (id est) does not introduce an example but a clarification
- *e.g.* (exempli gratia) introduces an incomplete list of examples (can be only one), never to be ended by etc.
- *cf.* (confer) is used to compare a statement to another part, e.g., a Figure presenting evidence for your statement (do not use cp.)
- i.e. and e.g. are always enclosed in commata: *my father, i.e., the guy over there* (some style guides only mention the first comma)
- *Figure 2.3, Listing 1.2, Table 8.2.3, Section 4.3, Chapter 1*, etc. must be capitalized (basically every term that indicates / is followed by a reference to another part of the document)
- there must be a reference to every table, figure, etc. that is included in your thesis, unreferenced elements need to be removed (or referenced)
- figures, tables, listing, etc. are not the carrier of information, they are only used for clarification and details that are not required for the overall understanding of what you are talking about. the crux of what you are talking about should always be contained in the written text so that figures could be removed without loosing the basic meaning of what you want to express
- *we* are explaining, performing, evaluating, etc. (do not use *I* or passive statement, like, e.g., *in Section 2.3, XYZ is explained*)
- stick to present tense, e.g., *we describe* instead of *we will describe*, *we evaluate XYZ* instead of *we evaluated XYZ*
- do not start sentences with *To ...*, use *In order to ...* or a different statement instead
- when writing about a person / user in general, do not use *he*, use *she* instead
- *we proof* is in most cases wrong! except for theoretical proofs, we basically never proof something. especially when talking about the results of measurements, you should never state that *these numbers proof that XYZ*. normally, our measurements (represented by figures, tables, etc.) *show that in this case (in our setup), the system has property abc which indicates that XYZ.*
- do not use emotional adjectives to describe scientific facts
- *large, giant, tiny, small, fast, slow* are not a precise description of the respective property, be as precise s possible when stating properties, e.g., of systems

6.2 Some resources

Some interesting (and potentially funny) resources for writing proper English:

- `http://theoatmeal.com/comics/ie`
- `http://theoatmeal.com/comics/misspelling`
- `http://theoatmeal.com/comics/semicolon`
- `http://theoatmeal.com/comics/apostrophe`

7 Accounts

In order to comply with the infrastructure of our group / that our group uses, you need to create some accounts to allow for a seamlessd workflow.

7.1 Github

Version control is very important when developing any kind of software. Because of the many drawbacks of SVN, I decided to use git exclusively. All projects are currently hosted on Github [10]. In order to use this infrastructure, you need to create an account. The most important feature of Github is the use of issues allowing for a very slick workflow for reporting, discussing, and fixing errors as well as feature requests (cf. Figure 2).

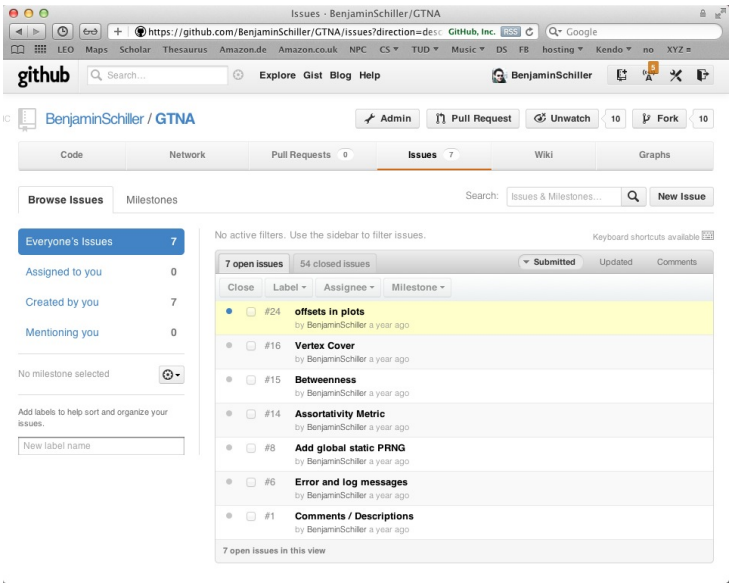

Figure 2: Github issues for a repo

If you are developing for GTNA [11], please follow the instruction provided in the repo to create an own repository as a fork of the main project [12].

In case you are starting a new project, I will create a repo for you in the Github organization of our group [13]. Please create the following branches for storing the respective data:

- *master* - code (of course other branches as well)
- *doc* - documentation files for the code
- *docs* - documents like lists, pdfs, etc.
- *thesis* - you thesis (latex code, images, other sources, no binaries except for thesis.pdf)

Please refer to the following guide on how to create an empty branch that does not branch from the master [14].

Please note that you can register as a student on Github [15]. This will give you a free micro plan (instead of paying the 7$ per month) which allows you to create five private repositories [16].

7.2 Mendeley

Mendeley [17] is basically a social network for researchers. But more importantly, it is a great way to organize articles, papers, etc. that you find and read during the first part of your thesis. Besides a web interface, *Mendeley* offers desktop applications in Windows, Mac, and Linux that allow you to organize papers, annotate them, and add notes to summarize their content and write down important facts about a paper you read (cf. Figure 3).

[10] https://github.com/
[11] http://www.p2p.tu-darmstadt.de/research/gtna/
[12] https://github.com/BenjaminSchiller/GTNA/blob/master/CONTRIBUTING.md
[13] https://github.com/organizations/P2PNetworksTUD
[14] http://www.bitflop.com/document/116
[15] https://github.com/edu
[16] https://github.com/plans
[17] http://www.mendeley.com/

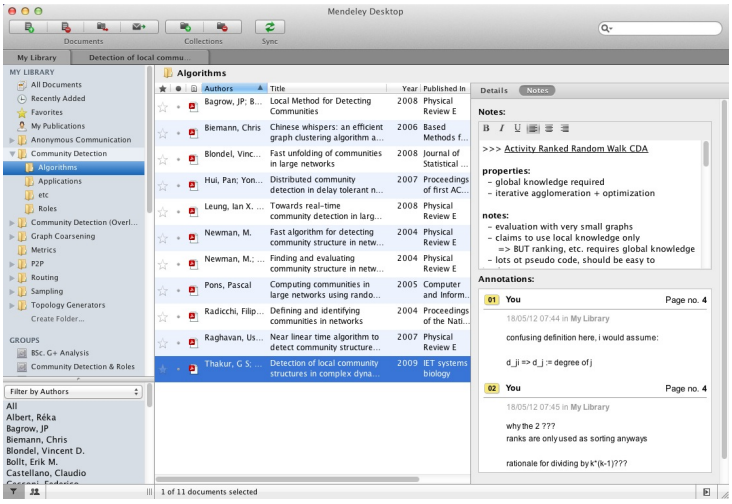

Figure 3: Mendeley Application on the Mac

During the first phase of your thesis, you will read a lot of papers. *Mendeley* can help you to organize them and also store the information you found about the papers' content. In order for me to see the paper you found and also see your annotations, please create a private (important!) group in Mendeley and invite me to it (you need to invite me with the address `benni@codingz.de` for it to work). Besides annotations, I highly encourage that you add a quick summary of the paper in the notes secion for each paper you have read. This can be as short as explaining parts of an algorithm in detail or quickly describing why the paper is not really interesting for your topic. This note should always include what the problem is the authors address (e.g., build the perfect Darknet) what is done in this paper (e.g., a new algorithm for XYZ), and how their solution is evaluated (e.g., simulation, theoretical, etc.).

While there are also mobile application for Android and iOS devices, they don't allow for annotations on the respective device. Since I like reading on the iPad and want to have digital annotations in the papers without transferring them in the Mendeley desktop application later, I use the following workflow for reading papers (iOs of course, but should work with similar applications for Android):

1. Create a *README* folder in *Dropbox* [18] [19]
2. Organize and categorize papers in subfolders
3. Import the *README* folder with *GoodReader* [20] (and sync regularly)
4. Open, read, and annotate the paper with *GoodReader's* annotation tool
5. Sync annotated paper back to*Dropbox*
6. Add paper from *README* to *Mendeley* using the desktop application

7.3 Wunderkit

Managing tasks for a thesis and organizing the notes during and after meetings is quite annoying. While sending emails is ok, finding and organizing the information in a suitable fashion is quite impossible. A nice tool that allows the collaborative administration of notes and tasks is *Wunderlist* [21] from *6 Wunderkinder* [22].

Currently, standalone applications are only available for Mac and iPhone (cf. Figure 4). But there is a nice web application that offers the same functionalities for non-Mac users.

In order to manage notes and tasks for your thesis, I create a new project in *Wunderkit* and invite you to it. This way, we can edit notes together, create new tasks, and complete them.

[18] `https://www.dropbox.com/`
[19] `http://itunes.apple.com/us/app/dropbox/id327630330?mt=8`
[20] `http://itunes.apple.com/us/app/goodreader-for-ipad/id363448914?mt=8`
[21] `http://wunderkit.com/`
[22] `http://www.6wunderkinder.com/`

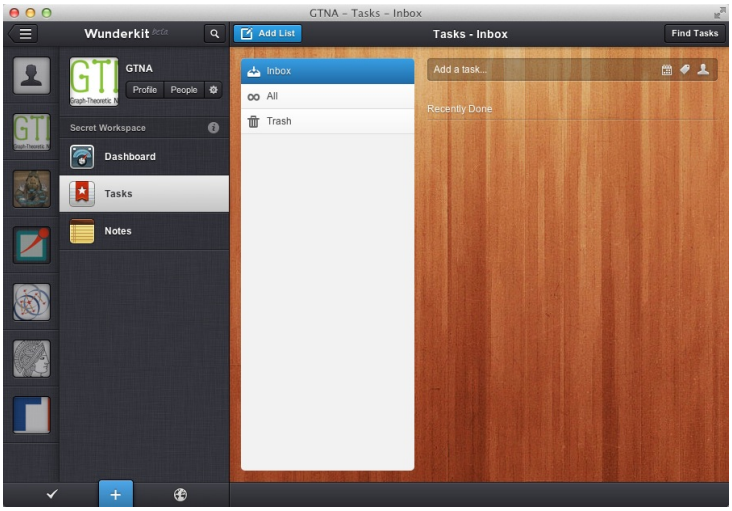

Figure 4: Wunderkit Application on the Mac

7.4 TK VPN

In case you are going to perform computations on one or multiple of our machines, you will need to be able to connect to them. Most machines are configures to only accept SSH connections from IP addresses from the TUD's range or even more restricted to our own pool of addresses. Therefore, it is necessary that you obtain a TK account allowing you to use the TK VPN.

The settings for connecting to TK's VPN are as follows:

- Server: *vpn.tk.informatik.tu-darmstadt.de*
- Username: *TK account*
- Password: *TK password*
- Domain: *TK*
- Type: *PPTP*

On Mac I have an option to send all traffic over the VPN connection. Assuming the existence of a similar option for other operating system, I advise you to NOT check this box. Since the VPN is only required to have a TU IP address for the communication with our servers, other traffic should not be sent over the VPN which would be rather slow!

7.5 BigCluster

In order to perform computation, you will most probably get an account on one of our machines, called *BigCluster*. The *BigCluster* is a blade server (HP ProLiant DL385 G7) with the following rough specs (cf. Figure 5):

- IP: *130.83.163.152*
- Username: *your username*
- Password: *your password*
- Requirement: IP address from TK's pool (e.g., using TK VPN)
- Processors: *(x2): AMD Opteron Modell 6174 (12-Core, 2,2 GHz, 12 MB L3) each*
- RAM: *16GB + 96GB (8 x 2GB + 12 x 8GB)*
- HDD: *3x500 GB SCSI (Controller: Smart Array P410i). RAID 5*
- Optical drive: *DVD/CD-RW*
- OS: *Debian Squeeze*

Most of the time, multiple students and members of our group are performing computations on the machine. We allocate resources on demand. After talking to the others, I will simply tell you how many CPUs and how much RAM you can use.

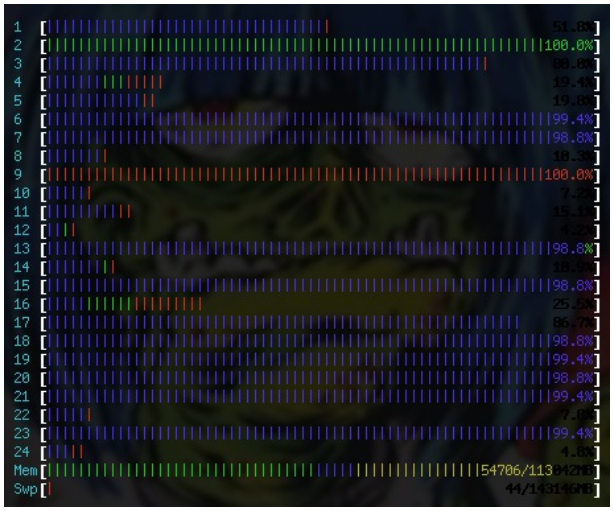

Figure 5: htop on *BigCluster*

7.6 IM

Often, it makes a lot of sense to be able to quickly communicate using one of the many instant messaging tools out there. Therefore, please add me in one of the following IM clients:

- Jabber: *benjamin.schiller@jabber.ccc.de*
- ICQ: *104659547*
- Skype: *badtaste232*

8 Emails & filenames

Whenever you write an email regarding your thesis, please use a subject in the form as shown in equation 1.

$$\$\{subject\}\ \ (\{B.Sc.|M.Sc.|Sem.\}\ \ \$\{thesisTitle\}) \tag{1}$$

When attaching files to an email, like, e.g., a current version of your thesis, please name them in the form as shown in 2.

$$YY\text{–}MM\text{–}DD\text{–}\$\{filename\} \tag{2}$$

9 Required actions / information

The following list contains everything you need to prepare before starting your thesis:

- Signup for an account on Github / Mendeley / Wunderlist
- Create a private (!) group in Mendeley and invite me to it (use *benni@codingz.de* for the invitation!)

The following list contains all information I need from you before starting your thesis:

- Name
- Email
- Student ID (Matrikelnummer)
- Course of studies (Studiengang)
- Current term (Fachsemester)
- Study and examination regulations (Prüfungsordnung)
- Github / Mendeley / Wunderkit account name (or email address used for registration)
- IM account name(s)
- main operating system

A LaTeX

Some hints and pointer for the use of LaTeXto write your thesis.

always add tudtemplate to your thesis!

A.1 Thesis template

A.2 Including images in LaTeX

In order to include graphics into your thesis, you need to use the package *graphicx* (cf. Listing 1). Using the *graphicspath* and *DeclareGraphicsExtensions* you can specify default locations where your images are located as well as the default file extensions (they are used in the order indicated in the list). If both properties are set as shown in Listing 1, you can simply use *my-awesome-image* instead of *images/my-awesome-image.jpg*.

```
\usepackage{graphicx}
\graphicspath{{images/}{./}}
\DeclareGraphicsExtensions{.pdf,.png,.jpg}
```

Listing 1: 'Packages and settings for including images'

A.3 PDF vs. PNG / JPG / etc.

It is always better to use PDF vector images for any kind of figures because of their vector-based representation. This includes images of workflows and architectures that you create as well as plots of data generated using gnuplot. Especially for printing your thesis, vector-based formats should always be preferred. Depending on the kind of data you are plotting using gnuplot or similar software, these images can contain a large amount of data point. Hence the rendering of the pages can take quite a long time when browsing the document in a pdf viewer. Therefore, I recommend that you create two versions of all generated plots: a *.pdf* version for printing the thesis and a *.png* or *.jpg* version for the digital version of your thesis. As indicated in Listing 1, you can specify the extensions of graphics and hence refer to images without their extension, e.g., *my-awesome-image* instead of *my-awesome-image.pdf*. In order to use *png* instead of *pdf* versions of the created images, you only need to change the order of the extensions int this setting as shown in Listing 2.

```
\DeclareGraphicsExtensions{.png,.jpg,.pdf}
```

Listing 2: 'Packages and settings for including images'

B Defense

B.1 Template

B.2

C Gnuplot

D Always keep in mind...

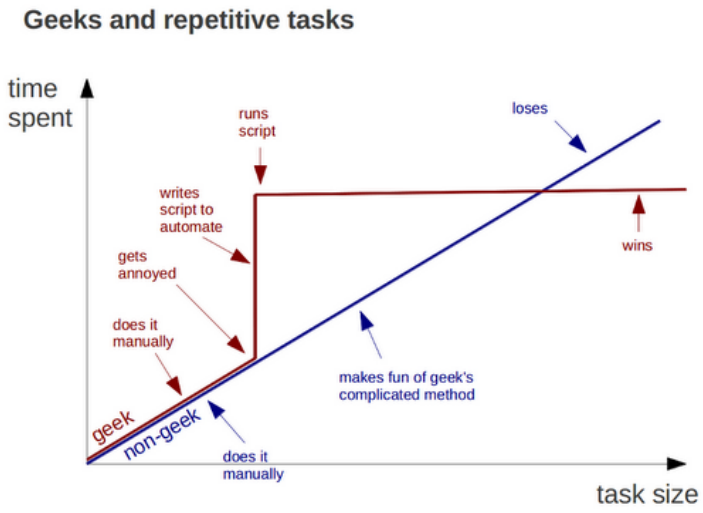

Figure 6: Sometimes it is beneficial to be a geek ;-)