# What is an MSc *"Thesis"*

J.N. Oliveira [1]

[1]DIUM and HASLab, University of Minho, Portugal


22nd January, Room CP2-102
Braga, Portugal

# Preamble

# Context: Learning cycles

BSc — 1st cycle: student expected to learn and apply **general**, well-established theories

*The "repeat" phase*

MSc — 2nd cycle: student expected to learn **specialized** theories and build solutions from them

*The "build" phase*

PhD — 3rd cycle: student (who thinks she/he can do better than his former teachers) expected to pursue a new **conjecture** (thesis) and provide scientific evidence of it

*The "create" ("invent") phase*

# Mind the terminology

MSc, PhD — post-graduation academic **degrees**

MSc, PhD thesis — a scientific **result** (from the Greek $\theta\varepsilon\sigma\iota\zeta$ = position)

MSc, PhD project — an action, **initiative** taking time (from the Latin *proicere* = throw forth)

MSc, PhD dissertation — a piece of **text**, originally a *discourse* (from the Latin *dissertatio* < *disserere* = discuss)

# Doing a post-graduation course — doing "science", ok?

- Post-grad **projects** are a standard way of advancing human **knowledge**.

- Post-grad **programmes** range over the
    - human (social) sciences
    - natural sciences
    - exact sciences.

However, what does **"science"** mean? What tells science apart from other forms of human knowledge?

- Post-grad students cannot ignore these questions!

# Overview of the Scientific Method

# Science? Pre-science?

In an excellent book on the history of scientific technology,

> *"How Science Was Born in 300BC and Why It Had to Be Reborn" (Springer, 2003),*

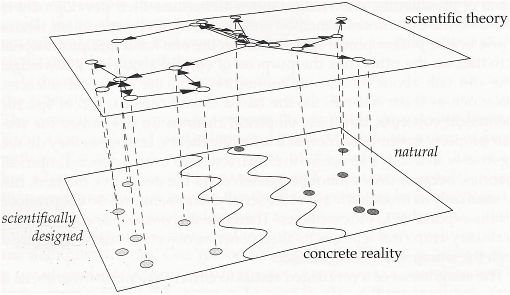
Lucio Russo writes:

> *The immense usefulness of* **exact** *science consists in providing* **models** *of the real world within which there is a guaranteed method for telling false statements from true. (...) Such models, of course, allow one to describe and* **predict** *natural phenomena, by translating them to the theoretical level via* **correspondence rules**, *then solving the "***exercises***" thus obtained and translating the solutions obtained back to the real world.*

Disciplines unable to build themselves around *"exercises"* are regarded as **pre-scientific**.

# Scientific engineering ($e = m + c$)

Also from Russo's book :




Vertical lines mean **abstraction**, horizontal ones mean **calculation**:

engineering = model first, then calculate
($e = m + c$)

# Example

- **natural phenomena** — planetary motion, objects falling down
- **correspondence rules** — Newton (1642-1727)'s laws of mechanics and gravitation stemming from **model**

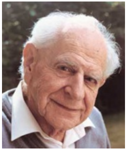
$$F = G\frac{mM}{d^2}$$

- **"exercises"** — Earth gravitational field,

$$g = \frac{GM}{R^2}$$

  then $F = gm$, then $F = m\frac{dv}{dt}i = ma$, then. . . (you know the rest!)
- **translation back to the real world** — ballistics, space missions, satellite technology, etc

# Where does it all begin?



K. Popper (1902-94)

Following the eminent philosopher of science of the 20c Karl Popper (1902-94), science does not arise from **observation** or **inductive** perception of reality only.

**Scientific theories**, and human knowledge in general, are conjectural or hypothetical, and are generated by **creative imagination**.

This links **science** with **art**.

It means that æsthetic attributes such as **beautiful**, **elegant**, **horrible**, **ugly**, etc apply to science.

Beware: this applies to research work as well!

# Besides imagination, which other skills?

Abstraction! — Quoting Jeff Kramer [1]:

**Abstraction** *is widely used in other disciplines such as* **art** *and* **music**. *For instance (...) Henri Matisse manages to clearly represent the* **essence** *of his subject, a naked woman, using only simple lines or cutouts. His representation* **removes** *all detail yet* **conveys** *much.*

**Economy** *of expression.*


[1] *Is Abstraction the Key to Computing?*, CACM 50:4, pp. 37–42, Apr. 2007.

# Computer science — back to 40 years ago

Phrase **software engineering** seems to date from the Garmisch NATO conference in 1968:

> *In late 1967 the Study Group recommended the holding of a working conference on Software Engineering. The phrase 'software engineering' was deliberately chosen as being* ***provocative****, in implying the need for software manufacture to be based on the types of* ***theoretical foundations*** *and practical disciplines, that are traditional in the established branches of engineering.*

Question:

- Provocative or not, how "scientific" do such foundations turn out to be, 40 years later?

# Complexity, complication, obfuscation

Software engineering (SE) is complex:

- **Complexity** — property of being intricate but with formalizable structure.

Negative aspects of software engineering research:

- **Complication** — messy, lacking structure
- **Obfuscation** — formalization intended for bewilderment rather than enlightening (worst of all).

So — in your project:

- Don't expect an easy task
- It will be **complex** — so, don't **complicate** it further.
- Never dare going into obfuscation!

# Planning your dissertation

# What is involved

Questions:

- **How** should I structure it?
- When should I start?
- What should I write?

Likely questions, aligned with the so-called **Aristotelian categories**:

> *Wherever your are, whatever you do, your ideas, concepts, "things" etc. are multidimensional in nature:*

| | |
|---|---|
| What | *the thing is about* |
| What for | *the purpose of the thing* |
| Why | *bother with the thing* |
| When | *did the thing happen?* |
| Where | *is the thing taking place?* |
| How | *is/was the thing carried out?* |

# What is it?

Recall that:

- A dissertation is a **document** which should provide scientific evidence of some result(s) in some area of knowledge
- Following the **scientific method**, the concepts involved in such results should be **formalized** first (vertical arrows in Russo's diagram) and then **reasoned** about (horizontal arrows in the same diagram).

This entails some structure in the text:

- **Definitions** for each correspondence rule (in Russo's sense)
- **Theorems** for each "exercise" (in Russo's sense).

What about the overall text?

# **How** should I structure it?

Recall the typical structure of a mathematical argument, leading to results in the form of **theorems**, each involving:

1. Thesis ($T$)
2. Hypothesis ($H$)
3. Proof ($H \Rightarrow T$)
4. Corollaries
5. Lemmas
6. Others' theorems.

# **How** should I structure it?

Since the purpose of a dissertation is that of providing scientific evidences, its **overall structure** should mirror the shape of a mathematical argument. Here it goes:

| **Maths** | **R&D** (parallel) | **Dissertation** |
|---|---|---|
| Thesis ($T$) | Main result | Contribution chapter |
| Hypothesis ($H$) | Context | State of the art [2] |
| Proof ($H \Rightarrow T$) | Evidence | Core chapters |
| Corollaries | Application | Case studies |
| Lemmas | Support results | Appendices |
| Others' theorems | Evidence elsewhere | Bibliography |

So, in a sense, writing up your dissertation means *proving* **your** *"theorem"*.

[2] Inc. previous work.

# How should I structure it?

Therefore, it's no wonder that a dissertation should be structured as follows [3]:

- Introductory material:

  1st Chapter — Context, motivation, main aims
  2nd Chapter — State of the art review; related work
  3rd Chapter — The problem and its challenges.

- Core of the dissertation:

  4th Chapter — Main result(s) and their scientific evidence
  5th Chapter — Application of main result (examples and case studies)
  6th Chapter — Conclusions and future work.

[3] Number of chapters not strict: may vary according to the needs.

# How should I structure it?

- Auxiliary material:

  Bibliography — List of works referred to in the main text
  Appendix A — Support work (auxiliary results which are not main-stream)
  Appendix B — Details of results whose length would compromise readability of main text
  Appendix C — Listings (should this be the case)
  Appendix D — Tooling (should this be the case)

This should be complemented by some extra matter, as in the following slide.

# How should I structure it?

1. Front matter:

   Title page — institutional, as a rule
   Abstract page — summary of the work
   Acknowledgements — thanks to tutors, colleagues, institutions (funding), etc
   Table of contents — overview of the whole document
   Glossary — list of acronyms and their meaning
   Lists — of tables, of figures etc (automatically generated if using a proper authoring system).

2. Rear matter:

   Index of terms — index of mentioned entities, with references to where (page numbers) they occur in the text.

# How should I structure it?

Last but not least:

- Don't **nest** your dissertation too much (Dewey Decimal Classification works against you if you do so)
- A chapter is not a section (**length**!)
- Each chapter can be regarded as a *mini-dissertation* (thus it shares, in a sense, the same structure — introduction, summary at the end [4], etc)
- Don't forget to **spell check** the whole document!
- **Symmetry** — introduction and conclusions should be "*matching* parentheses" (check at the end)
- **Aesthetics** — style, elegance and design alone are not enough, but help.

[4] Introduction chapter excluded, whose summary should be an overview of the structure of the dissertation.

# Writing up

# When should I write it?

- You should start writing up your thesis on the **very first day** you start your project.
- Of course, this assumes you've understood your project theme sufficiently well.
- On that day only a **sketch** of the dissertation can be written — but already mentioning the standard chapters.
- Use this skeleton as a **road map** and **diary** — you can always keep auxiliary information in the form of comments.
- Comments may even include **time stamps** — these will tell how fast you've done your work (useful in measuring effort and productivity).

# **Whom** should I write it for?

To **everybody** — ... I mean:

- Introductory and conclusive matter should be written in a style easy to understand by non-specialists.
- Core chapters will inevitably be technical, so they are bound to be written for the **specialist**.

Final check up — the question is

> *Do I master my domain of knowledge upon completion of my project?*

Well...

- you should be able to **explain** what you did to **anyone** you may meet in the street. (abstraction!)

# How should I write it?

Two sides of the question:

- **Style** (text quality, etc)
- **Production** (editing and publishing)

Style:

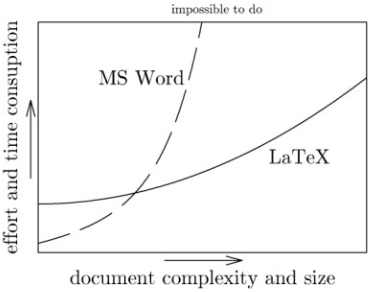
- Avoid colloquialisms and any form of *majestic* style ( *"we"*, *"our"*, ...) — be **modest**.
- Avoid past tenses (scientific writing is **not** story telling).
- Text "comes in pairs":
  - Backward integrity — **declaration** always before **use** (eg. definition before application).
  - Forward integrity — make sure you **fulfill** whatever you **promise**.

  Cf. offer / demand , client / server, etc

# How should I write it

Production — use a proper **text authoring system**. By **proper** I mean one that:

- Handles **references** and maintains **referential integrity**.
- Automates **routine tasks** such as numbering, bibliography, generation of lists and indices.
- Integrates well with **other tools**.




One such system is the Knuth-Lamport's LaTeX's text preparation system (Goossens et al., 1997).

(Maybe you know of others).

# How do I write it?

Handling references:

- Concepts, entities etc have a **name** (reference) and often a type.
- Textual information (implicitly) contains a set of **name spaces**.
- A name in each name space identifies a unique object — it is a **reference**.
- Name spaces call for **referential integrity**.
- Most of these are ensured by the text authoring system itself — eg. names (numbers) of **figures**, **tables**, **sections**, **theorems**, etc.
- One should be very careful about handling any other references (names).

# How do I write it?

For those not handled, here is how I like dealing with them (for LaTeX users only — sorry!): for each **entity**, eg.

- **Entity**: University of Minho
- **Acronym**: UM

define (under package `hyperref`) its (unique) reference **name**:

```
\newcommand{\uminho}[1]{
     \href{http://www.uminho.pt}{#1}
     \index{UM!University of Minho}}
```

Mind that, every time you write eg. `\uminho{the university}`,

- you provide a link to the website you've chosen for the mentioned entity;
- an entry is added to the **index of terms**, that is, the occurrence of term `uminho` in the **current page** is recorded.

# How do I write it?

Then an acronym (short-cut) can be defined:

```
\newcommand{\UM}{\uminho{\textsc{u.m.}}}
```

So, every time you use acronym `\UM`, LaTeX typesets U.M. and does the same as above concerning hyperlinking and index-management.

This saves you from referring to relevant entities which are not in the list of terms.

Last but not least:

- Keep your dissertation in a document **version-control system** like eg. SVN or DARCS — among many other alternatives, often web-based.

# Interfacing with others' work

Last but not least, we need to be concerned with **bibliography** management:

- Nobody doing relevant research is alone.
- Research is actually a **social** activity, with continued interaction in the form of meetings, conferences, and so on.
- Giving **credit** to the others' **contributions** is the main rule of the game.
- With the information resources of today, managing this may be hard (too much data!) without a proper infra-structure.
- This should take the form of a **bibliography database**.

# Interfacing with others' work

Systems around BibTEX provide for very easy management of bibliography data:

- A BibTEX record is like a database record, eg:

```
@book{GRM97
    , title     = {The LaTeX Graphics Companion}
    , author    = {Michel Goossens and
                   Sebastian Rahtz and Frank Mittelbach}
    , publisher = {Addison-Wesley}
    , year      = {1997}
    , note      = {ISBN 0-201-85469-4}
}
```

- You may add your **own attributes** (which don't get printed) like IDs of books in your own library, bibliometric stuff, and so son.

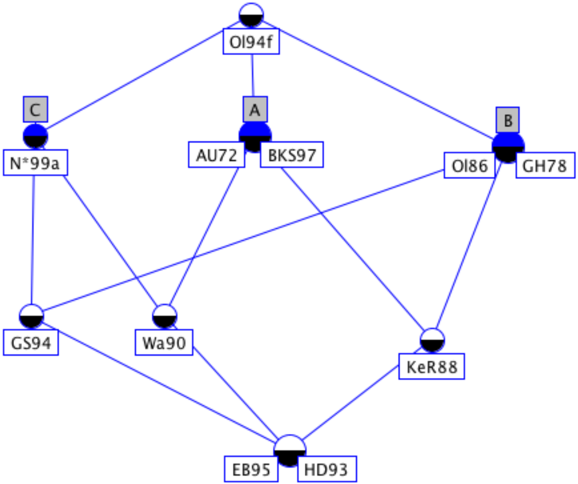
# Interfacing with others' work

Classifying your bibliography:

- In particular, you may add a BibTEX attribute named `keywords` to each record of interest.
- This will **classify** your records according to keywords relevant to your research.
- You many even use the technique of **formal concept analysis** (FCA) developed by Ganter and Wille (1999) to structure your bibliography in a **lattice of concepts**.
- Some FCA systems (such as CONEXP) offer you a user interface to manage and display your concept lattice (next slide).

# Interfacing with others' work

Example concept lattice (11 records, three attributes $A$, $B$ abd $C$):



# Interfacing with others' work

The classification which generates such a lattice is as follows:

| BibTEX key | A | B | C |
|---|---|---|---|
| Ol94f | 0 | 0 | 0 |
| AU72 | 1 | 0 | 0 |
| Ol86 | 0 | 1 | 0 |
| N*99a | 0 | 0 | 1 |
| KeR88 | 1 | 1 | 0 |
| GS94 | 0 | 1 | 1 |
| Wa90 | 1 | 0 | 1 |
| EB95 | 1 | 1 | 1 |
| BKS97 | 1 | 0 | 0 |
| GH78 | 0 | 1 | 0 |
| HD93 | 1 | 1 | 1 |

Such concepts should help in organizing your review of the state of the art.

# Interfacing with others' work

Careful review of the **state of the art** in the area you intend to work on is valuable in itself and can be published.

An example of this is reference (Couto et al., 2011) — a paper which emerged from the UCE15 report by Luís Couto (pg15260) on reviewing literature on software architecture quality, last year.

Using FCA, Luís Couto's enunciates a number of **research questions** which he tries to answer by generating FCA lattices for each of them.

(Worth having a look.)

# Some links

- *BibSonomy* (a system for sharing bookmarks and lists of literature) — www.bibsonomy.org
- *DBLP Computer Science Bibliography* (comprehensive account of BibTEXrecords) — www.informatik.uni-trier.de/˜ ley/db/index.html
- *Writing and Presenting Your Thesis or Dissertation* — www.learnerassociates.net/dissthes/
- *How to Write a PhD Thesis* — www.phys.unsw.edu.au/˜ jw/thesis.html
- *Small guide to making nice tables* — www.inf.ethz.ch/personal/markusp/teaching/guides/guide-tables.pdf

among many others Google will offer to you.

# Closing

Final suggestions:

- **Interact** with other researchers in your field.
- Once you have something to show, build a **research blog**.
- Try and **publish** your work in good conferences — the best way to validate your contributions.
- Good **papers** convert to good chapters in the dissertation.
- Offer your **services** in OC/PCs of **conferences** in your area.

and don't forget

- to be **creative** (recall K. Popper)
- to have **fun**: if you don't get excited with your project — who will?

# References

L. Couto, J.N. Oliveira, M.A. Ferreira, and E. Bouwers. Preparing for a literature survey of software architecture using formal concept analysis, 2011. Proc. of the SQM'2011 workshop, colocated with CSMR 2011, Oldenburg, Germany.

Bernhard Ganter and Rudolph Wille. *Formal concept analysis: Mathematical foundations*. Springer, Berlin-Heidelberg, 1999.

Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion*. Addison-Wesley, 1997. ISBN 0-201-85469-4.

L. Russo. *The Forgotten Revolution: How Science Was Born in 300BC and Why It Had to Be Reborn*. Springer-Verlag, September 2003. URL http://www.springer.com/978-3-540-20396-4.