Preparing LaTeX classes for journal articles and university theses

Tom Hejda

Abstract

There is a substantial difference between the requirements on a LaTeX class for a scientific journal and for university theses. The main point is that a journal class is by definition *restrictive* — the journal has to be very keen on the precise look and structure of the articles, whereas the thesis class is by definition *modular* — different theses ask for a slightly different layout and structure, some have appendices and some do not, etc. We discuss the differences and their implications on the class design.

1 Introduction

It is natural that different types of documents ask for different LaTeX classes. We will discuss the differences for journal articles and university theses. This is partly a response to a recent boom in LaTeX classes for theses issued and enforced by universities, where it is commonly seen that the classes do not meet good standards, students have difficulties using them and the result is in many cases far from satisfactory. Even though this is the case, we refrain from giving bad examples, and we rather focus on the core ideas that should be behind the design of such a class.

This paper is organized very simply. In the next two sections, we discuss the demands on classes with different purposes. In Section 4, we describe the solution to the demands that were used to design and code the `ctuthesis` class that is being developed at the Czech Technical University (CTU) in Prague. We believe that our proposed solution serves as a good example of how things *can* be done.

It should be noted that while graphical design plays an important role in the publication process, we will omit the discussion about graphics as this is mostly irrelevant to our points. We merely note that the class `ctuthesis` that will be used as an example is based on a plain TeX class called `ctustyle` [3], to which the next article in this *TUGboat* issue is devoted.

2 Different documents are made differently

The typical workflow for publishing articles in scientific journals involves several steps:

1. **Primary submission by the authors** — it need not be in the journal's style and need not strictly follow the typographical policies of the journal.
2. **A referee process leading to an accepted version** (or rejection, but that is not interesting for us) — at the end of this process, the authors provide a version of the article that should comply with all the in-house policies.
3. **In-house typesetting and editorial copy preparation** — The staff of the publisher take the sources (code, figures, etc.) and prepare the article to their liking.
4. **Proofreading** — the authors point out any mistakes made during the typesetting and possibly other things they do not like.

If we look into how theses are usually typeset, we see that most often the last two steps are missing: There is no one to typeset the thesis in a professional way nor to control the way that the thesis is typeset. This means that the thesis author is in some sense much more responsible for his work than the author of an article, at least from the typographical point of view.

3 Variety of documents

A second big difference between the two class types is in the variety of documents. In general, most articles in a single journal follow a similar scheme for sectioning, floating objects, references etc.; also, they are usually from a rather narrow field.

On the other hand, a single university has many faculties with many branches of study, and it is clear that a programming thesis looks significantly different from a theological text or an architectural study. It is quite natural that the first one will contain a lot of code samples and probably a reference manual, the second one will be basically a long text with a lot of direct quotes of paragraphs from other sources, and the third will contain a long graphical appendix. Also, the thesis is the student's child and he should be able to make it look as he likes, within the requirements.

To allow a single LaTeX class to accommodate all these needs, the class has to be highly modular; the presence of appendices has to be optional, for instance, and in general, more or less everything has to be configurable. The class should have only minimal fixed design in order to comply with the requirements of the university.

4 Our solution

The solution for article classes used by `actapoly` (the journal of the CTU) [1] does not involve any special tools — article authors set up the metadata of the article and these are then used by `\maketitle` to print the article title block. All the standard LaTeX

environments and commands such as sectioning commands, lists, floats, tables etc. are then given a fixed graphical design that forms the graphical identity of the journal. This is what nearly every journal publisher does in their class files.

By comparison, in designing our class — called `ctuthesis` [2] — for university theses, we needed a high level of modularity, as discussed. This is allowed mostly by two important ingredients:

1. **Good key-value interface.** Most modifiers of the class behaviour are implemented using this interface. The interface itself is coded using the very usable and highly versatile `l3keys` package. In general, the whole class is written in `expl3` as much as possible.
2. **Two-phase class and package loading.** The idea can be seen in Figure 1 — we load the class, then set everything up using the key-value interface, and then the command `\ctuprocess` inputs another file of the class. This additional file contains a lot of conditional package loading and package setup.

There are several types of keys for the key-value interface:

- appearance keys — languages, colours, a switch for the inclusion of the list of figures, etc.;
- metadata keys — title, subtitle, author, supervisor, name of the department, and a lot of other information;
- package options — customizable loading of certain packages for which it makes sense, including for instance: `amsthm` (since someone may prefer `ntheorem` or another package and there is no reason to forbid it), `listings` (we set up the listings design in a particular way that someone may not like), or `hyperref` (since it is sensitive to the order in which the packages are loaded and making it conditional can help in resolving the issue).

Also, in the internal design, we borrow the idea that is seen in `beamer` — namely what we call *templates* and *fields*. Examples are worth complicated explanations, so as an example, the titlepage, or the list of figures in the two-column frontmatter make up typical templates, whereas fields are things such as the title and the abstract (these are actually language-dependent, so we have a field for the title in all languages in which it is needed, and similarly for the abstract, the university name, etc.), the name of the author, the address of the supervisor, etc.

Also, there is an interface for *themes* — it could happen that a faculty of the university had a special requirement that "supervisor" should be called "project manager", and this is possible using a theme for this faculty that changes `\supervisorname` in the `english` language. It is of course possible to implement this without the themes interface, but it would mean adding strange conditionals at strange places in the class files for one-off issues like this one. We do, however, store all templates and themes in a single file with a clear structure.

```
\documentclass{ctuthesis}
\ctusetup{
   key1 = value1,
   key2 = value2,
   ...
}
\ctuprocess

% ... user stuff goes here ...

\begin{document}
\maketitle
...
```

Figure 1: Structure of the preamble of a document in `ctuthesis`.

5 Concluding remarks

To conclude, let us mention the most important points of the paper:

1. Different document types need different class designs.
2. Class authors should think of how the class will be used and who the users will be.
3. The more the users will interact with the class, the cleaner the class interface should be.

References

[1] Czech Technical University in Prague. Acta Polytechnica — submissions. `https://ojs.cvut.cz/ojs/index.php/ap/about/submissions#authorGuidelines` [2015-08-01].

[2] Tom Hejda. LaTeX template for theses at CTU in Prague. `https://github.com/tohecz/ctuthesis` [2015-08-01].

[3] Petr Olšák. CTUstyle — Plain TeX template for theses at CTU in Prague. `http://petr.olsak.net/ctustyle-e.html` [2015-08-01].

◇ Tom Hejda
Dept. Math. FNSPE, Czech Technical University in Prague
Trojanova 13
Prague 12000
Czechia
`tohecz (at) gmail dot com`
`http://github.com/tohecz/`