A patent application design flow in LaTeX and LyX

Peter J. Pupalaikis

Abstract

I describe a design flow for beautiful and consistently formatted U.S. patent applications using our favorite typesetting tools LaTeX and LyX along with a newly-created LaTeX class file and an accompanying LyX layout file.

1 Introduction

The patent process is often thought of as a thing of mystery cloaked in arcane terminology and rules. It is also thought to be very expensive. This is surprising because these thoughts are often held by inventors who are performing in highly technical fields where research itself is expensive. It turns out that the patent process is actually fairly straightforward and can be relatively inexpensive when the inventor takes an active role. It is helpful to stay grounded in some simple concepts:

- The patent process is one that preferably ends with a document describing certain intellectual property rights.
- The property rights obtained are the rights to exclude others from practicing the *claims*.
- In exchange for these rights, an inventor is obligated to disclose the best way he knows how to practice an invention.
- The document containing the disclosure is the patent application.
- In order to obtain patent protection, an invention must be novel, unique, and useful.

Some information that many are surprised to know:

- Anyone may file a patent application.
- With sufficient effort and narrowing of claims scope, a patent can almost always be obtained.
- The actual protection granted by a patent is discovered during litigation.

I have made some rather broad statements here. While anyone can file an application, it's advisable to get help from an attorney. The process of narrowing the scope of claims during prosecution can provide for a patent with very limited usefulness to the patent owner. Finally, a poorly written application can lead to poor protection and as I said, the quality of the protection is found out too late.

I have been inventing and patenting for about twenty years now. When I started out, I knew next to nothing about the patent process. The process (at medium to large companies) from an inventor's perspective is to fill out a *patent disclosure form*. This is a form that describes the invention and is used to supply necessary information to a patent attorney. These forms are very common at big companies. Then, a few months after supplying this information and a few discussions with an attorney, a patent application arrives. That's the theory anyway. I found my early patenting experiences very difficult and unsatisfying. It was a painful process of transmission of highly technical information from inventor to an (albeit technically trained) attorney and legal information from attorney to inventor. Over time, I found that the choices with this process are between expensive, time consuming and low (or at least unclear) quality on the one hand and very expensive, very time consuming with reasonable quality on the other.

My goal, therefore, was to lower expense and effort and improve quality. This led to the creation of patent application drafts and culminated in the creation of tools for improving this flow. Today, I write all of my patent applications myself in a form ready for filing. My application is not filed until a patent attorney has cleaned it up and rewritten the claims after many discussions. I've discovered that getting a higher quality patent disclosure into the hands of a patent attorney is the right way to ensure that money spent on legal fees is spent wisely. Working with me, the attorney concentrates on using her time and legal knowledge to get me the right protection for the right price.

No matter your opinions, training or systems for drafting patent applications, it is clear that the process benefits extremely from improved tools for producing the application itself. Improved tools can solve many of the mechanical problems with patent drafting and can ensure consistently generated applications. This article is about how LaTeX and its more user-friendly cousin LyX can be used to streamline the patent application process.

2 Patent writing tools

After my first frustrating experiences of rewriting patent applications, my first attempts to improve the process began with efforts at better patent disclosures using well-known word processing tools from well-known software companies. Using these tools, I would provide a disclosure filled out under the same headings as the patent application. Over time, I learned the intent of each section of the application and after each patent was filed, I compared what I provided to what ended up in the attorney-generated application and honed my skills; this culminated in the Knuthian leap of controlling the final output and properly formatting and typesetting the final

application. This is where I ran into difficult tool problems. Let me outline some of them:

1. The major problem — drawings! The patent office has rather arcane rules about drawings and specific standards for drawings. One requirement is that drawings contain elements that are *annotated* (i.e. a numbered arrow or line on the drawing pointing to an element) where the drawing-element name and accompanying number are used within the patent application to refer to the element. Despite looking bad, this is a cross-referencing nightmare as no tools (other than LaTeX, eventually) could be made to use variable names that get replaced with numbers in both the drawings and the patent specification.
2. Drawing and drawing-element name and number agreement. The problem is that the drawing-element name and the number must be consistent. If I refer to widget [23] in my specification, there had better be a drawing with a line pointing to the widget element and it had better be numbered with 23. Keeping the name of the element consistent and matching the number was a problem. Also, it would be nice if the next time I refer to element 23, I call it the same thing.
3. Figure numbers. There is one section in an application where the drawings are to be described *in order*. Most cross-referencing tools get the number right, but cannot place the drawing description in the correct order and cannot make the number agree with the description.

Other problems include:

1. The formatting capabilities of various word processors, believe it or not, are too flexible. One can edit the application, change the fonts, styles, headers, whatever one wants. They're flexible while missing key features.
2. Have you ever tried to get drawings in the right place using the world's most popular word processor? Enough said.
3. Consistency of claim language and support in the specification.

I was finding that in some cases it was nearly impossible to control the format of the final application the way I wanted. When I saw the paralegal at my company cut out a drawing I made and tape it onto a sheet of paper to get the drawings right, I determined that enough was enough. I set out to solve these cross-referencing and formatting problems.

All of the above problems benefit from a programmatic approach; an approach with variables and macros that can be used for cross-referencing, consistency of language, and consistency of format.

```
\documentclass[english]{uspatent}
...all front-matter definitions ...
\include{Drawings}% figure information
\maketitle
\patentSection{Field of the Invention}
\patentParagraph text ...
\patentSection{Cross Reference to Related
   Applications}
\patentParagraph text ...
\patentSection{Background of the Invention}
\patentParagraph text ...
\patentSection{Objects of the Invention}
\patentParagraph text ...
\patentSection{Summary of the Invention}
\patentParagraph text ...
\patentDrawingDescriptions
\patentSection{Detailed Description of the
   Preferred Embodiments}
\patentParagraph text ...
\patentClaimsStart
...
\patentClaimsEnd
\patentSection{Abstract}
\patentParagraph text ...
\patentDrawings
\end{document}
```

Figure 1: LaTeX Patent Application Structure

3 Patent writing in LaTeX

Anyone reading this *TUGboat* article is likely to be acquainted with LaTeX, so I'll dive right in. The solutions to some of the previously mentioned problems are solved through the incorporation of various macros and page settings into a LaTeX class file `uspatent.cls`. The class finally developed and published is based on the `memoir` class. The declaration of this class appears at the top of the LaTeX patent application file as shown in figure 1. Here you can see the layout of the document, which includes:

- various front-matter material. These are macros which define the title, inventor name, assignee information, patent attorney information, etc. These assignments are used in header, footer and title page creation.
- inclusion of the drawings file (which I will talk about in a bit).
- various sections of the patent which will be properly formatted via the `\patentSection` macro and where each paragraph is preceded by the `\patentParagraph` macro that causes the paragraphs to be numbered.

```
\figureDefinition{VisioDrawing}
\figureExtension{pdf}
\figureDescription{example drawing made in Visio}
\annotationDefinition{Widget}
\annotationName{widget}
\annotationDescription{a widget in the drawing}

\figureDefinition{TpXDrawing}
\figureExtension{tpx}
\figureCaption{PRIOR ART}
\figureDescription{example drawing made in TpX}
\annotationDefinition{input}
\annotationName{input}
\annotationDescription{the input}
\annotationDefinition{output}
\annotationName{output}
\annotationDescription{the output}
\annotationDefinition{mathProcessor}
\annotationName{math processor}
\annotationDescription{math processor on left}
```

Figure 2: Drawing definitions in LaTeX

- a patent claims area (which is actually just an enumerated environment).
- the `\patentDrawingDescriptions` macro that automatically emits the description of the drawings.
- the `\patentDrawings` macro that automatically emits the drawing pages.

The drawings file contains a list of drawings and annotations, as shown in figure 2. Here you see a list of macros that will define everything about the figures and annotations. Each figure's information is listed, in order starting with a `\figureDefinition` macro that defines the name of the file and how it is referenced from within the application. It is assigned a number from an internal counter. This is followed by a `\figureExtension` macro that defines its extension and the `\figureDescription` macro which provides a concise description. Usually patent drawings do not contain captions, but one can be optionally provided using the `\figureCaption` macro. You might be wondering why I didn't use a multiple argument macro. I found it difficult to remember the order of the arguments but more importantly, macros which have multiple arguments cannot be used within LyX.

After each figure definition its annotations are listed, each one beginning with a call to the macro `\annotationDefinition`. This specifies the name used to reference the annotation within the document. The macro also assigns a unique annotation number from an internal counter. Each annotation definition is followed by calls to `\annotationName` and `\annotationDescription`. The name is the word associated with the element that you want printed in your document when you refer to it. The description is a longer description that helps find the annotation in the drawing and is used with drawing packages that cannot make use of LaTeX cross-referencing. I will explain its use further when I discuss the Annotation List section.

The figure and annotation definitions can appear inline if you want, but it gets very long and bothersome to see all these definitions when you are editing the application so it's better to include them in a separate file. After the drawings file is processed, it enables the following features:

- the figures and annotations can be referred to using several macros:
  - `\referencePatentFigure` expands to the formatted patent figure (e.g. FIG 2.).
  - `\annotateWithName` is the normal way to reference the annotations in the application and maintain agreement between element name and number. It expands to the annotation name and formatted annotation number joined (e.g. widget [3]).
  - `\annotate` expands to the formatted annotation number (e.g. [3]). It is used when the name and number would appear awkward (e.g. ... is connected to the two widgets [3] and [4]).
  - `\annotationNumberReference` is used in drawings to point to the drawing elements. LaTeX friendly drawing formats (like Ti*k*Z) can be made to draw the correct number pointing to a drawing element.
- The `\patentDrawingDescriptions` macro expands into a section that automatically prints a section that looks like:

Brief Description of the Drawings

For a more complete understanding of the invention, reference is made to the following description and accompanying drawings, in which:
FIG. 1 is aaaa;
...
FIG. x is bbbb; and
FIG. y is cccc.

Here the figure's descriptions are the exact text in the descriptions provided, match the figure numbers in order, and are even punctuated properly according to how the patent office would like to see them.

- The `\patentDrawings` macro expands into a section that brings in all of the drawings and places them appropriately on numbered pages. If the `\annotationNumberReference` macro is used and the drawings have been produced with a LaTeX friendly tool (like TpX) and/or in a

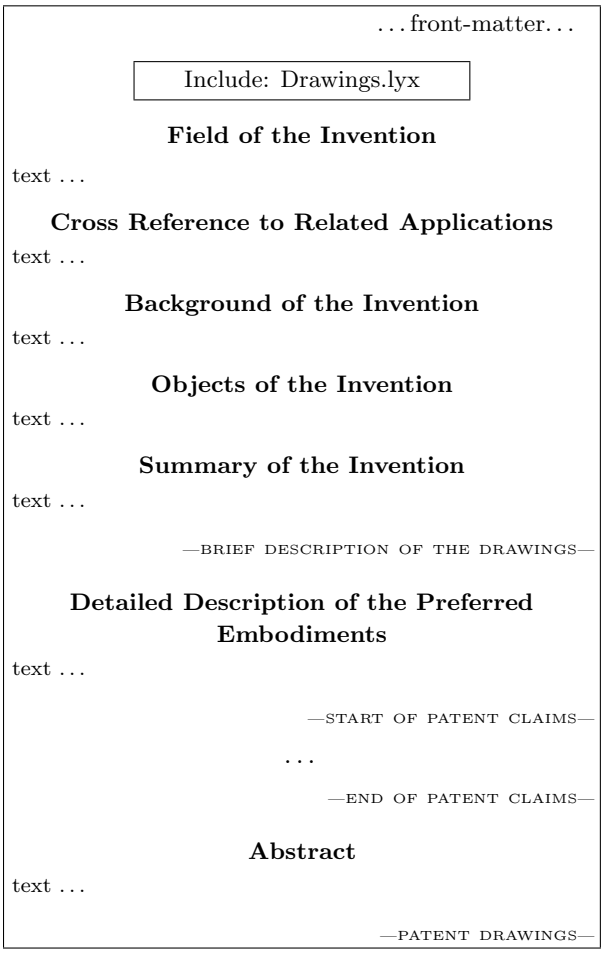

Figure 3: LyX Patent Application Structure

LaTeX friendly format (like Ti*k*Z), then the annotations are also numbered properly. I also include a printing mode switch via the macro `\setPrintingModeDraft`. Among other things, this macro causes an Annotation List section to precede the drawing pages. This section is not meant for filing; it lists all of the figures, their names, annotations, and their names and descriptions. This is so that they can be matched up with the drawings to ensure that the right numbers are in the right place. I also provided this for users who use drawing tools that are not LaTeX friendly and do not allow the referencing of the annotation with the `\annotationNumberReference` macro. These users must manually place the numbers on the drawing using the information in the annotation list after finishing writing the application.

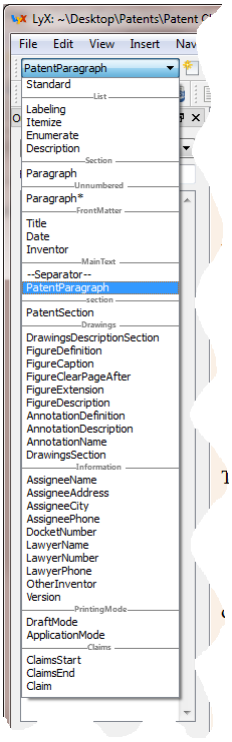

Figure 4: LyX Custom Environments

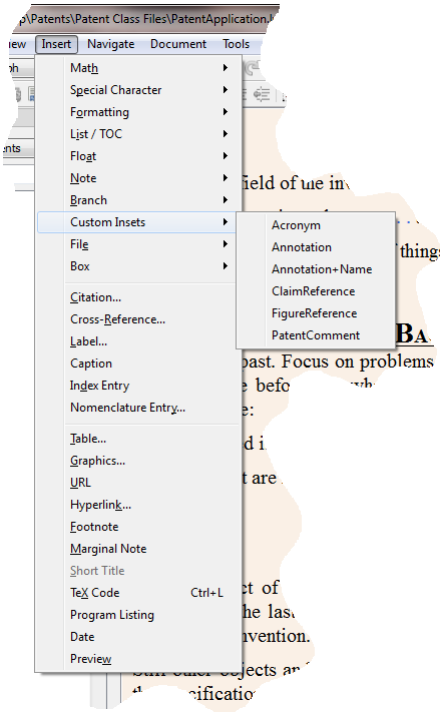

Figure 5: LyX Custom Insets

```
\def\annotationDefinition#1{%
 \expandafter\ifx\csname anonum#1 \endcsname\relax
 \global\advance\@annotationnumber by 1
 \expandafter\edef\csname anoele \the\@annotationnumber\endcsname{#1}%
 \expandafter\edef\csname anonum#1 \endcsname{\the\@annotationnumber}%
 \expandafter\edef\csname anofignum \the\@annotationnumber\endcsname{\the\@annotationfigurenumber}%
 \else % error handling here
 \fi}
\def\annotationDescription#1{\expandafter\def\csname anodesc \the\@annotationnumber\endcsname{#1}}
\def\annotationName#1{\expandafter\def\csname anotext \the\@annotationnumber\endcsname{#1}}
\def\annotationReference#1{[\thinspace\annotationNumberReference{#1}\thinspace]}
\def\annotationNameAndReference#1{\annotationTextReference{#1}~\annotationReference{#1}}
\def\annotationDescriptionReference#1{\csname anodesc \annotationNumberReference{#1}\endcsname}
\def\annotationTextReference#1{\csname anotext \annotationNumberReference{#1}\endcsname}
\def\annotationNumberReference#1{\csname anonum#1 \endcsname}
\def\annotationListVariableName#1{\csname anoele #1\endcsname}
\def\annotationListText#1{\csname anotext #1\endcsname}
\def\annotationListDescription#1{\csname anodesc #1\endcsname}
\def\annotationListFigureNumber#1{\csname anofignum #1\endcsname}
```

Figure 6: Annotation Macros in LaTeX

4 Patent writing in LyX

LyX has been written about in many *TUGboat* articles, but briefly, it is a front-end for LaTeX. LyX is a WYSIWYM editor, i.e. "What You See Is What You Mean". LyX takes the guesswork out of what equations will look like in the end, removes some of the verbosity of LaTeX from the user, and can be used to limit and target the formatting capability that the user has (he can always enter raw LaTeX in the end, so the full capability of LaTeX is never completely out of reach).

With a properly written layout file, LyX attempts to mimic to some degree what the document will mostly look like. With the `uspatent.layout` file that goes along with the `uspatent.cls` LaTeX class file, it provides the patent writing capability through the use of *environments* and *custom insets*. A typical LyX patent application looks much like the analogous LaTeX application, only the formatting mimics that of the final PDF and the user can choose not to see any LaTeX code, as shown in figure 3. After the LaTeX explanation, I think this figure does not need much explanation.

The environments are accessed in LyX using a drop-down box in the upper left corner of the editor. The customized environment list for patent applications is shown in figure 4. Here we see all of the environments for the front-matter, patent sections, patent paragraphs, claims and the drawings and annotations. Mostly, the use of these environments saves on typing the LaTeX macros and shows some sort of indication on the screen that the macro arguments are in that environment. Even more important, it restricts the user's choice of formatting.

The custom insets are how all of the references are made to claims, figures and annotations as previously described. This is shown in figure 5. These put small boxes inline with the text that can be expanded to show the arguments. Here you see the addition of an Acronym custom inset (which uses the LaTeX `acro` package) which is also useful in patent applications.

5 A few TeX tricks

The main tricks used in dealing with the figures and annotations are based on the use of `\csname`.[1] I want to explain the mechanics here because they might be useful for solving other similar problems. Please refer to the TeX listing in figure 6.

First, a brief summary of these macros:

- The first three macros are *Assignment* macros: `\annotationDefinition`, `\annotationDescription`, and `\annotationName` refer to the definition of an annotation.
- The next five macros are *Referencing* macros and are suffixed by "Reference": `\annotationReference`, `\annotationNameAndReference`, `\annotationDescriptionReference`, `\annotationTextReference`, and `\annotationNumberReference` provide reference of an annotation by a *VarName*.
- The last four macros are *List* macros and are prefixed by "annotationList":

[1] Amy Hendrickson, "The wonders of `\csname`", TUG 2012, Boston MA; *TUGboat* 33:2, pp. 219–224, `http://tug.org/TUGboat/33-2/tb104hendrickson.pdf`.

`\annotationListVariableName`, `\annotationListText`, `\annotationListDescription`, and `\annotationListFigureNumber` provide reference to an annotation by *Number*. The reason they are called *List* macros is because they are most useful for generating an annotation list.

The annotation is defined by first using the macro `\annotationDefinition`, where the argument supplied becomes the *VarName* for the annotation. In figure 6 you can see that an annotation counter is advanced and two new control sequences are defined. One control sequence is *anoele* followed by the annotation counter value. It is assigned to *VarName*. The other control sequence is *anonum* followed by *VarName*. It is assigned to the annotation counter value. These two assignments allow one to obtain the *VarName* given an annotation number and to obtain the annotation number given the *VarName*. The key *List* and *Referencing* macros are such that:

`\annotationListVariableName{`*Number*`}` = *VarName*
`\annotationNumberReference{`*VarName*`}` = *Number*

The macro `\annotationDescription` defines a control sequence named `anodesc` followed by the annotation counter value. Similarly, `\annotationName` defines a control sequence `anotext` followed by the annotation counter value. In this way, given an annotation number, it is easy to find the *VarName*, and the description and text name associated with an annotation through the use of the *List* macros provided. When we generate the annotation list in the draft mode patent application, we simply loop over the annotation numbers and list all of this information.

The *Referencing* macros are used within the patent application and within drawings. They are used to reference the annotation number, a formatted number (e.g. [*Number*]) and the annotation text that goes with that number. The annotation text is an interesting example because the `\annotationName` associated the text with the annotation number and we are referencing this text with the *VarName*. If you examine the `\annotationTextReference` macro, you see that the `\annotationNumberReference` macro is used to obtain the *Number* associated with the *VarName* supplied and this *Number* is used in conjunction with the associated *anotext* to form the control sequence that defines the text. In this manner, any of the annotation control sequences that define the annotation can be formed from either the annotation *Number* or *VarName*.

6 Summary and future plans

I've presented a method, using the `uspatent.cls` file in LaTeX and the `uspatent.layout` file in LyX, for typesetting patent applications that produce consistent and good looking results. Look for these files along with a patent writing guide on CTAN at `http://ctan.org/pkg/uspatent`. There are a number of things I'd like to improve in future versions. Some of these are:

- Usage of the `aux` file so that the figure and annotation definitions can go anywhere in the document (currently they need to go at the top before they are referenced).
- Elimination of dependence on the `memoir` class. The `memoir` class is great but heavy-handed for this application. I make use of very little of this class's capability and know that I can remove it and make use of a few smaller packages.
- I'd like to become more adept at my usage of `\csname`, which I still haven't quite mastered.
- I'd like to add better error handling and undefined reference handling capability so that users don't get a cryptic (LA)TeX error message when things fail. Currently, if an undefined annotation is referred to, it simply doesn't print; this is obviously a poor way to handle this type of error.
- I was too lazy to figure out how to parse the figure names to extract the extension.
- I'd like to eventually develop an offline piece of software for managing drawings and annotations. While the tools provided here fix some of the most difficult problems and are a step in the right direction, I know of many more ways to improve the patent writing process. A small step in this direction would be a drawing package that can import a PDF and, through a graphical interface, add Ti*k*Z code that draws a line pointing to drawing elements and the macro that expands to the element number.

This has been a fun and fruitful experience and is my first contribution to CTAN and the LaTeX community. I hope to make many more in the future!

Happy TeX patenting!

◇ Peter J. Pupalaikis
Ramsey, NJ USA
`pete_pope (at) hotmail dot com`
`http://mysite.verizon.net/petepope/id6.html`