Rationale for Platypus

The fundamental question of any rationale is why enter into this project at all? In the case of Platypus, this question has more merit due to the existence of one well-known package in this area: TeX.

Limitations of TeX that inspired Platypus

TeX is a great program and, due to its extensive use in academia, thoroughly debugged. It is one of the first, truly great open-source projects that predates many, if not most, of the cornerstone open-source projects of today. However, even diehard TeX users, acknowledge that despite its success, TeX has limitations, many of which Platypus seeks to address:

- Difficult to learn and use. There is considerable arcana that must be learned to understand how the pieces come together. Anything beyond a simple document, requires study and frequently debugging sessions.
- Very difficult to debug: When the document doesn’t turn out as you expected, it’s extremely difficult to figure out what went wrong. Except for trivial cases, you’re forced to consult other TeX users for advice.
- Irregular language: there is no grammar for TeX. Its designer, Donald Knuth, specifically designed it this way. The problem is not so much that its difficult to parse, but because there are so many features with their own rules, TeX requires memorization. For example, foreign accents are rendered one way in text. But in a table, they are rendered using an entirely different coding sequence. And the original sequence generates invalid results.
- Only one implemented standard: Knuth imposed the requirement that no other program except the TeX he wrote can call itself TeX. Hence, the only way to validate a document is by running it through TeX and looking at the results. Because of this aspect and point number 2, there are no tools that guarantee validation of TeX documents other than by running them through the one program.
- Difficult macro language: TeX has a notoriously cumbersome macro language. And the only way to add features to TeX is through macros. Due to the difficulty of the language, few people write these macros and, instead, rely on a few macro packages, such as LaTeX. The trouble with LaTeX is that documents generated with it look a lot alike, because most users simply use the default templates. Again, the difficulty of creating new templates and macros constrains imaginative use of TeX. Moreover, the macro language does not allow you to program a document.
- TeX has clumsy support for international character sets.
- Output is to dvi files, which then have to be converted by other processes to various output formats, rather than generating those formats directly.
- Lack support for numbers and counters larger than 32-bits.
- Default use of bit-mapped fonts.

For more information on some of these points by a TeX expert and aficionado, see http://www.math.utah.edu/~beebe/talks/2003/tug2003/beebe.pdf

What about Lout?

Lout is an alternative to TeX that is also open-source. Lout is supported by a small community, driven by one person at the core, who works on other projects. As a result, when Lout breaks, remediation has historically been a long time coming. At the time of this writing, the principal author of Lout is working on a new language to describe document formatting.

- Lout has an unusual programming model that must be understood to use the language properly. The documentation for Lout is comparatively sparse and so anything beyond basic documents requires experimentation.
- Lout lacks a macro language
- Generates only PostScript files

So, What does Platypus Offer?

The goals of Platypus are:

- Simplicity! You should be able to produce documents with a minimum of effort. Even hard things should be expressed simply and default to sane values and actions;
- A programmable macro language for dynamic documents;
- Full support for international character sets
- Output to multiple media formats. First PDF; eventually HTML and RTF as well.
- Use of standard fonts (TrueType, OpenType, etc.)
- Programmer-specific needs handled (Many ways to print code easily, etc.)
- Written in Java for universal portability.

To reach these goals in a reasonable time frame, the Platypus project is borrowing some components from other projects:

- Scripting language will probably be ECMAscript (JavaScript);
- The primary rendering engine for the PDF, HTML, and RTF formats will be the open-source iText library.
- Numerous algorithms will be derived from Knuth's superb books on TeX.