Developing and Releasing the Ω System
A White Paper, version 0.3

John Plaice* Yannis Haralambous†

November 16, 2002

This paper is a living document. For the moment, it should be considered a mini-white-paper, sufficient for initiating the cleanup of the current distributions. A much more complete document should be forthcoming in time.

1 Introduction

The Ω Typesetting and Document Processing System is a software suite whose aim is the high-quality production of multilingual documents in electronic or printed form. Originally designed as a series of extensions to the TEX typesetting system created by Donald Knuth, its goals have become more ambitious.

Given that the goals of the Ω system have become open-ended, it is important to make sure that stable intermediate releases, corresponding to clear goals, are undertaken. It is the purpose of this paper to define the current plans for Ω.

Our broad goal is that a major, stable, documented, widely distributed Ω release be made in Spring 2003, so that it can be presented at the *EuroTEX* conference in Brest, France (24–27 June 2003) and the *TEX Users Group* conference in Hawaii, USA (20–24 July 2003). We expect a number of minor releases to be undertaken along the way.

In our understanding, there is only room for one successor to TEX, and that this should be Ω. As a result, Ω must be capable of producing high-quality typeset output, for a variety of formats. It must also be usable to produce, from the TEX syntax, structural markup as well as layout markup. In time, it will be able to read directly from structural markup without passing through the TEX syntax. Therefore, Ω must be able to incorporate work from a number of existing projects, such as pTEX, ε-TEX, pdfTEX, LATEX2HTML and TEX4HT, and be developed with the advice and support of the creators of large macro packages such as LATEX and ConTEXt.

This paper focuses not just on the programming that must be undertaken, but also on all of the organization (Web pages, mailing lists, translation, bug-tracking, etc.) that must take place.

2 Setting Up the Infrastructure

The Ω project has for a number of years been run quite loosely. The aim here is to transform it into a tightly-managed free software project, which clearly distinguishes between research and development issues.

*School of Computer Science and Engineering, The University of New South Wales, UNSW SYDNEY NSW 2052, Australia. plaice@cse.unsw.edu.au

†Département Informatique, École Nationale Supérieure des Télécommunications de Bretagne, Technopôle Brest-Iroise, BP832, F-29285 Brest Cedex, France. Yannis.Haralambous@enst-bretagne.fr

2.1 Web page URL

The Ω Web page resides at `http://omega.cse.unsw.edu.au` , which can also be reached through `http://omega.enstb.org` . Previously there was an Ω Web site at `http://www.ens.fr/omega` . as well as at `http://omega-system.sourceforge.net` .

Task 2.1.1. There are numerous links to the Ω Project around the world (Donald Knuth's Web page, various TEX Users Groups, TEX FAQs, etc.) These should all be upgraded to point to `http://omega.cse.unsw.edu.au` .

2.2 Development site

The main Ω development site is `omega.cse.unsw.edu.au` . There also exists an Omega development site at `omega-system.sourceforge.net` , whose status is currently unclear.

Task 2.2.1. The rôle of `omega-system.sourceforge.net` should be clarified. It should be completely cleaned up, with everything being moved to `omega.cse.unsw.edu.au` . It may be appropriate to continue to use the site as a mirror, because of slow download times from Australia.

2.3 Mailing lists

There are currently two official Ω mailing lists.

- The *Omega Mailing List* (`omega@omega.cse.unsw.edu.au`) is the general mailing list, which should be used for finding general information, for support and for general notices.
- The *Omega Developers Mailing List* (`omega-developers@omega.cse.unsw.edu.au`) is the mailing list for discussing current development.

The first Ω mailing list was at `omega@ens.fr` , which is now shut down. There are also two no-longer-used Sourceforge Ω mailing lists (`omega-system-devel@lists.sourceforge.net` and `omega-system-glyphs@lists.sourceforge.net`). In addition, the `OMEGAMO@plain.co.jp` mailing list was created for using the Mojikyo font (`http://www.mojikyo.org`) with Ω.

Task 2.3.1. The archives of all of the Ω-related mailing lists need to be properly merged, so that they are all accessible from the Ω Web site. Work initiated by Roozbeh Pournader.

2.4 Source code repository

The Ω source code is stored in a CVS repository, currently at

```
:pserver:anonymous@omega.cse.unsw.edu.au:/home/cvs/root, password anonymous
```

Task 2.4.1. The Ω system is currently developed within the context of the TEX Live series of CD-ROM distributions, which is loosely based on Thomas Esser's teTEX collection of TEX-related software, teTEX being itself based on the `web2c` software suite, currently maintained by Olaf Weber. The interdependencies must be clarified, so that the Ω CVS repository need not hold all of TEX Live, itself a moving target. Work initiated by Roozbeh Pournader.

2.5 Bug tracking

The `bugzilla` software, developed for bug-tracking of the Mozilla open-source browser, has now been installed at `http://omega.cse.unsw.edu.au/bugs/index.cgi` .

Task 2.5.1. All of the known bugs must be inserted into the bug-tracker. A policy for bug-fixing must be defined. Work initiated by Roozbeh Pournader.

2.6 Release management

A complete Ω release includes many things. In addition to source code under the `web2c/omega...` source code directories, there is the `texmf/omega` directory of supporting files, scripts, fonts, that must be simultaneously distributed. In addition, documentation, preferably in several languages, must be distributed. Furthermore, the code must be amply tested on several platforms, to ensure that it behaves consistently from one platform to another.

Task 2.6.1. Define what the structure of an Ω release should look like. This means defining the directory structure, for the source code, the supporting files in the release, the Web site, etc.

Task 2.6.2. Define the rôles to be played by the different people who wish to participate in the development of Ω and in preparing Ω releases.

Task 2.6.3. Define the procedures for building and validating a release. This process should be as automated as possible, and update the Web site in the process.

Task 2.6.4. A number of Ω releases have been made in several years. It would be of interest for software archæologists to bring together as many old releases as possible. Keeping all of the old releases should also help in bug tracking and fixing.

Task 2.6.5. License cleanup should be undertaken. All files should have the GPL header addedd. Licenses should be provided with papers, for code, for documentation, and so on.

Task 2.6.6. From Chris Rowley: Detailed *technical* documentation of the system. We need to agree as precisely as possible what the new primitives and underlying new code are supposed to do in order to analyse bugs. It will allow us to reveal discrepancies in different perceptions.

2.7 Web site

The Ω Web site is currently an Intensional HTML page, which is interpreted by a special version of the Apache Web server, that interprets an *intensional context* contained in the URL and enclosed by angle brackets. Clicking on the different links gives a different *version* of the same Web page. The context is multidimensional, so this gives many possibilities for further development of the Web site.

Task 2.7.1. The introduction is quite old. It needs a serious rewrite.

Task 2.7.2. All text should be translated into as many languages as possible. This will require many volunteers. Switching language on the Web site will be done using the intensional framework.

Task 2.7.3. The Web site should be useful to the curious, the novice user and the expert.

Task 2.7.4. The current mascot, a picture of the *sumo yokozuna* Akebono, should be replaced, for licensing reasons. We should ask Duane Bilby to draw a traditional TEXish rendition of our mascot.

Task 2.7.5. Many Ω papers describe solutions for using Ω for different scripts and languages. For each paper, if it has been superceded by an official release, then a pointer to the new documentation should be provided.

Task 2.7.6. An Ω gallery, giving examples of books and other works produced using Ω, should be a highlight of the Web site.

2.8 Development languages

Currently, the Ω engine is defined as a series of Pascal Web change files on top of Knuth's original `tex.web`. In addition, several C files have been written to be linked with the C files generated from the Pascal Web by the `web2c` package. Additional Ω utilities have been written in Pascal Web, in C, in Perl, as well as in Java.

In the future, all Ω engine and driver development will take place in ISO C++, using the Standard Template Library. Therefore, existing Ω code will be migrated to C++, and all new code will be written in C++. Useful references for C++ are

- Bjarne Stroustrup. *The C++ Programming Language*, special edition. Addison-Wesley, 2000. ISBN 0-201-70053-5. Essential, as it includes the full language and library definitions. Previous editions of this book are woefully out of date.
- Matthew H. Austern. *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*. Addison-Wesley, 1999. ISBN 0-201-30956-4. Very useful.
- Nicolai M. Josuttis. *The C++ Standard Library: A Tutorial and Reference*. Addison-Wesley, 1999. ISBN 0-201-37926-0.

3 Polishing the Existing Distribution

The current Ω system is in use around the world. Although it is not currently designed with a high-level interface that makes it easily used by the average novice user, it does merit stabilization, so that those who are currently using it may use it reliably.

Choosing this path allows Ω's visibility to increase around the world, and will allow experimentation to channel further development. Not to overlooked is the book by Apostolos Syropoulos *et al.*, which is quite detailed in its discussion of the current Ω primitives.

The current Ω release, as available on the standard CVS repository, is 1.23. The documentation, not up-to-date, is available on the Web site.

3.1 Generating Typeset Output

There are several known bugs, which must all be documented and fixed. Here are some key ones.

Task 3.1.1. The interaction with the `kpathsea` primitives from the Ω engine is not safe. The Ω 16-bit character strings are converted to 8-bit character strings without any checking whether this is safe. Character set conversion should be undertaken when calling the `kpathsea` routines.

Task 3.1.2. The current implementation of local paragraph directives and of multidirectionality (change files `ompar.ch` and `omdir.ch`) has memory leaks. This seems to occur because of problems in modifications to the memory management routines to support these new features.

Task 3.1.3. The multidirectionality code has broken the generation of DVI code from leaders. This should be easily fixed.

Task 3.1.4. The rewrite in C of the font utilities needs to be fully debugged.

3.2 Portability

It is crucial that the code port trivially from one architecture to another. This is not always true.

Task 3.2.1. When Ω attempts to dump a format on a Mac OS X machine, it crashes. This is symptomatic of not enough attention being paid to portability, particularly with respect to byte-order and word-size issues.

3.3 Modification of standard utilities

The Ω system uses modified versions of a number of utilities, such as `dvipdfm`, `dvips` and `xdvi`. All of these evolve at their own rates, through the work of their main developers.

Task 3.3.1. For each of these utilities, define a protocol to work with the main maintainers so that changes to Omega specs are quickly reflected in the standard code bases, and vice versa.

3.4 Minimal Interface

It is not the purpose of this release to produce a full high-level interface to the Ω primitives as they currently stand. Nevertheless, it is quite reasonable to define basic interfaces for certain languages and scripts, so that people may use Ω while waiting for the infrastructure supporting the higher-level interfaces to be developed.

Task 3.4.1. Upgrade the programmer-level documentation.

Task 3.4.2. Create user-level documentation.

Task 3.4.3. Document the supported ΩTPs, ΩTP-lists, fonts, and higher-level environments.

Task 3.4.4. Negotiate with LaTeX3 team for compatible interfaces. See, for example, Frank Mittelbach and Chris Rowley, Language Information in Structured Documents: A Model for Mark-up and Rendering, *TUGboat* 18(3):199–205, 1997.

4 XML

Coming soon....

5 Fonts

Coming soon....

6 Going Multidimensional

In order to develop a high-level interface that can be used for high-quality multilingual typesetting, as well as being able to generate both structural and layout markup, we need some significant changes to the way in which the Ω engine functions.

In particular, we will need to introduce a *multidimensional context*, as in the aforementioned IHTML used in the Web site, that can be tested at any moment by the engine and by all of its components, in order to selectively adapt their behavior.

If the work in the previous section can still be considered working with some kind of improved TeX, the steps in this section take us well beyond. The references, available from the Web page, should be read in order to understand the tasks described below.

- John Plaice, Yannis Haralambous, and Chris Rowley. An extensible approach to high-quality multilingual typesetting. Submitted for publication, October 2002.
- Frank Mittelbach and Chris Rowley. Programming extensions needed in Ω. May 2001.

The tasks described below are not as detailed as above, given that some of them are still unclear. They will be developed into sections of their own, with subtasks, when appropriate.

Task 6.1. All fixed-size arrays in the engine will be replaced by dynamic C++ `vectors`.

Task 6.2. Characters will become 31-bit entities, rather than 16-bits. Modifications will be made to how `tangle` currently generates its code.

Task 6.3. Multidimensional macros, working with a current context, will be introduced.

Task 6.4. Fonts will be redesigned, to include an arbitrary number of parameters instead of the current four.

Task 6.5. Low-level typesetting will be redefined, to take advantage of the current context.

Task 6.6. The markup generation code will be modified so that it is integrated with the current context.

Task 6.7. Output from the typesetter will be provided in either a more flexible format, or in multiple formats (DVI, PDF, SVG). Details still need to be worked out.

Task 6.8. A high-level interface, usable by packages such as LaTeX or ConTeXt, will be defined.

7 Conclusion

Let's get to work....