

Lisp Infrastructure Development and Distribution

Christophe Rhodes*
Centre for Cognition, Computation and Culture
Goldsmiths College, University of London
New Cross Gate, London SE14 6NW, UK

June 10, 2005

Abstract

We have recently seen a dramatic improvement in the ease of assembly and use of freely-available Lisp software. In addition to support from Linux distributions, the development and propagation of tools such as `asdf` and `asdf-install` has made it noticeably easier than in the past for the individual user to configure and maintain a large body of software for their own use. Nevertheless, there are deficiencies in the current infrastructure which, if addressed, would further lower the barrier to participation in this ecosystem and allow the easier development of more code. This breakout session aims to identify and begin to address these deficiencies.

1 Introduction

Over the past few years, there has been a noticeable increase in the quantity of Free¹ Lisp software which is available for download and use; in parallel with this increase, there has also been considerable development of infrastructure for managing these software packages: in particular, `asdf-install` and CLiki. This development of infrastructure has meant that the software writers and users have largely been able to gain a wider perspective over the software ecosystem: rather than considering files, at the level of individual calls to `cl:load`, the user can often simply install a complex package, and its dependencies, with a simple Lisp function call. In addition, the focus of CLiki on DFSG-free software licences effectively reduces the burden of examining and understanding a multitude of different licence terms.

However, there remain problems: there is no automatic notification of new releases of software packages; there is no scope for uninstallation of installed software; there is no mechanism for declaration of relationships weaker than strict dependency; and there is no way to have simultaneously functioning versions of the same library.

Users of certain Linux distributions may recognize this list of problems as issues which are solved by the system; for instance, Debian's packaging tools allow specification of 'Recommends' and 'Suggests' as well as 'Depends', which are used to indicate that installing a second package

*c.rhodes@gold.ac.uk

¹This term is used here to mean "licensed under terms compatible with the Debian Free Software Guidelines"; the major point of interest in this discussion is that such software is freely redistributable.

would make the one asked for perform better in some way.² Further, it is possible to have multiple parallel installations of shared libraries, through cooperation with the system dynamic linker, and of binaries, through the use of ‘alternatives’.

Some of this functionality can doubtless be achieved very simply for individual pieces of software; for instance, software which does not depend on the name of its Lisp packages at runtime can be renamed to avoid a collision with a different version. However, there are cases where this will not work in its simplest form, and there are other potential areas of collision. This breakout session will investigate the technical and social merits of these and other features being added to the Lisp development infrastructure.

2 Aims

The aims listed below are not intended to be exhaustive. We write this preliminary list of items for discussion from the perspective of a user and developer of a number of Lisp software packages³ on Linux and Mac OS X, but we hope that items on this list are applicable to systems of which we are not aware.

- Document best practice with respect to distribution of code. There is a reluctance to introduce ‘external’ dependencies, with a preference to include instead a verbatim copy of non-trivial modules (for instance, the `Maxima` and `Slime` projects include conflicting versions of `nregex`).
- Identify policies for Lisp code to follow to allow enhancement of the infrastructure: from identifying idioms for ‘relocatable’ software, through extensions to `asdf` metadata information to providing for means of declaring dependencies on non-Lisp libraries.
- Develop or design `lint`-like software to check automatically for common violations of this policy, and to build conforming bundles from `asdf` descriptions.
- Develop or design software (possibly as an extension to `asdf-install`) which can cope with multiple versions of Lisp libraries installed. To encourage further development by the users, consider an extension to `asdf-install` to allow installation from remote version control rather than just tarballs.
- Find a mechanism for encouraging development and publishing of entry-level documentation, to allow initial evaluation of software without downloading it all.
- Discuss the range of existing software packages and identify areas where one package can enhance another, by providing optional features, another stage in a pipeline, or other such interactions.

²There is also an ‘Enhances’ field, indicating the reverse relationship.

³One possible aim for this breakout group would be to come to agreement on terms to disambiguate ‘package’ as software from its technical meaning in Common Lisp.