Lisp Infrastructure Development and Distribution

Christophe Rhodes

Goldsmiths College
University of London

Outline

- Where am I coming from?
- Making every developer count
- Technical barriers and overcoming them
- Social barriers and suggestions

Who am I?

- academic (boundary between Physics, Music and EEng)
- not primarily using Lisp for work (except if attending ECOOP is work)
- fallen into Lisp community “by accident”
- gentleman amateur
  - not writing Lisp to survive
  - exploring interesting (to me) ideas

What would I like to achieve?

- There are others like me!
- (e.g. SBCL developers: c. 15 amateurs; SLIME developers: c. 40 amateurs + one or two pros)
- work on things which interest them (us)
- how to make interesting things work well together
  - difficult problem
  - can it be made interesting?

Developer community

- Relatively small (~100)
  - some names pop up “everywhere”
  - (power-law distribution expected)
- Non-uniform culture
  - multitude of implementations
  - islands of communication (#lisp, info-mcl)
  - many people doing their own thing
  - NIH syndrome

- Competition versus coöperation
  - Both have their place
  - Competition: finding long-term optima
    - Improving on interfaces
    - no backwards-compatibility to break
  - Coöperation: maximizing network effects
    - using established protocols or libraries
    - saving time and work for difficult or interesting things

(aside: the DFSG)

- Debian Free Software Guidelines
  - “What is Free Software”?
    - Free Redistribution
    - Source Code
    - Derived Works
    - No Discrimination (People or Endeavour)
  - why is this useful?
    - I know that I can do interesting things with software under a DFSG-compliant licence
    - and I know that so can other people

Maximizing what we have

- Lower barriers to entry
  - make it easy to start using/hacking
    - package-level: asdf/asdf-install
      - missing pieces: updates, parallel installs
    - whole-environment level: lispbox
      - missing pieces: true integration
  - encourage useful hacking
    - easy access to version control
    - didactic use of source
    - reusable protocols

asdf-install

- Good things
  - it basically works
- Bad things
  - no support for
    - uninstallation
    - parallel installs
    - installation from CVS (SVN, arch, darcs, ...)
    - pre-downloading documentation

lispbox

- Good things
  - it basically works
- Bad things
  - doesn't provide
    - policy
    - integration
    - granular updates

NIH syndrome

- Often easier to reimplement something than use someone's package
  - particularly in Lisp!
    - friendly language
    - (often) poor documentation
- Even use of package can cause problems
  - SLIME and Maxima both use nregex.lisp
    - but both **include** copies directly
      - ... until recently, copies of **different** versions...

Libraries

- Is it hard (or possible?) to install two versions of one library
  - on the same filesystem
  - in the same lisp image
- Pros
  - “release” branch and “development” branch
  - less requirement for large coördination
- Cons
  - “release” branch and “development” branch

Upstream

- For fast-moving (α - or β -software) every user of released versions is suboptimal
  - better than not using at all, but worse than using HEAD
    - bug reports outdated
    - patches may not apply
  - FAIRLY_STABLE compromise
    - measures internal stability
    - does not capture dependence on state of other pieces of software

Policy (coöperation)

- Decrease complexity of problem space
  - remove choice except where genuine need for choice exists
  - propagate solutions to problems (even if not optimal solutions)
  - enable integration work by providing stable platform
  - also known as “Best Practice”

Summary: technical stuff

- asdf-install enhancements
- documentation extraction and publishing
- lint-like thing to support Best Practices
- better metadata
- identify interesting potential connections

Summary: social stuff

- policy / Best Practices
- better documentation
- communication between islands
- work on the “long tail”
- user-developer transition (or Ajax “total programming”)