ESPResSo: the future

Olaf Lenz, FIAS

14 July 2006

Overview

- ESPResSo's environment 2 years ago
- changes in the environment
- resulting problems
- possible solutions

ESPResSo's environment – two years ago

Two user types

- Core developers
  - work at MPI-P
  - work on the whole code
  - can read and write the CVS
- Users/Developers
  - mostly work at MPI-P
  - work on Tcl/Tk-scripts
  - work on a part of the C-code
  - can read an write the CVS

Communication

- core developers and users/developers meet regularly: (almost) everybody is at MPI-P
- mailing list and CVS log is less important

Development model

- source code is manageable
- everybody works on the Tcl/Tk-level as well as on the C-code
- everybody updates the code from CVS
- everybody commits code to the CVS

Documentation

- written documentation is of less importance, as core developers and users/developers are at the MPI-P
- the focus is on the source code documentation
- introductory docs, C-code docs, Tcl/Tk-docs and developer's docs are intermingled (doxygen)

Changes in the environment

Changes in the environment

- MPI-P is no longer in the center of ESPResSo's development
- core developers are not at MPI-P
- ESPResSo spreads!
- this results in new user types:
  - users that only work on the Tcl/Tk-script level
  - beginners that have few contact to other users
  - in the following: *Tcl/Tk-users*
- field of interest of the core developers has changed

Problems

Tcl/Tk-users' problems

- high initial barrier
  - access to the source code is relatively complicated
  - compiling the source code is complicated
  - many dependencies: MPI, Tcl/Tk, FFTW
- intermingling of C-code docs and Tcl/Tk-docs is confusing
- to have up-to-date code, CVS is required (no source distribution available)
- no forum for beginner's questions

User/Developers' problems

- keeping the source code up-to-date via CVS is laborious
- relatively small changes require changes in the core files of ESPResSo
- new code that is not yet to be committed to the CVS has to be merged into every single updated version – or one big merge at the end
- new developments should not always be immediately accessible for everybody (bad tested code, new developments)

User/Developers' problems 2

- many conventions exist only in the core developers' heads, nothing is written on
  - strategical aims
  - coding standards
  - documentation standards
  - CVS policy (when to commit what changes?)
- core developers are less easy to reach

Core developers' problems

- overview over the growing code is getting harder
- basic changes are getting harder
- less contact between core developers: strategical aims are less clear

Other problems

- so far only few input from external developers
  - no communication with core developers?
  - unclear CVS policy (when to commit?)
- because of the growing user community, not everybody can have CVS access
- When somebody asks me for ESPResSo: Which version do I give out?

Possible solutions

Dividing the documentation

- User's manual
  - contents: tutorial, general ideas, algorithms, manual of the Tcl/Tk-commands
  - classical manual (to be read front to back)
  - useful for Tcl/Tk-users and beginners
  - lowers initial barrier
- Developer's documentation
  - contents: C-code docs (doxygen), coding and docs standards, CVS policy, strategical aims
  - e.g. in doxygen
  - makes conventions explicit

Modularising the source code

- modularisation on the source code level is already good!
- missing: definition of the interfaces
- ideally, modules are even more independent of the core:
  - no changes to any core file are necessary
  - can be compiled independently
  - can be versioned independently
- a set of “core modules” could be added to the core

Modularisation 2

- Pros:
  - core code becomes simpler
  - independent development of modules
  - as long as the interface is unchanged, modules and core are independent
  - modules can be tested, published and distributed independently
  - no CVS access required for module development
  - enables consistent release policy
- Cons:
  - probably performance loss?

Source code distribution

- source code should be downloadable on the home page
- download of modules could be independent
- minor version number is bumped when interface changes
- a *release* only contains a well-tested, stable core and core modules
- meaningful release policy!

... any other ideas?