The build system of ABINIT 5

Y. Pouillon[1,2], X. Gonze[1], T. Deutsch[3]

(1) Université Catholique de Louvain, Louvain-la-Neuve, Belgium
(2) Universidad del País Vasco (UPV/EHU), Donostia-San Sebastián, Spain

(3) Commissariat à l'Énergie Atomique, Grenoble, France

2007/01/29

Outline

Outline

4 → 5: the big mutation

Breaking the monolith

- Better conformance to top-level coding standards
- Better separation between software components
- From benevolent dictatorship to participative democracy

1. Prepare the code, aka *"beautification"* (4.4.3)
2. Add support for the GNU Autotools (4.4.3 → 5.3.3)
3. Strengthen code quality checks (4.4.3 → 5.2.3)
4. Improve flexibility of test suite (4.5.3 → ?)
5. Restructure and enhance documentation (4.5.3 → ?)

4 → 5: the big mutation

Breaking the monolith

- Better conformance to top-level coding standards
- Better separation between software components
- From benevolent dictatorship to participative democracy

1. Prepare the code, aka *"beautification"* (4.4.3)
2. Add support for the GNU Autotools (4.4.3 → 5.3.3)
3. Strengthen code quality checks (4.4.3 → 5.2.3)
4. Improve flexibility of test suite (4.5.3 → ?)
5. Restructure and enhance documentation (4.5.3 → ?)

A modular source tree

- Component-driven restructuring
  - note: most names become lowercase
  - `Lib_*` $\longrightarrow$ `lib/*`
  - `Src_*` $\longrightarrow$ `src/*` (levels have been refined)
  - `Test_*` $\longrightarrow$ `tests/*`
  - `Infos` $\longrightarrow$ `doc/*`
  - `Utilities` $\longrightarrow$ `util/*`
- Adding files required by the GNU Coding Standards
  $\longrightarrow$ `COPYING`, `INSTALL`, `NEWS`, `...`
- Adding `extras/`: miscellaneous add-ons
- Adding `config/`: new build system
- Adding `configure.ac` = master file of the build system

Three classes of contributors

- End-users: 4 $\longrightarrow$ 5 = more comfort
  - need not know about the build system
  - need not install the GNU Autotools
  - may help improve compiler support
- Developers: 4 $\longrightarrow$ 5 = pre-build stage
  - have to install the GNU Autotools
  - have to know the public part of the build system
  - should not hack into the build system
    $\longrightarrow$ use provided config files instead
- Maintainers: 4 $\longrightarrow$ 5 = better-defined tasks
  - have to know the whole build system quite well
  - may modify and extend the build system with great care
  - `configure.ac`: the order of the commands is CRITICAL

Three classes of contributors

- End-users: 4 $\longrightarrow$ 5 = more comfort
  - need not know about the build system
  - need not install the GNU Autotools
  - may help improve compiler support
- Developers: 4 $\longrightarrow$ 5 = pre-build stage
  - have to install the GNU Autotools
  - have to know the public part of the build system
  - should not hack into the build system
    $\longrightarrow$ use provided config files instead
- Maintainers: 4 $\longrightarrow$ 5 = better-defined tasks
  - have to know the whole build system quite well
  - may modify and extend the build system with great care
  - `configure.ac`: the order of the commands is CRITICAL

Three classes of contributors

- End-users: 4 $\longrightarrow$ 5 = more comfort
  - need not know about the build system
  - need not install the GNU Autotools
  - may help improve compiler support
- Developers: 4 $\longrightarrow$ 5 = pre-build stage
  - have to install the GNU Autotools
  - have to know the public part of the build system
  - should not hack into the build system
    $\longrightarrow$ use provided config files instead
- Maintainers: 4 $\longrightarrow$ 5 = better-defined tasks
  - have to know the whole build system quite well
  - may modify and extend the build system with great care
  - `configure.ac`: the order of the commands is **CRITICAL**

Building ABINIT 5

- Important: no need to install the GNU Autotools
  $\longrightarrow$ everything contained in the `configure` script

- First create a build directory, e.g.:
  "`mkdir tmp && cd tmp`"
  $\longrightarrow$ will preserve a clean source tree
  $\longrightarrow$ several builds with one source tree
  $\longrightarrow$ something you should always do

- Then follow the traditional build trilogy:
  1. `../configure [options]`
  2. `make`
  3. `make install`

Building ABINIT 5

- Important: no need to install the GNU Autotools
  $\longrightarrow$ everything contained in the `configure` script

- First create a build directory, e.g.:
  "`mkdir tmp && cd tmp`"
  $\longrightarrow$ will preserve a clean source tree
  $\longrightarrow$ several builds with one source tree
  $\longrightarrow$ something you should always do

- Then follow the traditional build trilogy:
  1. `../configure [options]`
  2. `make`
  3. `make install`

Building ABINIT 5

- Important: no need to install the GNU Autotools
  $\longrightarrow$ everything contained in the `configure` script

- First create a build directory, e.g.:
  "`mkdir tmp && cd tmp`"
  $\longrightarrow$ will preserve a clean source tree
  $\longrightarrow$ several builds with one source tree
  $\longrightarrow$ something you should always do

- Then follow the traditional build trilogy:
  1. `../configure [options]`
  2. `make`
  3. `make install`

Tuning the configuration

- By default: auto-detection of compilers, libraries, ...
- Without options: make the safest build possible
  $\longrightarrow$ built-in database of optimisations
- Tunable support for external libraries
- `--prefix=DIR`: install into `DIR`
- `--with-<compiler>-optflags=DIR`:
  user-defined optimisations (*<compiler> = cc, cxx, fc*)
- `--disable-mpi`: disable build of parallel code
- `--enable-<lib>`: activate support for <lib>
  $\longrightarrow$ *<lib> = linalg, netcdf, etsf-io, etsf-xc, xmlf90, fftw*
- `--help`: obtain the list of recognised options

Using a config file

- Purpose: set-up options once and for all
- Shell-script style (sourced by the *configure* script)
  $\longrightarrow$ replace "-" by "_" in command-line option names
- Options may be saved
  - system-wide: `/etc/abinit/build/<hostname>.ac`
  - per-user: `$HOME/.abinit/build/<hostname>.ac`
  - per-source-tree: `<top_source_dir>/<hostname>.ac`
  - per-build-dir: `<top_build_dir>/<hostname>.ac`
- Priorities

  system-wide < per-user < per-source-tree < per-build-dir
  ALWAYS overriden by command-line

- Template & examples in `<source_dir>/doc/config/`

The build stage itself

- Standard practice: `make` from the top source dir
- Build one executable: `make <binary_name>`
- Build a component: `cd <component_dir> && make`
- `make check`: build binaries and perform selected tests
  $\longrightarrow$ *still under development*
- `make dist`: create a source tarball
- `make distcheck`:
  - create a source tarball
  - uncompress tarball in a temporary directory
  - compile with default options
  - performed specified tests
- `make binary_package`: build all & create binary tarball

Performing tests

Going into the `tests/` directory

- `make`: obtain help on how to perform tests
- `make test_in`: perform built-in tests
- `make test_<series> start=#a stop=#b`
  - perform tests of <*series*>
    $\longrightarrow$ *cpu, fast, physics, tutorial, v1, v2, v3, v4, v5*
  - start at test #a
  - stop at test #b
  - results stored in `<series>/„<host>_<arch>_<date>`
  - to perform only one test: use either *start* or *stop*
  - omitting *start* and *stop*: perform whole series
    $\longrightarrow$ may require > 1 Gb of free disk space

Installing binaries and data

- From the top source dir: `make install`
- Default install prefix: `/opt/etsf`
- Without arguments
  - use `<prefix>/abinit/x.y/` as base directory
  - install binaries in `<base_dir>/bin/`
  - install tests in `<base_dir>/tests/`
  - install documentation in `<base_dir>/doc/`
  - install wrapper script in `<prefix>/bin/`
    $\longrightarrow$ *still under development*
- `make install prefix=DIR`: change prefix for DIR
- `make install DESTDIR=DIR`
  - install in `<DESTDIR>/<prefix>`
  - main use: Debian, RPM, and Gentoo packages

Filing a build-time bug report

Important note

- The `configure` script is not meant to be hacked
- Asking help is better than messing-up with the build system

- Information needed by the maintainers
  - nature of the bug
    $\longrightarrow$ crash, faulty behaviour, feature request, ...
  - error messages and moment of the bug
    $\longrightarrow$ configure-time, build-time, install-time
  - please provide `config.log`
  - please provide `config.mk` if present
  - please provide `config.h` if present
- For now: `mailto:yann_pouillon@ehu.es`

Outline

Structure of the build system

Subdirectories of `config/`

- *scripts/* : build scripts (front-end: *makemake*)
  $\longrightarrow$ very core of the build system
- *m4/* : Autoconf macros for ABINIT
  $\longrightarrow$ shell-script commands included into `configure`
- *makefiles/* : data to add to intermediate makefiles
- *optflags/* : database of optimisation flags
- *build-examples/* : various build configurations
  $\longrightarrow$ format: <*variable*>***<tab>***<*value*>
- *robodoc/* : data for on-line documentation
- *detect/* : improved compiler detector (*under development*)
- *gnu/* : Autotools stuff (please ignore)

Public part of the build system

Self-documented files in `config/specs/`:

- `libraries.cf`: description of internal libraries
- `binaries.cf`: description of binaries
- `tests.cf`: description of the test suite
- `documents.cf`: structure of the documentation
- `extlibs.cf`: management of external libraries
- `other.cf`: non-source directories of the source tree
- `env.cf`: relevant environment variables
- `options.cf`: options of the `configure` script
- `autoconf.cf`: additional information for `configure`
- `linalg.cf`: enhanced linear algebra support
  $\longrightarrow$ *under development*

ABINIT 5 developing howto

- Modifying files
  $\longrightarrow$ regenerate Fortran interfaces if needed (`abilint`)
- Adding, renaming or removing files
  $\longrightarrow$ update `abinit.src` and `abinit.amf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding a directory
  $\longrightarrow$ edit `config/specs/(libraries|binaries).cf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding an external library
  $\longrightarrow$ write `lib/<libname>/<libname>.mk`
  $\longrightarrow$ install prefix: `lib/<libname>/tmp`
  $\longrightarrow$ edit `libraries.cf` and `extlibs.cf`
  $\longrightarrow$ run `./config/scripts/makemake`

ABINIT 5 developing howto

- Modifying files
  $\longrightarrow$ regenerate Fortran interfaces if needed (`abilint`)
- Adding, renaming or removing files
  $\longrightarrow$ update `abinit.src` and `abinit.amf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding a directory
  $\longrightarrow$ edit `config/specs/(libraries|binaries).cf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding an external library
  $\longrightarrow$ write `lib/<libname>/<libname>.mk`
  $\longrightarrow$ install prefix: `lib/<libname>/tmp`
  $\longrightarrow$ edit `libraries.cf` and `extlibs.cf`
  $\longrightarrow$ run `./config/scripts/makemake`

ABINIT 5 developing howto

- Modifying files
  $\longrightarrow$ regenerate Fortran interfaces if needed (`abilint`)
- Adding, renaming or removing files
  $\longrightarrow$ update `abinit.src` and `abinit.amf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding a directory
  $\longrightarrow$ edit `config/specs/(libraries|binaries).cf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding an external library
  $\longrightarrow$ write `lib/<libname>/<libname>.mk`
  $\longrightarrow$ install prefix: `lib/<libname>/tmp`
  $\longrightarrow$ edit `libraries.cf` and `extlibs.cf`
  $\longrightarrow$ run `./config/scripts/makemake`

ABINIT 5 developing howto

- Modifying files
  $\longrightarrow$ regenerate Fortran interfaces if needed (`abilint`)
- Adding, renaming or removing files
  $\longrightarrow$ update `abinit.src` and `abinit.amf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding a directory
  $\longrightarrow$ edit `config/specs/(libraries|binaries).cf`
  $\longrightarrow$ run `./config/scripts/makemake`
- Adding an external library
  $\longrightarrow$ write `lib/<libname>/<libname>.mk`
  $\longrightarrow$ install prefix: `lib/<libname>/tmp`
  $\longrightarrow$ edit `libraries.cf` and `extlibs.cf`
  $\longrightarrow$ run `./config/scripts/makemake`

Preprocessing options

- Standardised naming conventions (*still in progress*)
  - Now following the GNU coding standards
  - Capital letters, digits and underscores **only**
  - Optional features: **`HAVE_*`**, e.g. `HAVE_NETCDF`
- Organised by sections: architectures, compilers, MPI, ...
- Mandatory in all routines (use `mkroutine`):

```
#if defined HAVE_CONFIG_H
#include "config.h"
#endif
```

- Full description & correspondences in (yet incomplete): `doc/developers/preprocessing-options.txt`

Outline

Moving on with the Autotools

- GNU M4
  - 1.4.4 still OK but 1.4.8 recommended (performance)
  - version 2.0 will bring many great improvements
- Autoconf 2.61
  - 2.59 $\rightarrow$ 2.60: issue fixed in ABINIT 5.3
  - will ease support for NetCDF
  - now requires Perl
- Automake 1.10
  - still better Fortran 9x support
  - full support for Python
- Libtool
  - version 1.5.22 still up-to-date
  - version 2 likely to be released in 2007
- Autotest: coming soon, yet interface still may change

Enforcing abirules

- Dependency tree of the routines now available
  - helpful for maintainers & developers
  - violations of the abirules identified
    $\longrightarrow$ dependencies on routines from higher levels
- Possible fixes
  - change levels of some directories
  - move some routines into other directories
  - rewrite parts of some routines or split them

$\longrightarrow$ Very delicate operation
$\longrightarrow$ To be scheduled for next beautification

Improving the test suite

Wish list

1. Improve clarity and ease of use
2. Consider optional features only when enabled
3. Facilitate bug fixing

- Structure type: *v1, v2, v3, . . .*
  $\longrightarrow$ OK for not-too-big monolithic code
  $\longrightarrow$ conflicts with enhanced modularity
- Running well within the sources, not so well elsewhere
  $\Longrightarrow$ have it truly location-independent
- Prepare integration with Autotest

Next-generation test suite: proposal

- No interference with current test suite
  - independent project
  - semi-automatic conversion procedure $\longleftarrow$ feasible
- New structure: topic-based
  - restrict checking to modified part
  - pinpoint bugs much more easily
- Atomic tests
  - Each test completely independent from others
    $\Longrightarrow$ group a few tests together
  - Single file with description, instructions, and inputs
    $\Longrightarrow$ automatic list of input variables
    $\Longrightarrow$ automatic generation of on-line documentation
- Running everywhere
  - Python-based?
  - GUI?

Better handling of input variables

- Database of input variables
  - much better overall integration
  - on-line editing made easy
  - XML descriptions $\Longrightarrow$ any other format
- Automatic generation of documentation
  - HTML, PDF, for GUIs
  - sections, tables of contents, ...
- Links with test suite
  - which tests are using a variable
  - which variables are used by a test
- Links with routines
  - which routines are using a variable
  - which variables are used by a routine

Version control with Bazaar-NG

- Important features of `bzr`
  - depends on Python 2.4 **ONLY** $\Longrightarrow$ very high portability
  - fully decentralised $\Longrightarrow$ local and off-line commits
  - GUIs available $\Longrightarrow$ no need for special skills
  - easily extendable $\longrightarrow$ Python plug-ins
  - mixed distributed/centralised structure possible
  - partial merges and cross-merging made easy

- Strategy
  1. Wait for first stable version (spring 2007)
  2. Ensure Python 2.4 support by all ABINIT committers
  3. Mirror the current archive and prepare scripts
  4. Train the committers
  5. Activate new archive

Version control with Bazaar-NG

- Important features of `bzr`
  - depends on Python 2.4 **ONLY** $\Longrightarrow$ very high portability
  - fully decentralised $\Longrightarrow$ local and off-line commits
  - GUIs available $\Longrightarrow$ no need for special skills
  - easily extendable $\longrightarrow$ Python plug-ins
  - mixed distributed/centralised structure possible
  - partial merges and cross-merging made easy

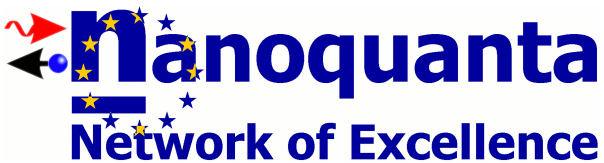
- Strategy
  1. Wait for first stable version (spring 2007)
  2. Ensure Python 2.4 support by all ABINIT committers
  3. Mirror the current archive and prepare scripts
  4. Train the committers
  5. Activate new archive

Running ABINIT on 1000 processors?

- I^3 infrastructure @ Barcelona Computing Center (Spain)
  - High-Performance Computing on > 1000 processors
  - Scientific challenges: justify the tuning of codes
  - Reach-out: create user communities $\longrightarrow$ "ETSF users"
- Collaborations with CINECA (Italy) and IDRIS (France)
- Computer scientists hired to tune the codes
- Around 3 M€ over 7 years
  $\longrightarrow$ traveling, PhD students, post-docs, reach-out, computing time
- Full acceptance of Free Software philosophy
- Siesta in progress, Octopus scheduled
- ABINIT: need for a coordinator within Nanoquanta/ETSF

Acknowledgments

- Funding by the Nanoquanta NoE

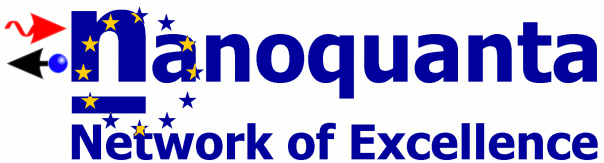


- Nanoquanta IT9 & ETSF File Formats Team (*ETSF I/O*)
  Special thanks to D. Caliste!
- M. Marques, M. Oliveira (*ETSF XC*)
- Organizers of the ABINIT workshop

Thank you for your time!

Acknowledgments

- Funding by the Nanoquanta NoE

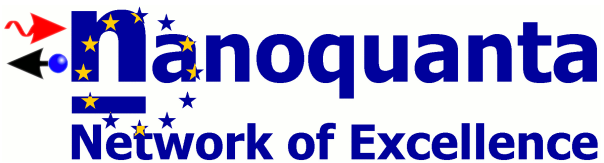


- Nanoquanta IT9 & ETSF File Formats Team (*ETSF I/O*)
  Special thanks to D. Caliste!
- M. Marques, M. Oliveira (*ETSF XC*)
- Organizers of the ABINIT workshop

Thank you for your time!

Acknowledgments

- Funding by the Nanoquanta NoE

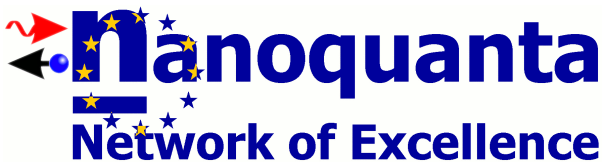


- Nanoquanta IT9 & ETSF File Formats Team (*ETSF I/O*)
  Special thanks to D. Caliste!
- M. Marques, M. Oliveira (*ETSF XC*)
- Organizers of the ABINIT workshop

Thank you for your time!

Acknowledgments

- Funding by the Nanoquanta NoE

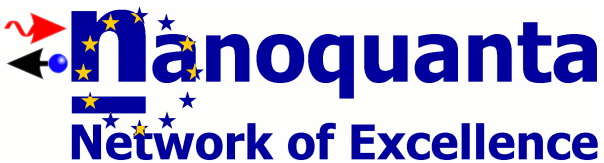


- Nanoquanta IT9 & ETSF File Formats Team (*ETSF I/O*)
  Special thanks to D. Caliste!
- M. Marques, M. Oliveira (*ETSF XC*)
- Organizers of the ABINIT workshop

Thank you for your time!

Acknowledgments

- Funding by the Nanoquanta NoE



- Nanoquanta IT9 & ETSF File Formats Team (*ETSF I/O*)
  Special thanks to D. Caliste!
- M. Marques, M. Oliveira (*ETSF XC*)
- Organizers of the ABINIT workshop

Thank you for your time!