Good (or not-so-bad) practices for QUANTUM ESPRESSO development

P. Giannozzi
Università di Udine and IOM-Democritos, Trieste, Italy

27 March 2013

– Typeset by FoilTEX –

The QUANTUM ESPRESSO development framework

Currently, development in QUANTUM ESPRESSO is

- geographically distributed (though mostly concentrated in few places)
- often performed on independent or loosely bound parts of software, but occasionally extensive changes to core routines are needed
- not subject to a strict centralized control or acceptance rules
- not following strict programming rules.

Problems: *How to ensure a coherent and effective development?*
How to avoid conflicts, duplication of efforts, breaking existing stuff...?
How to guarantee maintainability in the future?

(Of course, there is no unique and ultimate answer)

General rules

The ideal procedure to be followed depends upon the kind of new development you have in mind (see next slides).

Unless you plan something for your own usage with no ambition to become part of QUANTUM ESPRESSO, or to be long-lived: *use svn*.

If you need to modify something in QUANTUM ESPRESSO, *modify the svn version, not a stable, released version*, or *open a svn branch* (see next slides). The time you will spend in the beginning and during the development will be amply compensated by the time spared at the end.

Before doing anything, *think* a little bit how you can make your contribution easier to maintain. Sometimes it is preferrable to make extensive but well-structured changes rather than introducing tricks and workarounds in order to minimize changes!

A taxonomy of QUANTUM ESPRESSO contributions

- "Small" changes, i.e. limited to a single routine or groups of routines, to an existing package (e.g.: adding a new GGA exch-corr functional)
- "Big" changes, i.e. involving extensive changes and long development times, to an existing package (e.g.:Car-Parrinello with **k**-points)
- A new package that uses unmodified routines (or with minimal modifications) from QUANTUM ESPRESSO (e.g.: GIPAW)
- A new package that uses modified or old versions of routines from QUANTUM ESPRESSO (e.g. Genetic Algorithm, EPW)
- A new "external" package that just reads data produced by QUANTUM ESPRESSO

Simple case 1: Small changes

Ideal procedure for simple modifications:

1. Download svn trunk, either as a developer or anonymously
2. Make your changes, verify if everything works and if existing functionalities have been broken
3. If everything works, save a copy of what you did (just in case), update the svn, solve conflicts if any
4. If you have permission and if you feel confident, commit
5. If you do not have permission or if you do not feel confident, send modified files to a more confident/authorized developer

Simple case 2: New package, no other changes

To add a package using QUANTUM ESPRESSO routines, unmodified or with very small changes, you have the choice between

1. adding a new directory to the svn of QUANTUM ESPRESSO, along the lines of existing packages;

2. opening a new project on `qe-forge.org`, working there.
   In this case, you are advised to make your svn *visible to other developers*. In this way they can download your package into the svn of QUANTUM ESPRESSO (using `svn propedit externals`), so that if changes are needed to modules or routines used by your package, the other developers can make the appropriate changes without breaking your package.

Simple case 3: “external packages”

To add a package that just uses data produced by QUANTUM ESPRESSO, you are advised to open a new project on `qe-forge.org` and work there. If changes are needed to data file formats, or if you need to modify the routines reading the data file(s), contact the QUANTUM ESPRESSO developers.

It may be a good idea to import the installation mechanism of QUANTUM ESPRESSO: it will make life easier for you as well as for us (when implementing download on demand)

Not-so-simple cases

For more extensive/dangerous/invasive changes, the recommanded procedure is to make a svn branch, using `svn cp`. Example:

```
svn cp \\
svn+ssh://NAME@qeforge.qe-forge.org/svnroot/q-e/trunk/espresso \\
svn+ssh://NAME@qeforge.qe-forge.org/svnroot/q-e/branch/myespresso \\
-m "Branching at Rev.12345"
```

(you may get the revision number using `svn info | grep Revision`)

Work in your branch: nothing nasty will happen to the trunk. From time to time, align your branch with the trunk, using `svn merge`. Example (you may choose to align to a given revision):

```
svn merge [-r REVISION] \\
svn+ssh://NAME@qeforge.qe-forge.org/svnroot/q-e/trunk/espresso
```

The final merge of the branch back into the trunk is relatively easy

More difficult cases

- *I have written a new package calculating the Answer (again 42) that works on top of QE v.-1, how can I bring it to the latest version?*
  Bring your package into svn revision closest to v.-1; when everything works, move the revision gradually up, removing conflicts.
  Alternative: bring your package into latest svn, find out what to do.

- *I have implemented calculation of the Answer (42) by modifying v.-2 of QE, how can I bring it to the latest version?*
  Make a diff between your modified version and v.-2; apply the same changes to the svn revision closest to v.-2; when everything works, move the revision gradually up, removing conflicts. Alternative: apply your changes to the latest svn, figure out what has to be done.

Unless the needed changes are localized to a few places, it is a slow and time-consuming operation, that may or may not take place in practice.