The ERP5 source code repository has evolved over the last two years with more contributions than ever and even stricter quality assurance.

However, if we look at statistics provided by Ohloh, it looks like contributions have decreased. One could draw easily wrong conclusions from such a diagram [1]:

The diagram seems to show that in 2011, the number of contributions suddenly decreases. However, the number of contributions actually increased as all ERP5 developers know.

Let us analyze what causes this apparent drop. The following events happened in 2011:

1. Adoption of git version control.
2. ERP5 repository was split into multiple repositories.
3. Adoption of even stricter quality assurance rules.

ERP5 adopted in 2011 git instead of SVN for version control. Due to its technical nature, git is better used by splitting large projects into smaller independent ones. We thus decided in May 2011 to split ERP5 project smaller repositories.

1. SlapOS repository contains all the code which is used to install ERP5. [2]
2. cloudooo repository contains the code document conversion. [5]
3. JIO repository contains the code of ERP5 future API to handle input/output in javascript forms. [6]
4. renderjs repository contains the core rendering code of ERP5 future user interface. [7]
5. neoppod repository contains the distributed transactional NoSQL object database optimized for big data in ERP5. [8]
6. etc.

If we only consider SlapOS repository [3], its activity from 2011 is higher than the highest activity of ERP5 in the years 2003-2011 as the following diagram shows [4].

Overall, the total activity in all ERP5 repositories from 2010-2013 reaches a total of 777 commits per month in average, compared to 530 in 2007-2010 and 291 in 2004-2007 [11]. This means, the commit activity in 2010-2013 is 167% higher compared to 2004-2007.

The introduction of git had other consequences too. Instead of doing many commits (and reverts) in a day as we used to do with SVN, developers using git tend to commit their contributions locally first. After a couple of days, they review their local commits, group them in a consistent way, add useful documentation to each commit group and of course do not push in the shared repository any commit which they have reverted. The adoption of git development style necessarily leads to contribution statistics which are lower than with SVN. This lower contribution is actually the sign of increased quality in the contribution process rather than a sign of less contributions.

Examples of major contributions which could be achieved this way include parallel expand in ERP5 simulation, portal types as classes, new simulation, etc. They are all major contributions which have brought ERP5 to levels of scalability which make it ready for 24/7 mission critical, big data operations such as Big&Go tolling system.

In parallel to git, we also introduced a new rule: contributions must not bring unit test regressions.

Before any code is accepted into the master branch of ERP5, developers must now run all of the 4000+ unit tests. Since it would take more than 12 hours to run all tests on a developer workstation, we use a SlapOS cloud with more than 120 cores and provide continuous integration for all branches of all projects of all developers in near real time. A continuous integration module was added to ERP5 to replace our old buildbot environment. Thanks to this approach, the ERP5 repository no longer suffers from regressions introduced by poorly tested commits. The quality of commits also improved thanks to git cherry picking.

Nowadays, developers work on branches [9], check unit test results, fix regressions and then push their changes into master, usually after grouping their commits with good documentation. For example, all developments related to ERP5 future ZODB-based code management system are held in a specific branch.

ERP5 based client projects also happen in a branch. Some times, it is necessary to add quick and dirty hacks in order to meet client deadlines. Those quick and dirty hacks are no longer contributed to ERP5 main branch. Developers are now requested to find a clean and generic solution first before contribution to ERP5 main branch any improvement driven by a client project.

Overall, as ERP5 is now a mature project, we chose to reduce a bit the speed of contributions and raise one level higher our quality assurance standards on every contribution.

Let us now look again at Ohloh's statistics and find out who are the biggest contributors to ERP5. Apparently, one contributor to ERP5 has a very high score. He has been able to contribute multiple times a day during many months. If you look at contributions, most of them relate to a change of title, a minor modification of a form, changing tabs into whitespaces in python code or removing trailing spaces before a new line. Another contributor with 20 apparently times less contributions
is Yoshinori Okuji who has contributed to ERP5 a long list of major improvements and innovative designs which make ERP5 shine nowadays in enterprise applications. With this example, one understand how important it is to take time to interpret statistics accurately.

As a conclusion, here is the github history of ERP5 contributions in the last 52 weeks. I'm sure many project would love to have such statistics [10].

References:
2. Old buildout folder in erp5 repository: http://git.erp5.org/gitweb/erp5.git/history/11aa0af8e6ceb0c:/buildout?js=1
5. cludooo repository: http://git.erp5.org/gitweb/cludooo.git?js=1
6. jio repository: http://git.erp5.org/gitweb/jio.git?js=1
7. renderjs repository: http://git.erp5.org/gitweb/renderjs.git?js=1
8. neoppod repository: http://git.erp5.org/gitweb/neoppod.git?js=1
9. list of erp5 branches: http://git.erp5.org/gitweb/erp5.git/heads?js=1
10. github last commit statistics on ERP5 repository: https://github.com/erp5/erp5/graphs/commit-activity