PyPy: Agile methods in an EU-funded Open Source project

Beatrice Düring, PyPy (Change Maker) bea@changemaker.nu
www.pypy.org/http://codespeak.net/pypy
(Experience report proposal, 31-01-2006)

Can agile methods be employed in an Open Source project with a distributed/dispersed development style while also being funded by European Commission, following a fixed price contract model with structural requirements much more similar to plan-driven approaches? In this paper we will present the agile methods that made this hybrid structure not only possible but that formed the basis of a successful first year of the PyPy project.

The PyPy project started as a grass root effort in the Python Community, building a highly flexible and fast Python implementation purely written in Python. After less than a year of work the community and the core developers started to work on a proposal for EU-funding (summer 2003). The partially funded EU-project started in December 2004 and have, in January 2006, successfully passed the EU review of the results of the first year. A consortium of seven companies and one university was created around and within the original community in order to fulfill the EU-contract.

Test-driven development
The PyPy project have since it´s inception been "test-driven" and has employed automated test suites for language compliance tests (Python) as well as for function tests. This test framework together with an extensive "coding style" guide (covering style of code, style of tests, naming conventions etc) and version control support (Subversion) created the platform that allows for continuous integration into the code base.

The almost extreme test-driven environment and the evolved automated tools and rigorous version control (of documentation as well as code) made qualitative, distributed work in the community possible. It has also been one of the reason for why the community grew – inviting developers and giving them full commit rights (no hierarchies) to the code base from the start. In that sense, keeping the same approach for test-driven development for the consortium based work of PyPy as well as having all work focused into one shared development area was, and is, instrumental for balancing the interests of the community and the consortium.

Open and transparent communication
Another agility aspect relates to transparent and open communication within the project and between the project and the community. Only very few (EU-contract related) documents are access restricted, everything else is freely available.
Seen from purely development perspective there us no, or very small, differences being made between the community contribution and the development work of the consortium companies. Developer mailing lists and IRC channels are open for the entire community – the consortium adheres to these communication rules.

Moreover, the PyPy developers implemented a method with weekly 30-minute IRC chat meetings where topics were briefly discussed, delegated or decided upon. Those meetings are open to all active developers and usually do not touch upon internal EU matters much except that funded developers keep EU goals more in mind than others. Minutes of these weekly developer meetings get archived and posted to the development list.

The formal project organization required by the EU imposed more structure on the previous more free-floating agile process. Roles and responsibilities where staked out,
conforming with the requirements of the roles but delegating as much as possible of the responsibilities and decision-making to the core developers. The strategy was to
keep "conceptual integrity" (Brooks) of the vision and the idea in the hands of the core developers. A somewhat negative result was the added workload and responsibility on developers regarding EU related work. It was also a conscious strategy to employ the same version-control/review based scheme regarding EU documents and consortium level issues as those being used in the

technical development.

Sprints

PyPy first started during a one-week meeting, a *"sprint"*, held at Trillke-Gut in Hildesheim February 2003. The sprint was inspired by practices used by other Python projects such as Zope3. Originally the sprint methodology used in the Python community grew from practices applied by the Zope Corporation. Their definition of a sprint was: *"two-day or three-day focused development session, in which developers pair off together in a room and focus on building a particular subsystem" (1).* Inspired by practices such as pair programming in eXtreme Programming, sprints were first used within the commercial work and later tried and used within the Open Source context around Zope development.

There seems to be no specific sources relating the Zope/Python version of sprinting to the terminology used in Scrum, signifying an iteration around a specific increment – lasting up to a month, although it could be viewed as an accelerated version with a more lightweight process of less, or no specific documentation and roles tied to it.

Sprinting spread through the Python community and today almost all Python projects in the Python Open Source community sprint at least once every year.

Sprinting up to a week became the initial driving factor in developing the code base and the community/people around PyPy. The early (no-funded) PyPy sprints were organized by core developers together with local Pythonistas in Louvain La Neuve, Gothenburg, Vilnius and Amsterdam. Sprints gave the opportunity to both help, participate and influence the ideas within PyPy. Sprint preparation and planning as well as the actual organizing rotated between the developers, using their contacts and networks to identify locations and facilities to sprint in to as low costs as possible for both travels and accommodation for the sprint attendants. Already from the start the strategy to travel and sprint, visiting different local communities and “recruiting” contribution was a conscious one – also for the reason of “justly” distribute the load of travel costs in the developer community.

EU-funding allowed a systematic approach of sprinting and the project continues to try to adapt the method of sprinting, evaluating feedback from sprint participants. Tutorials, start up planning meetings as well as daily status meetings evolved, the latest additions to the sprints are closing planning meetings (planning the work between sprints) and work-groups - a version of pair-programming in groups. Documentation such as planning texts as well as sprint reports are now an established procedure in the PyPy community.

It seems that the implementation within the PyPy team of sprinting is slowly conforming to the Scrum standard of sprinting in some aspects, creating an iteration that starts with the end of one sprint, planning the work to be done until the upcoming sprint – creating iterations of ca 6-8 weeks. The project purposefully designed a process of 14 sprints during a 2 year period – creating an iterative process with sprints as the iterated focal point of work, the open and shared forum for work both within the community and the consortium. The main difference from the approach in Scrum is that the version of systematic “sprint-driven development” in PyPy provides an accelerated and collaborative physical practice in an otherwise virtual work style in which shifting locations as well as creating an open forum for participation and contribution are strategies that facilitates cooperation with the community.

Conclusion

We believe that the systematic process of “sprint driven development” is the core agile practice that allowed the Open Source community of PyPy to:

- *work distributed and dispersed – otherwise viewed as the largest threat to a truly agile work style*
- *design a developer driven process that fit within a more formal consortium work style*
- *to balance the continued work of the community and consortium*