Open Source, EU Funding and Agile Methods
22C3 Proceedings

Bea Düring
Change Maker
bea@changemaker.nu

Holger Krekel
merlinux GmbH
hpk@merlinux.de

Abstract

This paper walks through different aspects of agility within the open-source driven PyPy project. Agility played a key role from the beginning. The PyPy project started from some mails between a few people, quickly had a first one-week meeting, a "sprint", from where it evolved into a structure that was able to carry out a research project - and got accepted by the European Union. During the course, two companies were founded. They are now growing and employing key developers.

PyPy's technical development is strongly rooted in open-source contexts and this adds another agility aspect - free communication, co-operation and exchange with other people and projects.

The process of obtaining EU-funding is a continous challenge to the community-rooted PyPy project; how to connect agile open source culture with formal structures, interaction with requirements like planning, budget estimation, work distribution and resource tracking.

After our first "funded" year we are reasonably happy with the balance we strike between organisations and EU funding on the one and the developers driving the technical aspects of the project on the other side.

1 Agility in Technical Development and Organisation

1.1 Agile approaches: sprinting

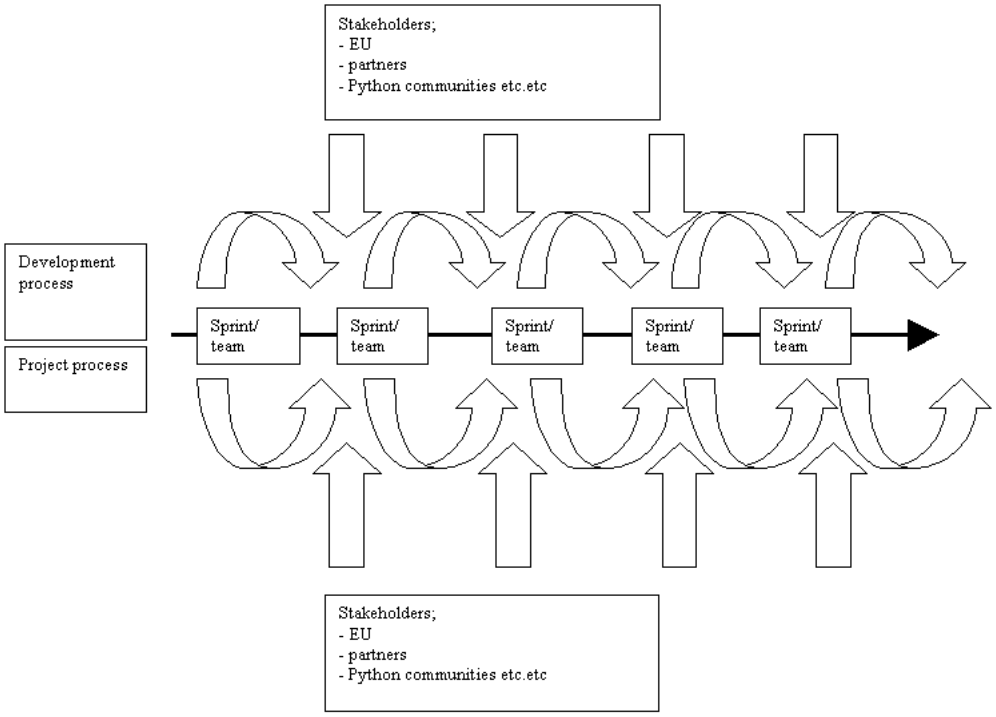
The first bits of PyPy started during a one-week meeting, a "sprint", held at Trillke-Gut in Hildesheim February 2003. The sprint was inspired by practices used by other Python projects such as Zope3. Originally the sprint methodology used in the Python community grew from practices applied by the Zope 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".

Sprinting up to a week became the initial driving factor in developing the code base and the community/people around it. The early PyPy sprints were organised 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.

Sprints are actually not part of the traditional Agile portfolio of techniques, the closest thing to it comes from Scrum who names the 30 days long programming iterations "sprints", covering a certain increment. With the Scrum method, considerable effort is put into performing the sprint planning as well as creating and documenting the "sprint backlog" which is then feedbacked into the "Product backlog". The sprint ends with a "sprint review" - an informal planning session in which the team decides on upcoming work. There are also techniques in which the team looks at ways to improve the development methodology and future sprints.

To our knowledge, open-source projects these days are sprinting for at most a week which reflects the fact that many contributors give their time and even money to gather and work together. This is different from having fully funded people from one company working together.

Why did PyPy choose sprinting as a key technique? It is a method that fits distributed teams well because it gets the team focused around visible challenging goals while working collarobatively (pair-programming, status meetings, discussions etc) as well as acceleratedly (short increments and



tasks, "doing" and testing instead of long startups of planning and requirement gathering). This means that most of the time a sprint is a great way of getting results and getting new people aquainted - a good method for dissemination of knowledge and learning within the team.

1.2 Agile approaches: test-driven development

Test-driven development is a technical cornerstone for programming efficiently together in a distributed team. Seen from the Agile Manifesto perspective it is right up there as one of the key elements since it puts focus on producing working code, rather than diagrams, plans and papers (and then faulty software).

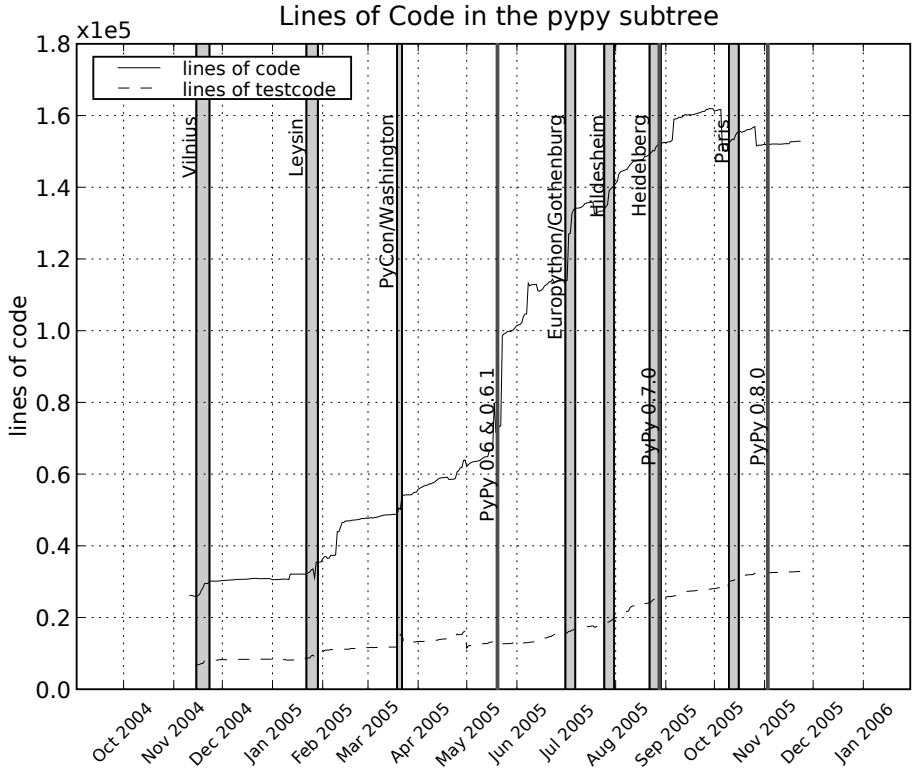
Seen from an Open Source community perspective it is a vitalising strategy - especially in combination with a transparent open process in which anyone interested can participate - if only for just a few days at a sprint. One of the key problems identified by Frederick P. Brooks in the latest version of "The Mythical Man-Month" (unfortunately still very actual today) is estimating correct amount of time for communication and testing/debugging. Automated testing, rather barrier-free communication and strict version tracking helps with that problem, especially in the hands of a team sprinting its way through the Python community - welcoming everyone to participate.

Apart from rewriting a practical programming language within itself, PyPy also evolved a number of development tools useful for writing tests and glueing things together. PyPy's testing tool ("py.test") is used separately and evolves on its own by now.

1.3 Agility: Open Communication and organisation

Another agility aspect relates to transparent and open communication within the project. Only very few (EU-contract related) documents are access restricted, everything else is freely available. There are no hierarchies for commit rights. In fact, the server also hosts a couple of other projects and all projects share commit rights ("Coding Wiki"). Announcing Sprints, Releases and development goals lead to an increasing amount of people subscribing to mailing lists or participating in development.

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.

A rather recent invention is the postings of "This week in PyPy". The text is a summary of what is going on in the lively IRC development #pypy channel - main place of technical coordination.

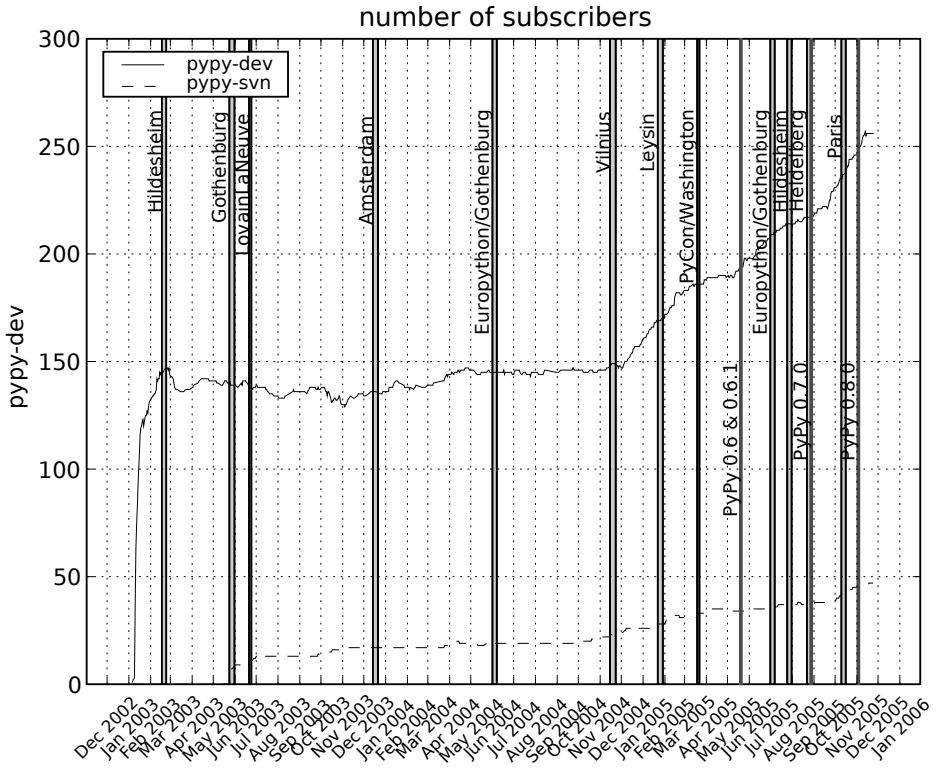
2 How and why EU funding?

Mid 2003 the idea of trying to get EU-funding for the project was born. It became clear that the project had an arbitrarily large scale and that receiving some funding would dramatically increase the pace and seriousness of the project - because funded developers can dedicate more of their time to the project. The involved developers and people stretched outside of the Open Source ecologies to try to gather as much information and contacts as possible in order to answer the question: "Should we go for it?" to which the answer quickly became "Let's see how far we get!".

2.1 Making things fit with EU perspectives

There had been a growing interest from the European Commission, IST division, to look closer at the Open Source world and its achievements. Several funded research projects in the 5th framework programme studied the phenomen (FLOSS-POLS, FLOSS) - its organization, business models and licensings. A few other funded software projects used Open Source in their work as tools (languages and applications). There was no previous experience of an Open Source community based project making a bid for funding.

The areas in the 6th Framework programme (second call) fit very well with the objectives of PyPy. The idea of strengthening the European Software development companies and businesses with supporting an open source language implementation was new but appealing to the EU. However, being an Open Source project wasn´t enough. The challenges and the idea of a flexible, configurable "translator" or "compiler" met the research



targets of the FP6, as well as trying out and documenting the agile methodology being used. It is interesting to note that todays computer industrial language research and development occurs mostly in the US.

In short, we argued that EU funding allowed the project to go from reaching a critical mass and position to continue to evolve from there, and that it would help European Organisations to make some ground.

Acting on this strategy proved to be a more difficult task. The entire proposal and negotiation process took over a year (Autumn 2003 until November 2004). Satisfying the formal requirements, a proper description of planned work, had not previously been part of the development focus and both the EU and the parties involved had to adapt to the situation. Yet, drafting the high-level requirements (in total 14 workpackages and 58 deliverables) was done using the same version-control/open-communication based work style, including evolving the proposal at sprints. Writing the proposal and specifying according objectives on a higher level has proved to be generally useful for clarifying goals on a longer term. It also helps others to understand the project better.

Unfortunately the negotiations with the EU got stuck in organizational limbo and the project is still suffering from the effects of this even today. The goal of funding contributors especially coming to sprints was originally based on a non-profit association. This solution wasn't seen as realistic or feasible by the EU although we think it remains a viable approach for the future. During negotiations, we got to an alternative solution which had a few drawbacks: contributors have to become Contract Partners within the EU-level Consortium (which is by itself not difficult) and can then at least claim travel and accomodation costs when attending sprints.

However, this construction does not allow them to get paid for work time and also has some formal requirements. In practice this leads to current considerations of developers to shift private money between them in order to circumvent the current problems with implementing an agile model within the EU contract framing.

2.2 Seven Organisations / The consortium

The guiding idea for receiving funding is to have organisations in which key developers and other parties are employed. Two companies out of the seven organisations in the initial consortium were funded during the EU negotiation process. what first might have felt as an EU-related obstacle became an opportunity, but with some overhead like legal and organizational responsibilities.

Other adjustments and recruiting companies with previous EU project experiences took place. There also is one company involved quite unrelated to the previous developer work but rather focused on process management and designing learning processes with a background from the Chaospilot school in Aarhus, Denmark. When creating the formal consortium of seven partners new cultures and perspectives were mixed with the strong collaborative Open Source core team, adding new complexities in communication and cooperation. Getting the new "playmates" to adopt the vision, culture and spirit of the original idea and holding true to it during the work on the proposal and negotiation process was a challenge indeed.

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 is not too surprising that the consortium with its member organisation now employs a version-control/review based scheme regarding EU documents reflecting the technical development approaches.

It remains a challenge for all partners of the consortium, universities and companies alike, to connect an ongoing medium-scale open-source project with EU regulations and requirements - not to speak of the fact that companies need to fund 50% of the costs themselves. It is, in fact, too early to judge the overall success of our approaches although we are confident that things work out reasonably well.

2.3 challenges: balancing community interests with EU requirements

The nature of sprints changed when EU funding started. The need to meet milestones of promised *deliverables* and the goal to keep an open sprint process, still welcoming newcomers into the world of Pypy, made the sprints longer (at least 7 days with a break day in the middle) but also changed the nature of the sprints. The team started to distuingish between sprints open for all to attend, without any prior PyPy experience, and sprints requiring earlier PyPy involvement. 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.

Some other effects of sprinting within the EU-structure is that the sprint becomes a forum for non-development work - coordinating and tracking the project. The challenge here is not affecting the main work and "disturbing" visiting developers with EU-related work. It could also be argued that the prolonged sprints could possibly make it more difficult for non consortium members to attend the full time, disturbing other engagements etc.

The project continues to try to adapt the method of sprinting, evaluating feedback from sprint participants. Maybe the implementation within the PyPy team is slowly conforming to the Scrum standard of sprinting, but not as a conscious effort?

2.4 Managing diversities: agile business

For a diverse group of organisations and people, agility is helpful at various levels: you cannot make all-encompassing plans and hope to statically follow them and succeed. New developments, twists and opportunities evolve all the time.

Our experience with evolving PyPy from a loose Open Source project to a partially funded EU research project shows the following:

- what first seemed like too diverse interests and views, impossible to tailor into a single project, was instead a fruitful mix of diversities. The challenge is to manage these diversities and channel them into constructive team efforts. Aiming for homogenity is the real threat.

- what first seemed like unbeatable odds and too big obstacles even turned sometimes into new possibilities. The challenge is to maintain an atmosphere in which a team can act on those and within short timeframes of opportunities. Change is inevitable - how you handle it is the real challenge.

- there are many other projects and organisations who are heading in similar directions of trying to connect and evolve agile open source strategies with business matters. Emerging models for developers distributed between different countries allows people with special interests to effectively work together and learn from each other.

Concluding - the cumulative effects of an agile, open and dynamic team process combined with a market and curious first adopters facilitates agile business. A positive result is that a lot of people within the PyPy context found enjoyable jobs and there now already is evolving commercial interest despite the still early stages of the project - mostly from US companies though ... why european companies, especially larger ones, appear to prefer taking rather naive views on agile open-source development ("great, it's cheaper, no license fees!") is another interesting topic.