IST FP6-004779

PYPY

Researching a Highly Flexible and Modular Language Platform and Implementing it by Leveraging the Open Source Python Lanugage and Community

STREP

IST Priority 2

D14.1: Report about Milestone/Phase 1

Due date of deliverable: September 2005

Actual Submission date: 23th December 2005

Start date of Project: 1st December 2005 **Duration: 2 years**

Lead Contractor of this WP: Change Maker

Authors: Beatrice Düring (Change Maker), Holger Krekel (merlinux)

Revision: final

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)

Dissemination Level: PU (Public)

Abstract

This document describes the results of work in phase 1 of the PyPy project. The results are summarized on a technical level, dissemination and community level. We also present the organizational context of the project (the consortium). In the conclusion we state some of the effects of the work done and we also present briefly work planned for phase 2.

Contents

1 Executive Summary

The milestone of phase 1 was the successful completion of a self-contained novel Python Implementation, mainly through the 0.6 (20th of May 2005) and the 0.7 (28th of August 2005) public PyPy releases.

In this report about the first PyPy milestone we will describe and summarize the main development results of the first phase, and we will include an overview of the internal PyPy Work Phases.

We will also give an overview of the dissemination efforts that were completed during phase 1 explaining how we incrementally raised community support and interest.

We will then briefly conclude the results of phase 1 and discuss Community Aspects and we finally give an outlook for the planned work in phase 2.

In order to provide organizational context for the work achieved in phase 1 we will also provide a brief description of the consortium structure and related activities.

2 Introduction

2.1 Purpose of this Document

This document provides a summary for both a domain specific and a non-domain specific readership. It presents a summary of work done during phase 1 of the EU/PyPy project and explains related project methodologies and dissemination results.

2.2 Scope of this Document

This document describes the achievements of phase 1 of the PyPy project, the main technical result and the also related dissemination efforts and their effects. It does not contain financial or related contractual information.

2.3 Related Documents

This document has been prepared in conjunction with other tasks in work packages 1, 4, 5 and 14. It is helpful to read this document in conjunction with the deliverables of work packages 4 and 5, in particular:

- Annex 1 Description of work
- D04.2 Complete Python Implementation on top of CPython
- D05.2 A compiled, self-contained version of PyPy
- D05.1 Compiling dynamic language implementations

3 Overview on Internal PyPy Work Phases

The EU PyPy project started on the 1st of December 2004 and has a project time-frame of two years. The project is internally structured in three phases. The first one lasted until 31st August 2005.

The three phases have the following focus:

- Phase 1: Building a Novel Language Research Tool. PyPy itself is developed, using our novel and flexible approach of Object Spaces both for implementing the Python language and the translation tool-chain.
- Phase 2: High Performance. The phase 1 code base is used as a research/integration platform, targeting especially performance related research and work.
- Phase 3: Validation & Flexibility. Specific applications can be implemented and disseminated on top of the results of the other phases.

The achievements of phase 1 are the prerequisites for the other two phases. The successfully reached goals of phase 1 provide a research platform, comprising an implementation and executable specification of the Python language and a translation tool-chain to produce custom self-contained versions of Python.

4 Summary of Work Achieved During Phase 1: Development

Our goal for the first phase was to target a novel architecture for research of a VHLL language. At the same time we aimed to be compatible with the language specification of Python. Phase 1 covers the following major technical achievements:

- A bytecode compiler which inputs Python source code and outputs an internal intermediate representation, the bytecode.
- A bytecode interpreter which interprets bytecodes and manages the supporting internal structures (frames, exception, tracebacks...). It considers objects as black boxes and delegates all individual operations on them to a library of built-in types, the Object Space.
- An Object Space which captures the semantics of the various types and objects of the language.
- A translation tool-chain based on abstract interpretation and performing type-inference that is able to translate the PyPy Python implementation (bytecode compiler and interpreter plus the Object Space) to low-level targets.

The main technical achievements during phase 1 resulted in 2 public releases, the 0.6 (and 0.6.1, which corrected a trivial bug in 0.6) on the 20th of May 2005 and the 0.7 release on 28th of August 2005.

The 0.6 public PyPy release presented a successful implementation of the following aspects of PyPy core functionality:

4.1 Bytecode Interpreter

An implementation of the bytecode interpreter conforming to the complete Python language specification. This measurable goal was targeted through compliance tests - see test statistics below. The compliance tests were based on CPython 2.3.4 release and some modifications to this actual release (regarding the standard library implementations) were done during phase 1 by the PyPy developers.

4.2 Core Test Results as of 20th of May 2005

Total test compliance	92.08%
Test modules passed completely	88.15%
Test modules (partially) failed	11.85%
Test modules timeout	0.00%

The PyPy interpreter succeeded in implementing the novel and flexible approach of cleanly abstracting out operations on objects to an Object Space.

The Standard Python "builtin" library needed to be ported. The PyPy team ported some 80% of the built in types as part of the 0.6. release:

- *types* (there are some 100 of them)
- *built-in functions*
- *built-in modules*
- *other built-in objects*, e.g. exception classes.

It is also worth mentioning that 90% of the core modules of the standard library are supported by the standard interpreter.

4.3 Translation and Self-contained PyPy

The 0.7 release leveraged the PyPy Python implementation and managed to translate it to low-level machine code by producing C code to be compiled via a C-compiler.

After performing abstract interpretation on our own interpreter, we implemented a type inference engine ("the annotator") which is able to perform whole-program type-inference on a reasonably powerful subset of the Python language (we call this subset "RPython"). The toolchain is able to analyse any RPython-compliant python program, capture its control flow and behaviour, annotate it with types, translate it to low-level code and compile it. By applying these techniques to our own Python interpreter we were able to produce a first self-contained Python version - no longer running on top of CPython (a detailed explanation of our translation approach is in D05.1 *Compiling dynamic language implementations*).

The PyPy team also managed to begin experimenting on different back-ends although in this matter no finalized results were part of the public 0.7 release. We succeeded in providing a backend for the Low Level Virtual Machine (LLVM) and also added support for different memory management strategies and threading models.

These releases and documentation can be downloaded from:

http://codespeak.net/download/pypy/pypy-0.6.0.tar.bz2,
http://codespeak.net/download/pypy/pypy-0.6.1.tar.bz2 and
http://codespeak.net/download/pypy/pypy-0.7.0.tar.bz2.

5 Summary of Work Achieved During Phase 1: Dissemination

The different target groups for dissemination of the PyPy project (technical aspects as well as methodology aspects) are:

- the PyPy community (around 300 developers, less than 5% being funded by the Commission)
- the Python community (perhaps 100,000 developers but no proven number exists)
- other language communities (Java, C, C++, Smalltalk, Perl etc)
- other Open-Source Software (OSS) projects
- other EU funded projects, researching OSS or implementing OSS
- companies (working with implementations connected to the PyPy findings)
- research bodies and government bodies in the partner countries

During phase 1 the main strategy for dissemination within the PyPy community as well as the Python community has been to organize sprints (our agile approach of one week long programming sessions) in conjunction with the major community conferences EuroPython and PyCon (USA).

During sprints the goal has been to create in-depth understanding of the development work. The agile methodology has been applied and we invited contributors on several occasions.

Six sprints have been organized during the first nine months to achieve these goals successfully:

- Leysin, Switzerland,
- PyCon, Washington DC, USA,
- before EuroPython, Göteborg, Sweden,
- after EuroPython, also in Göteborg, Sweden,
- Hildesheim, Germany and
- Heidelberg Germany.

It is worth mentioning that the public releases of 0.6 and 0.7 were organised and launched during two of these sprints.

Other key strategies for dissemination and reaching out to these target groups have been achieved by talks and participation in conferences. During phase 1 we attended the following conferences:

- CCC conference Berlin (December 2004) (“Hacking EU-funding for a FOSS project”)

- Calibre, 3rd Calibre workshop, Paris (March 2005) ("py.test and the pylib")
- PyCon, Washington D.C, (March 2005) ("py.test and the pylib", "PyPy and Type Inference")
- ACCU, Oxford, (April 2005) ("Sprints - a new agile development methodology")
- EuroPython, Gothenburg, (June 2005) ("Sprinting the PyPy way", "PyPy as a compiler", "py.test - making testing fun")

All talks have been published on the PyPy project development website:

http://codespeak.net/pypy

Documentation of the work done in phase 1 was created parallel to the development work and was also continously published on the project website (covering such topics as translation, architecture, sprint methodology, a "getting started" tutorial, dynamic-language translation, low-level encapsulation and the bytecode interpreter). Tutorials have also been created, primarily being used during the sprints, and these are also available on the project website.

The primary communication channels have been mailing lists and IRC channels on the freenode network. They create ample opportunity to get in contact with the project and get help and have questions answered:

- pypy-funding@codespeak.net (consortium coordination)
- pypy-dev@codespeak.net (pypy developer community)
- pypy-sprint@codespeak.net (sprint organisation)
- #pypy (IRC channel for day-to-day development work)
- #pypy-funding (IRC channel for consortium work)

In addition to the documentation on our website there have been several articles written on external websites - leading to a continously increasing number of subscribers and contributions.

A summary webpage was created on http://pypy.org with a brief introduction to the projects main objectives and the consortium partners (with contact information) and linking to the main development site.

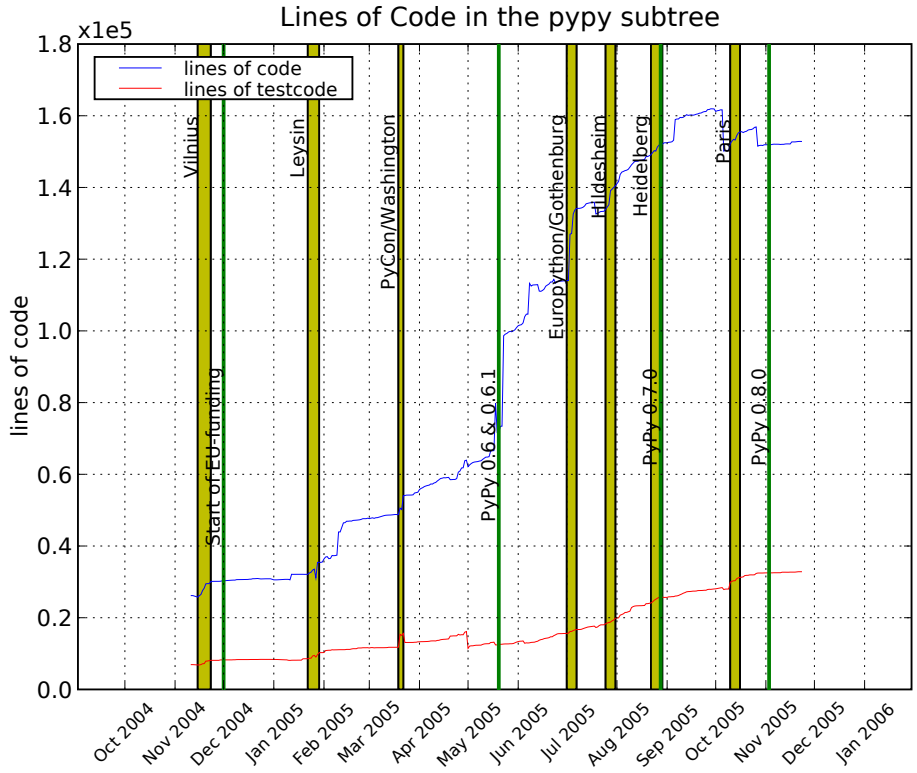
6 Community Aspects and Conclusion Phase 1

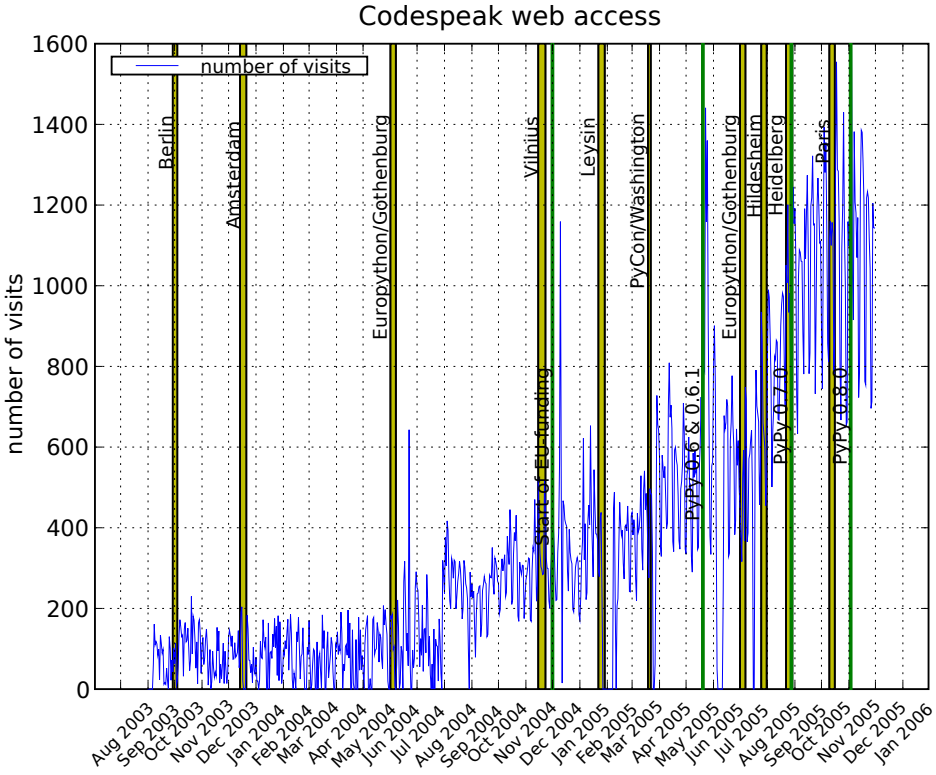
The public PyPy releases marked our fulfillment of milestone 1 in the EU project. Within the PyPy and the Python community the reactions have been very positive and considerably raised interest. The releases made it possible for people to test and experiment in a direct way - because we provided detailed tutorials on how to get started.

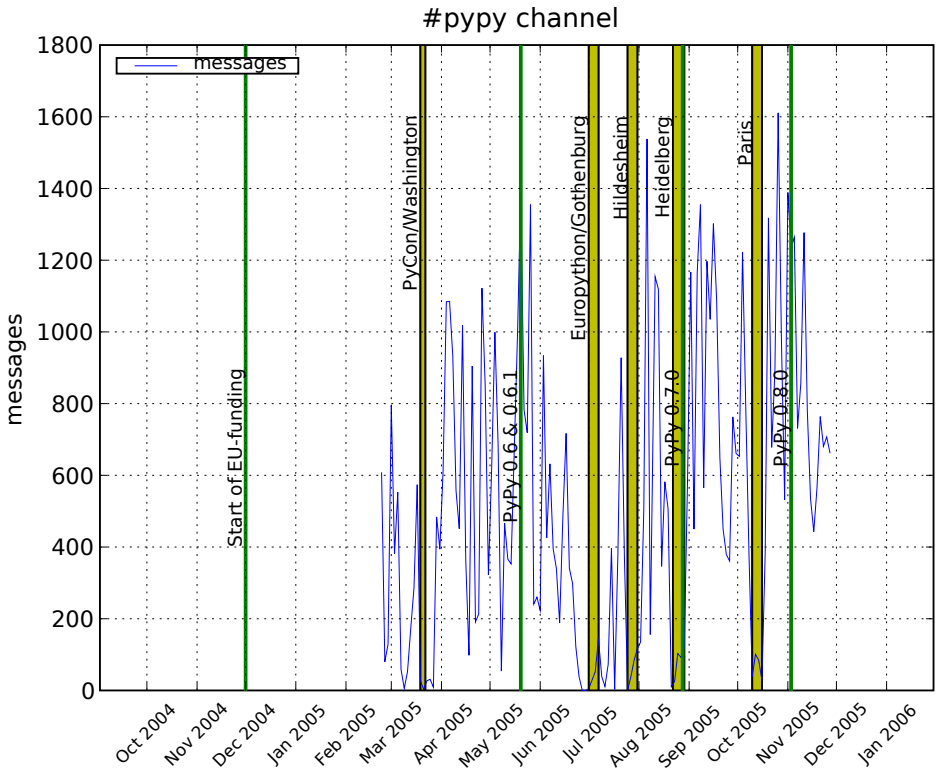
This created synergies as people identified connections to their own implementation projects (both open source and commercial) and a larger group of developers started to test the code, and through peer review also contributing in various ways. Discussions on various possible back-end supports (Javascript, Smalltalk/Squeak, Java etc) show the potential of the

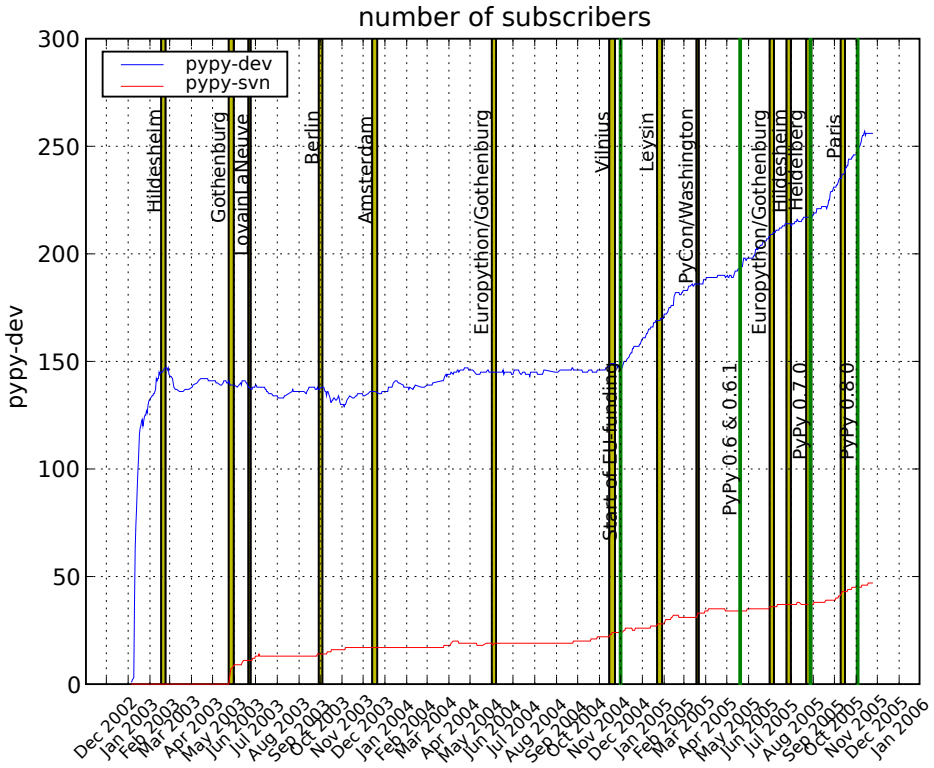
flexible and modular approach being used in the PyPy implementation as well as it shows the interest in these areas.

The development of this community growth connected to sprints and releases is very visible in the following metrics:




Codespeak web access
number of visits
number of visits
Berlin
Amsterdam
Europython/Gothenburg
Vilnius
Start of EU-funding
Leysin
PyCon/Washington
PyPy 0.6 & 0.6.1
Europython/Gothenburg
Hildesheim
Heidelberg
PyPy 0.7.0
Paris
PyPy 0.8.0
0
200
400
600
800
1000
1200
1400
1600
Aug 2003
Sep 2003
Oct 2003
Nov 2003
Dec 2003
Jan 2004
Feb 2004
Mar 2004
Apr 2004
May 2004
Jun 2004
Jul 2004
Aug 2004
Sep 2004
Oct 2004
Nov 2004
Dec 2004
Jan 2005
Feb 2005
Mar 2005
Apr 2005
May 2005
Jun 2005
Jul 2005
Aug 2005
Sep 2005
Oct 2005
Nov 2005
Dec 2005
Jan 2006


#pypy channel
messages
messages
Start of EU-funding
PyCon/Washington
PyPy 0.6 & 0.6.1
Europython/Gothenburg
Hildesheim
Heidelberg
PyPy 0.7.0
Paris
PyPy 0.8.0
0
200
400
600
800
1000
1200
1400
1600
1800
Oct 2004
Nov 2004
Dec 2004
Jan 2005
Feb 2005
Mar 2005
Apr 2005
May 2005
Jun 2005
Jul 2005
Aug 2005
Sep 2005
Oct 2005
Nov 2005
Dec 2005
Jan 2006



First of all a clear increase in interest can be identified around the start of the EU funding. In second place it is obvious, that each sprint and release increased interest and activity within and around the project. We estimate that at the end of phase 1 approximately 300-500 people follow the project closely.

The synergies created from our pervasive open-source dissemination techniques are significant:

- Increased contribution from the community; a result of both the challenging goals of the project on a technical level and the agile approach of sprinting as a strategy of integrating with the community.
- Specific contacts being structured into possible new partners in the consortium; a result of our dissemination efforts - mainly sprints and conferences but also the open and transparent culture of the project.
- Specific contacts with coordination action project Calibre, which have already resulted in talks planned at international conferences covering research on OSS projects; a result of dissemination efforts such as participating in conferences targeting groups "outside" of the Python community and networking. Organizations and projects working specifically with researching OSS learned about PyPy and the unique hybrid nature of the project (agile, OSS and distributed) which in turn created opportunities for cooperation and increased dissemination in phase 2.

7 Work Planned for Phase 2

As mentioned above, phase 2 targets high performance. Dynamic translation and core optimizations are now the main focus. We also start to investigate aspect-oriented programming and logic programming and the implementation effects of these on PyPy.

During phase 2 we plan to organize another 5 sprints, both in Europe and the US. We will continue and also increase the dissemination efforts through an increased amount of talks as well as papers and articles, for example:

- 2nd CALIBRE International Conference (http://www.calibre.ie)
- Project Management Institute, Swedish Chapter (http://www.http://www.pmi-se.org/)
- 22C3 congress (http://www.ccc.de/calendar/2005/22c3?language=en)
- PyCon (http://us.pycon.org/TX2006/HomePage)
- ACCU (http://www.accu.org)

On a consortium level we will strive to succeed in the strategy of funding contribution for sprint attendance as well as inviting new partners, exploiting the synergies created in phase 1 and actively seeking for new possibilities to collaborate with the Python communities. Moreover, we plan to increase efforts in collaborating with other EU projects (Calibre, Calibration) as well as starting to target larger companies for domain specific workshops.

8 Summary of Work Achieved During Phase 1: Consortium

The partners contributing to the phase 1 work were:

- DFKI Saarbrücken, Germany
- AB Strakt, Göteborg, Sweden
- Logilab, Paris, France
- Change Maker, Göteborg, Sweden
- merlinux GmbH, Hildesheim, Germany
- tismerysoft GmbH, Berlin, Germany
- Heinrich-Heine-Universität Düsseldorf, Düsseldorf, Germany (HHU)

The consortium of partners were organized with the following specific roles:

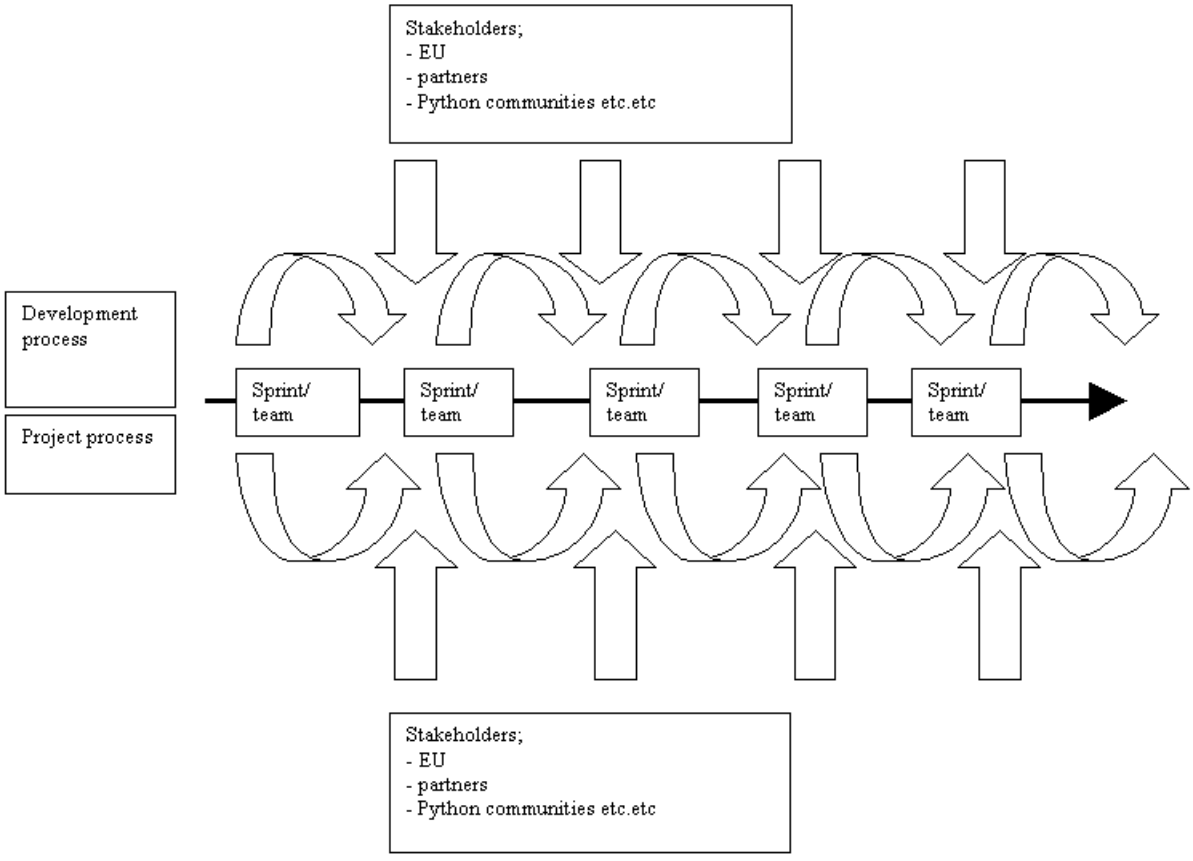
- Project coordinator: Stephan Busemann and Alastair Burt (DFKI)
- Project manager: Jacob Hallén (AB Strakt)
- Assistant project manager: Beatrice Düring (Change Maker)
- Technical director: Holger Krekel (merlinux GmbH)

- Technical board: Holger Krekel (merlinux GmbH), Armin Rigo (HHU), Christian Tismer (tismerysoft GmbH), Samuele Pedroni (AB Strakt)

The main coordination of work on a consortium level during phase 1 was done during consortium meetings. Since April 2005 these meetings became a monthly event via IRC. Work in between consortium meetings was done within the management team (coordinator, project manager and assistant project manager) and the Technical Board.

In July the technical director implemented pypy-sync meetings with the PyPy developers in order to coordinate work on a weekly basis. This fact allows both the partner developers as well as the non funded contributors to influence the work plans.

These different levels of the project organization (management team, technical board, developers - also being partners and consortium members) worked together in an open and agile approach with the systematic structure of sprints (one week long programming sessions). During phase 1 six sprints were organised.



The consortium also prepared amendments during phase 1 in order to be able to partially fund contribution from interested developers (Eric van Riet Paap, Richard Emslie, Laura Creighton) as Physical Person partners in the consortium. The consortium also invited Impara GmbH to join as a full partner. These amendments are still being processed at the Commission at the end of reporting period 1.

It has been an active strategy during phase 1 to try to support both developers with partial funding for contributions during sprints as well as identifying new partners and new possibilities of cooperation with other projects and organizations.

9 Glossary of Abbreviations

The following abbreviations may be used within this document:

9.1 Technical Abbreviations:

AST	Abstract Syntax Tree
CPython	The standard Python interpreter written in C. Generally known as "Python". Available from www.python.org.
codespeak	The name of the machine where the PyPy project is hosted.
docutils	The Python documentation utilities.
GenC backend	The backend for the PyPy translation toolsuite that generates C code.
GenLLVM backend	The backend for the PyPy translation toolsuite that generates LLVM code.
Graphviz	Graph visualisation software from AT&T.
Jython	A version of Python written in Java.
LLVM	Low Level Virtual Machine - a compiler infrastructure available from University of Illinois at Urbana-Champaign
LOC	Lines of code.
Object Space	A library providing objects and operations between them, available to the bytecode interpreter via a well-defined API.
Pygame	A Python extension library that wraps the Simple Directmedia Library - a cross-platform multimedia library designed to provide fast access to the graphics framebuffer and audio device.
ReST	reStructuredText, the plaintext markup system used by docutils.
RPython	Restricted Python; a less dynamic subset of Python in which PyPy is written.
Standard Interpreter	The subsystem of PyPy which implements the Python language. It is divided in two components: the bytecode interpreter, and the standard object space.
Standard Object Space	An object space which implements creation, access and modification of regular Python application level objects.

9.2 Partner Acronyms:

DFKI	Deutsches Forschungszentrum für künstliche Intelligenz
HHU	Heinrich Heine Universität Düsseldorf
Strakt	AB Strakt
Logilab	Logilab
CM	Change Maker
mer	merlinux GmbH
tis	Tismerysoft GmbH