Open-Source Development Processes and Tools

Cornelia Boldyreff, Janet Lavery, David Nutter, and Stephen Rank,
Department of Computer Science,
University of Durham
Stephen.Rank@durham.ac.uk

Abstract

Open-source software projects rarely have high-quality process support. Any tools which are used to control workflow in a project are usually fixed to a particular process, which is often not made explicit.

In order to improve the support for process modelling, monitoring, and control, it is necessary to use tools which allow the creation and manipulation of process models. This paper describes the GENESIS platform, an open-source lightweight process-neutral toolkit which supports distributed collaboration over the internet for software engineering. The capabilities provided by the platform make it suitable for adoption by open-source projects. Many open-source projects have little or no automated process support. The GENESIS project offers these capabilities in a form which is ideal for adoption by such projects.

1. Introduction

This position paper describes aspects of some open-source development processes, and identifies ways in which the GENESIS platform could be used to assist open-source development.

The GENESIS platform is an EU-funded open-source process-aware toolkit to support distributed software development. Although it is initially targeted at development within commercial organisations, it can be useful to open-source software development. The GENESIS platform is described in section 4.

2. Open-Source Development: Tool Support

There are two kinds of tool used in software development: those which support the product (editors, compilers, build tools, *etc.*) and those which support the process (such as configuration management tools). In open-source software development, most tool use is in product development, or at the less-sophisticated end of the process-support spectrum (*e.g.*, CVS). Process support is usually limited to informal use of communication tools (such as email and IRC clients). There is rarely any maintenance of relationships between informal knowledge (which is often the largest and most useful body of knowledge about a project) and the components which make up the software. This leads to inefficiency, as developers (or others) who wish to discover information about particular components have to search mailing list archives (for example) in order to discover information.

3. Open-Source Development Processes

Much open-source software development is carried out by small groups with a single leader [2]. In larger projects, a `core group' or official organisation (such as the Apache Software Foundation) usually takes the place of the leader, performing the same tasks [2].

Open-source development usually has very little formalised process modelling or support. Some projects use issue-tracking software to provide limited support for their evolution processes, but very few have any explicit model of their design or development activities. There are many reasons for this, including the perceived overhead of introducing and enforcing formal models of the software process, and the perceived inflexibility of such models [3].

Coordination and discussion in open-source projects usually takes place using communication tools such as email or IRC [6]. When logs of messages are kept, they are usually kept in separate repositories, and are not organised in a way that makes them easily available to the user. For example, mailing list archives are usually available over the web, but with no links from software components to relevant postings. Open-source software is often lacking in design documentation. In this case, mailing list archives are the only record of design decisions. If these archives were readily related to the components which they describe (and, perhaps more

usefully, vice versa), a developer wishing to understand the design of a component could more easily discover the rationale for decisions.

There is very little formalised software process support used in open-source software projects. Tools such as Bugzilla provide some support for the process model implicitly contained within them [4], but do not provide capabilities to modify the process model, or to use a different model entirely. Neither do they provide a mechanism for supporting informal documentation.

All projects operate under some form of `government'. For example, the Debian project has a project leader, elected by the developers, while many projects have a self-selected leader (usually the person who initiated the project). Leaders have some form of control of the development process, including delegation (to volunteers), the ability to make rulings on (for example) design decisions, and the use of a `casting vote' in disputes. There is frequently no tool support for this process, nor any way of monitoring the delegation of tasks.

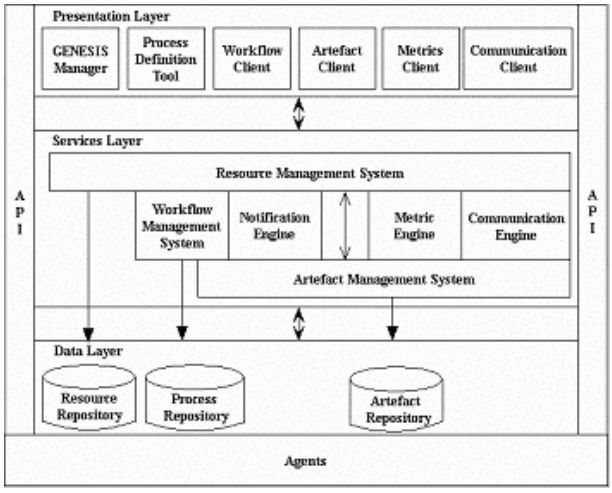

Figure 1: The GENESIS Platform Architecture

4. The GENESIS Platform

The GENESIS project is creating an open-source process-aware platform to support distributed cooperative software development [5,1]. The architecture of the platform is shown in figure 1. The GENESIS platform's process-modelling component GOSPEL allows project managers to model their organisation's existing processes. In the context of an open-source project, the `project manager' is the leader or core group. Within the GENESIS platform, it is possible to assign work units (as small as fixing a simple bug, or as large as designing the entire system) to individuals or to groups of people. The resource management system is used by GOSPEL to manage the allocation of people to workflow items.

Coordination between tasks is at the level of artefacts: each task in a process model has a set of input artefacts (which must be supplied before the task can be begun) and a set of output artefacts (which are supplied to later tasks). Artefacts can consist of any item of data which can be represented in a file, and are stored in the repository, known as OSCAR, whose architecture is shown in figure 2. OSCAR provides superior SCM capabilities, allowing the maintenance of relationships between artefacts, which provides for traceability in the software development process.

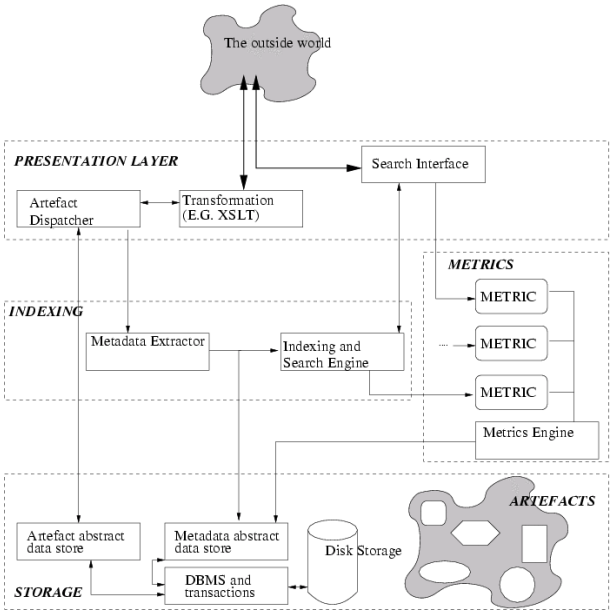

Figure 2: The Architecture of OSCAR

The GENESIS platform is distributed: each site (which can have as few as one developer) has an instantiation of the process manager and the artefact manager, which may be simply stubs to allow remote access to another site's components. In this way, the distribution of GENESIS is flexible, and can be configured to suit the nature and size of each site. This distribution is shown in figure 3.

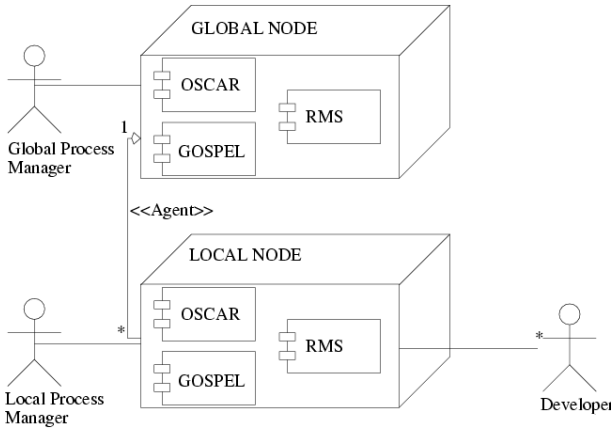

Figure 3: GENESIS Site Deployment

As the GENESIS platform is process-aware, it can be adapted to support whatever process is in use by a project, rather than enforcing a process model in place of an already existing method. This allows for low-impact introduction of the system into a project. The component nature of the platform allows progressive introduction of the platform: OSCAR can be introduced initially, in place of the configuration management system (such as CVS, from which data can be extracted) which is already in use. The use of OSCAR can later be supplemented with the other components, in order to use the facilities provided by them.

The use of the GENESIS platform provides a level of visibility of the software process that is not available to projects which do not use a process-enactment tool. This is useful both in planning and in monitoring project progress. This provision of visibility allows the project leaders greater control over and knowledge of the state of the project. The project manager (or `key developer', or `maintainer') can use the process management tool to record delegation. It is possible for each process element (a step in the workflow) to be delegated to a team, with one member of the team holding overall responsibility (only the leader can make the final commit of the output artefacts). This allows sophisticated multi-level decomposition of the workflow.

The repository is designed to store both formal software artefacts (such as code, design documents, test cases, process models, *etc.*) and informal material. The informal material can take the form of annotations to artefacts (added by the user) or can be mailing list archives, IRC logs, and so on. Each item can be linked to the artefact(s) to which it is related, in order to provide connections between formal and informal artefacts. This linking provides a capacity for informal documentation of (for example) software components. When formal documentation is missing from open-source projects, informal documentation is the next-best alternative. Currently there are no links between the artefacts and the informal documentation describing them (mailing list archives have to be searched, for example), which results in a loss of efficiency. There is a danger of `information overload'; if there are many discussions relating to an artefact, there will be too much material to read. If this occurs, a filtering mechanism can be applied, or a set of intermediate artefacts used to classify the informal artefacts. For an example of this concept, see figure 4.

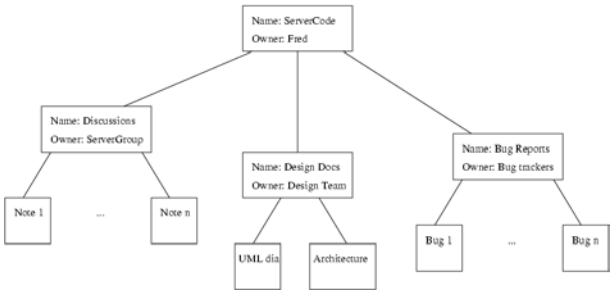

Figure 4: Using intermediate artefacts to classify annotations

Currently, the GENESIS platform is under development, nearing the release of the first stable version. Once this version has been released, it will be adopted by the project, and will be used in pilot studies in the two participating companies. The results of these studies, and the experiences of the project members, will be used to evaluate the platform and to motivate future development.

5 Conclusion

This paper has outlined some reasons for the lack of explicit process support in many open-source projects, and has shown how the GENESIS platform can be introduced into an open-source project, and the benefits which this will bring to such a project. The GENESIS platform is intended to be simple to install and impose as little overhead on the project as possible. Because of this, it will be suitable for adoption by projects which already have artefact repositories but no process support.

Introducing the components of the GENESIS platform into an open-source project will, we claim, increase the productivity of the project by decreasing the management overheads of the project leaders, and by allowing the creation of a richer representation of the artefacts created

and used by the project members. Once the project has reached a stable and usable condition, it will be interesting to carry out case studies of both open-source and commercial development using the GENESIS platform. An initial case-study will be provided by hosting the development of the GENESIS project using the GENESIS platform.

Bibliography

[1] C. Boldyreff, D. Nutter, and S. Rank. "Active artefact management for distributed software engineering". In *Workshop on Cooperative Supports for Distributed Software Engineering Processes, Proceedings of COMPSAC2002*, pages 1081-1086. IEEE, August 2002.

[2] J. Feller and B. Fitzgerald. *Understanding Open Source Software Development*. Pearson Education, 2002.

[3] E. S. Raymond. *The Cathedral and the Bazaar: Musing on Linux and Open Source by an Accidental Revolutionary*.
O'Reilly and Associates, 1999.

[4] C. R. Reis and R. P. de Mattos Fortes. "An overview of the software engineering process and tools in the Mozilla project". In *Proceedings of the Open Source Software Development Workshop*, pages 155-175, Newcastle, UK, Feb. 2002.

[5] P. Ritrovato. "Generalised enviroment for process management in cooperative software engineering". In *Workshop on Cooperative Supports for Distributed Software Engineerng Processes, Proceedings of COMPSAC2002*, pages 1049-1053. IEEE, August 2002.

[6] J. Sandred. *Managing Open Source Projects*. Wiley, 2001.