Open-Source Artefact Management

Cornelia Boldyreff

David Nutter

Stephen Rank

{Cornelia.Boldyreff,David.Nutter,Stephen.Rank}@durham.ac.uk

Research Institute for Software Evolution,

Department of Computer Science,

University of Durham, UK

15th March 2002

Abstract:

This paper presents the GENESIS project, which aims to develop an open-source, light-weight, process-aware (and process-neutral) workflow management system. In particular OSCAR, the artefact repository is discussed. The requirements of a system for artefact management and storage are described, and the concept of active artefacts is explained. The software engineering methods which will be used in the project are described, and some examples of the open-source tools which may be used are described.

Introduction

This paper introduces the GENESIS (GEneralised eNvironment for procEss management in cooperatIve Software engineering) project, an EU project funded under the Information Society Technologies programme. GENESIS aims to produce a light-weight process-aware, and process-neutral, workflow management system to support distributed software engineering. In addition to Durham, work on the project is being carried out at CRMPA, Sannio, LogicDIS, SchlumbergerSema, and MoMA. In this paper, the OSCAR (Open-Source Component and Artifact Repository) component, which will be developed at Durham, is examined. OSCAR is a repository for managing active artefacts, and is designed to be as non-intrusive as possible, in order to make minimal impact on the processes already in place in organisations which adopt the GENESIS products.

The GENESIS project is at an early stage, so this paper concentrates on the requirements and aims of the tools. The tools are to be released as open-source

products at the end of the first year of the project, after which they will be developed in the open.

Section 2 briefly describes the aims of the GENESIS project, as this is the context in which OSCAR will be used. Then, in section 3, OSCAR is introduced. The concepts relating to artefacts are introduced, and a simple example of the use of the repository is given. Section 4 outlines the process by which the project will proceed, and section 6 indicates the directions that the work will take in the near future.

GENESIS

The GENESIS project addresses some of the problems of distributed software engineering. When developers on a particular project are distributed across (or even within) organisations, their software processes may not match. Tools which aim to support such teams of software engineers must support each process that is being used, as well as the overall process.

The project aims to provide workflow and process support for distributed teams of software developers, potentially working across organisational boundaries. As the tools must support software engineers and managers working in different organisations, they must be process-neutral, in order to be applicable in projects that involve many organisations, each potentially with different processes. In addition, the tools must also be process-aware, to enable them to support each style of process which is involved. In this kind of `virtual corporation', modelling of the software process is problematic due to the `inherent uncertainty' of the software development process [1]. This project aims to respect the processes used in each individual organisation while at the same time supporting the overall process enacted by the virtual team of software engineers.

The tools which will be created as part of the GENESIS project are aimed at distributed development. This means that they will have to be able to manage potentially distributed repositories of artefacts. Distributing the repository will also

potentially have dependability benefits, allowing most of the system to continue to function even if part of it fails (*e.g.*, if a node loses its connection to the network).

As part of the GENESIS tool set, a repository will be created. This will be known as OSCAR, and is described in the next section.

OSCAR

In this section, the kinds of artefact that OSCAR [2] will be used to manage are described, as is the form of OSCAR repository itself.

Artefacts

An artefact is any entity which can be stored in electronic form. The most obvious kind of artefact for a software repository is source code, but this is only one of many kinds of artefacts which OSCAR will be able to manage. As the GENESIS systems involves process and workflow management, process models and instances of processes will also be stored by OSCAR, as will documentation (such as requirements documents, architectural descriptions, *etc.*), and annotations and other informal notes. The existence of a particular artefact type for this kind of informal notes is an attempt to allow the capture and representation of rationale for design decisions, exploration of alternative choices which are later dismisses, and so on. This kind of informal notes form a category of implicit organisational knowledge which is often lost when a project is completed [3].

Artefacts will have a type hierarchy, similar to the object-oriented style. Each artefacts type will inherit from a `parent' artefact types (with the basic `Artefact' type as the eventual ancestor of all types). For example, there will be a type of artefact used to represent source code, a type for representing process models, and another used to represent software tools.

Each artefact will have associated meta-data, which will describe the artefact to allow both the repository and the user to manipulate and understand the artefact. The slots in meta-data descriptions will vary depending on the type of artefact in

question. For example, a source code artefact will have a slot for programming language, a software tool will have a slot to enable the description of the platform(s) on which it will run, and an element of a process model will have a slot for the roles involved. Each artefact will have more slots than this, of course. One of the important uses of meta data will be in revision and configuration management; artefacts' history and changes will be stored as meta-data. Meta-data will enable dependencies between artefacts to be described, in a similar way to configuration-management systems such as the Debian package management system. Each artefact's meta-data will show which artefacts it depends on, recommends, suggests, *etc.* This allows consistency to be maintained when a user retrieves an artefact from the repository; if the user doesn't already have a required artefact, this can be retrieved at the same time as the requested artefact.

The Repository

An outline of the proposed architecture of the OSCAR repository is shown in figure 1 [2]. This shows the main components of the repository, and their interactions. The storage layer manages access to the physical and logical storage of artefacts and meta-data. The repository may be distributed (as the GENESIS tools are intended to support distributed software engineering), and the components in the storage layer will be responsible for managing network interactions as well as local storage transactions.

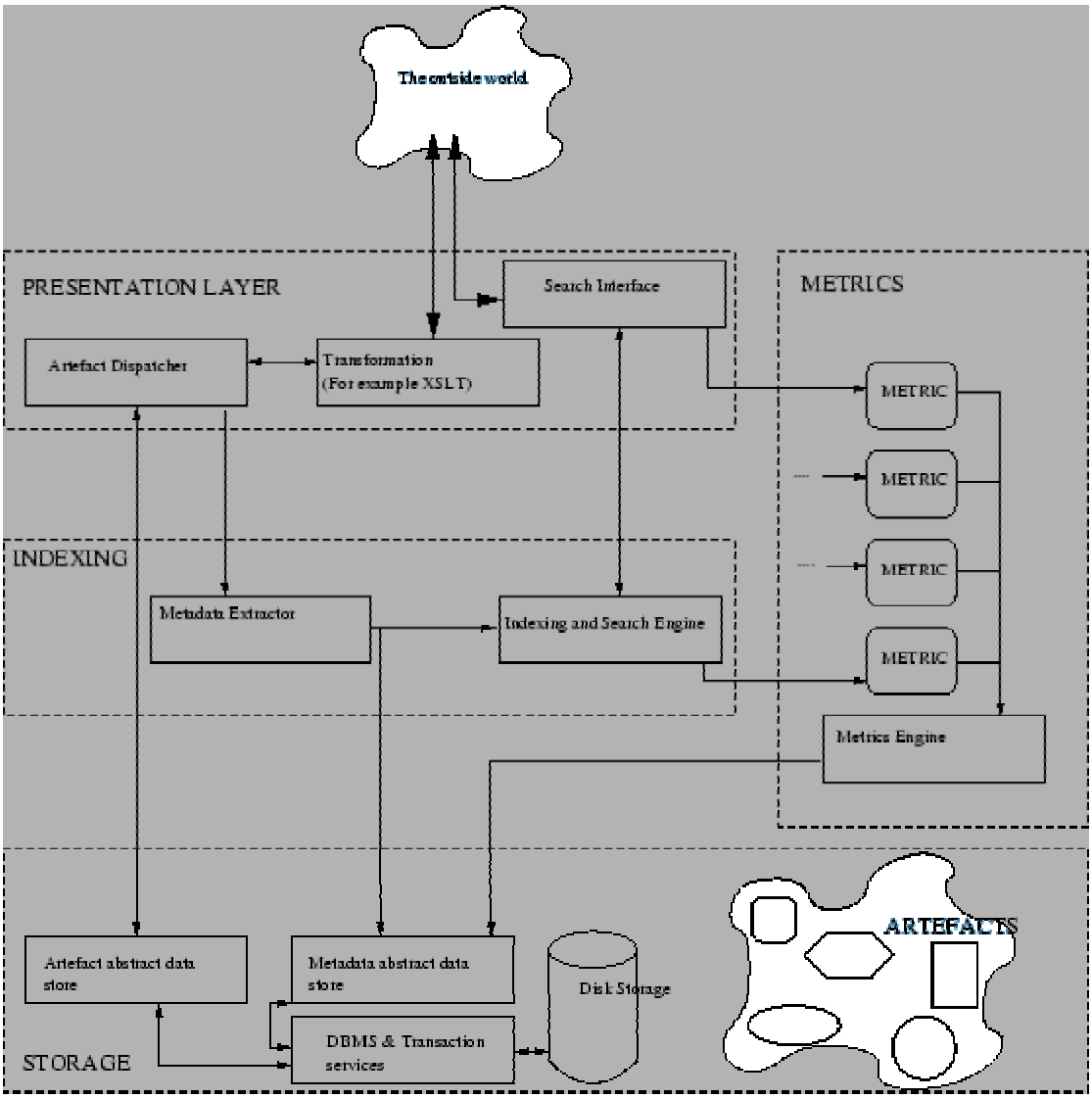

Figure 1: Outline of the OSCAR Repository

Indexing is an important feature of artefact repositories: it is important to be able to search for artefacts without requiring that the user knows the name of the particular artefact that he or she is looking for. Components in the indexing layer will primarily use component meta-data to construct indexes of the components in the repository. Many kinds of indexes may be constructed, in order to allow different kinds of retrievals. For example, a simple keyword index will allow searching in a similar manner to web search engines, while a more sophisticated indexing technique (such as self-organising maps [4]) are being investigated in order to potentially provide a more powerful search, of the form ``I have a

description of a component that I would like; find the components which you have that match this description as closely as possible".
The presentation layer manages interaction with the users of OSCAR. Users can include both people and other components of the GENESIS tool set. Finally, the metrics component interacts with the rest of the OSCAR system to monitor usage and properties of the artefact base and the artefacts contained within it. The resulting metrics are also stored as artefacts in the repository, to enable them to be accessed in the same way as any other artefact.

Project Process and Tools

This section describes the software engineering process that will be used to develop the GENESIS tools, and mentions some open source tools which might be used to support that process.

Process

For the first year of development, the GENESIS project (and, therefore, OSCAR) will be in closed-source development. At the end of this period, the first release will be made public, and the software will then be developed in the open. The leaders of the projects will be members of the GENESIS consortium. The project is currently at the stage of requirements capture and description. Once the requirements have been determined, the high-level design of the system will be determined. In contrast to some open-source projects, this will give the project a well-defined and documented architecture as a basis for the implementation of the system. It is hoped that this will allow the system to be both maintainable and understandable, and enable new developers to quickly understand the software.

Tool Support

Many open-source projects have little tool support. The tools developed by the GENESIS project will support open-source as well as closed-source software engineering. While the tools are being developed, we will be using various tools which have been developed for other open-source projects. For example, we may use CVS as source-control, Bugzilla or GNATS to track bugs and other issues, and open-source compilers and interpreters as

development tools. Development will take place on a variety of platforms, with the aim of producing portable software. As the tools which will be produced in the project will be released as open-source software, and developed by members of the open-source software community, we will ensure that we only depend upon open-source software ourselves. Eventually, the tools will be self-hosting, though obviously this will not be the case initially.

Open-Source Software Engineering

There are many open-source tools which are used to support open (and closed) source software engineering (for example: Emacs, GCC, GDB, DDD, GHC, *etc.*). Having said this, there are relatively few open-source tools for *process* support. Notable exceptions include Bugzilla [5]. Most open-source projects use ``lowest common denominator'' support, with communication *via* email. Many open-source projects have a very informal process, based on a central group of developers who perform technical and social management of the project, often with a lead developer who has overall control. Nearly every open-source project involves a distributed team of developers yet has little in the way of process support. This is acceptable for small projects with few developers, but quickly becomes unmanageable, unwieldy, and inefficient in large projects. As the Genesis tools are intended to be lightweight and flexible, and impose few burdens, they can be used to support informal processes and (potentially) to help move from informal and ill-defined processes to well-defined repeatable processes. This would provide benefits to both open and closed-source software engineering.

Conclusions and Further Work

This paper has outlined the goals of the GENESIS project and its proposed tool developments, in particular the OSCAR component repository. The concept of active artefacts has been introduced and discussed.

When the GENESIS tools have been built, they will constitute a powerful open-source resource which can be used to support open-source software engineering. This will enable better-defined and better-supported software processes to be used in open-source software engineering than is used at present in some projects. As the GENESIS tools will be lightweight and process-neutral, it should be easy for existing projects to adopt them.

Bibliography

1

C. Boldyreff, J. Newman, and J. Taramaa.
Managing process improvement in virtual software corporations.
In *Proceedings of the Project Coordination Workshop, IEEE Fifth Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 96)*, 1996.

2

C. Boldyreff, D. Nutter, and S. Rank.
Architectural requirements for an open source component and artefact repository system within GENESIS.
In C. Gacek and B. Arief, editors, *Proceedings of the Open Source Software Development Workshop*, pages 176-196, Newcastle, UK, Feb. 2002.

3

G. Button.
Keynote speach: Organisational considerations in the work of software engineering.
In C. Gacek and B. Arief, editors, *Proceedings of the Open Source Software Development Workshop*, page 1, Newcastle, UK, Feb. 2002.

4

T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela.
Self organization of a massive document collection.
IEEE Transactions on Neural Networks, 11:574-585, 2000.

5

C. R. Reis and R. P. de Mattos Fortes.
An overview of the software engineering process and tolls in the Mozilla project.
In C. Gacek and B. Arief, editors, *Proceedings of the Open Source Software Development Workshop*, pages 155-175, Newcastle, UK, Feb. 2002.