Environments to Support Collaborative Software Engineering

Cornelia Boldyreff
David Nutter
Stephen Rank

Dept of Computer Science
University of Durham

Mike Smith
Pauline Wilcox
Rick Dewar

Dept of Computing
Heriot-Watt University

Dawid Weiss
Institute of Computing Science
Poznan University of Technology

Pierluigi Ritrovato
CRMPA
University of Salerno

Abstract

With increasing globalisation of software production, widespread use of software components, and the need to maintain software systems over long periods of time, there has been a recognition that better support for collaborative working is needed by software engineers. In this paper, two approaches to developing improved system support for collaborative software engineering are described: GENESIS and OPHELIA. As both project are moving towards industrial trials and eventual public releases of their systems, this exercise of comparing and contrasting our approaches has provided the basis for future collaboration between our projects particularly in carrying out comparative studies of our approaches in practical use.

1 Introduction

From the advent of programming support environments such as UNIX Programmer's Workbench in the late 70s, through to the development of CASE tools and the Integrated Project Support Environments of the 80s, and more recent Integrated Development Environments, there has been a trend for the software industry to develop systems to support their own activities throughout the software lifecycle from development through to maintenance. Many research projects such as the Portable Common Tool Environment (PCTE) and Eureka Software Factory (ESF), and more specific projects such as PACT, REDO, AMES, ASPIS, EAST and AIMS have all developed prototype support environments. In addition, there have been language specific projects such as the Common Ada Programming Support Environments and object-oriented design and development toolsets, both from the research community and from commercial tool suppliers, the most prominent being Rational. All of these developments have focused on supporting software engineering activities throughout the lifecycle for individuals and co-located teams. Only recently has there been much attention given to supporting collaborative software engineering where the team members are distributed both in their locations and in time.

Three important factors have driven recent developments in improved system support for collaborative software engineering. With increasing globalisation of the software industry, cross organisational multi-company projects are becoming commonplace. Most large software projects are undertaken by teams of software staff working across a number of organisations. This is typical within the open source software community where projects are undertaken by variably sized teams of individuals from around the world. In both cases, the composition of the teams varies over time as members join and leave the team throughout the projects; and in some instances as one project takes over the work of another as happens when a system passes from the development team to the maintenance team and thus the nature of the work and associated support changes.

Secondly, the trend towards component based software engineering means that few projects rely entirely on their own development and maintenance efforts and consequently are forced to work in collaboration with component suppliers.

Thirdly the long life enjoyed by many large software systems means that over time large numbers of people are involved in their evolution as the composition of the support team itself changes. Thus, there is a recognised need for more flexible environments to support these diverse approaches to software engineering. It can no longer be assumed that all the members of a project will be following the same software process models, nor can it be assumed that they will be all be employing the same methods and associated software tools, nor can it be assumed that important project knowledge and expertise will be preserved over time as the project team changes, or persist along with the software system. Preservation of relevant software artefacts, i.e. all relevant work products and documentation, both formal and informal records, is of critical importance. Much of Reverse Engineering is focused on rectifying situations where the system code is the only reliable source of documentation. However, preservation of software artefacts while necessary is insufficient to support software evolution if they are disorganised and inaccessible both in the physical sense and the intellectual sense.

At present, there are two complementary projects working on the development of support for collaborative software engineering: GENESIS - GEneralised eNvironment for procEsS management in cooperative Software Engineering and OPHELIA - Open Platform and metHodologies for devELopment tools IntegrAtion in a distributed environment.

GENESIS intends to develop an Open Source platform that supports co-operation and communication among software engineers belonging to distributed development teams involved in modeling, controlling, and measuring software development and maintenance processes. Moreover, it includes an artifact management module to store and manage software artifacts produced by different teams in the course of their work.

The OPHELIA project has a similar aim of developing an open source platform to support software engineering in a distributed environment. Its primary product is a set of core interfaces that support interoperability between a range of tool categories: project management, requirements capture, modelling and software design, code generation and bug tracking accompanied by a methodology appropriate to working in a distributed manner.

The remainder of the paper will consist of the following: more detailed overviews of both GENESIS and OPHELIA. A discussion of their key differences, similarities, and complementary points; and finally an outline of areas where future collaboration is planned.

2 GENESIS Overview

The GENESIS project's focus is multi-site projects where each site is able to execute instances of a software process or subprocess that accept software artefacts as process inputs and produce software artefacts as process outputs. These artefacts form the basis for inter-site interaction. Co-ordination of software engineering activities at each site is supported by a workflow management system [1] , a notification engine, and a communication engine following an Event/Condition/Action paradigm. These components, together with an active artefact management system, OSCAR [2], allow the management of both formal and informal communication among software engineers. The whole platform will be released under an Open Source software licence and it has been conceived following a service oriented approach facilitating extensibility and simplifying its tailoring to any specific organisation's software process needs. These services form a layer sandwiched between a resource management system and the artefact management system. An overview of the GENESIS platform is given in Figure 1; while Figure 2 gives an overview of the GENESIS site architecture.

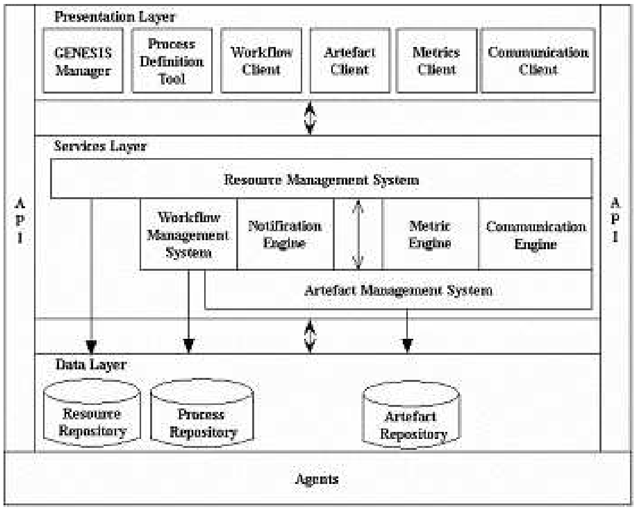

Figure 1. GENESIS Platform Architecture

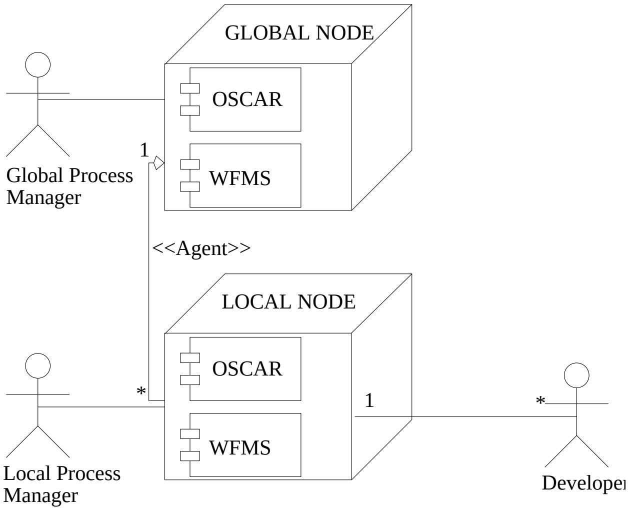

Figure 2. GENESIS Site Architecture

One of the key project objectives is to keep the level of invasiveness as low as possible. Each site is free to choose whatever tools are appropriate to their processes, varying from generic tools such as word processors, to more specific software engineering tools such as design tools or compilers. The only requirement being that the tool output should be locatable by the artefact management system, e.g. in a file system, on a web server, accessible from a document management system, etc. Once submitted to the artefact management system, an artefact's primary content remains unchanged but is augmented by metadata to facilitate its future use within the current project and potential reuse by other projects. The artefact management system also holds process descriptions and personnel profiles as artefacts to assist project managers.

Version control of artefacts is achieved through an abstraction over core configuration management system functionalities. At any particular site's instance of OSCAR, these are mapped to an underlying SCM server, for example, in the present prototype to CVS. In this way, conventional configuration management discipline can be applied to all artefacts, but the choice of system employed is left to each site.

Two novel aspects of the GENESIS project are particularly relevant to our vision of collaborative software engineering. First, the provision, through an integrated and Open Source platform, of services supporting three key software engineering aspects, namely the software process enactment and management, the active artefacts management, and the software engineers' ability to communicate and collaborate through these. Second, the choice of events as a mechanism for achieving loosely coupled interoperability of software services by delegating the management of these events to specific components such as software agents acting as global co-ordinators.

3 OPHELIA Overview

The main goal of OPHELIA [9, 7] project is to unify various types of software development tools into an abstract, transparent platform, where access to project elements and relationships among these elements is seamless with regard to the underlying software used to create and maintain them.

Among the central objectives of the project are: to provide an abstract set of programming interfaces, representing types of tools used in software development; to define how existing software can be adopted to those interfaces and to develop a prototype implementation.

The novelty of the OPHELIA project thus lies in bridging tools from different vendors into one project workspace. This integration is achieved using a set of CORBA [3] interfaces, responsible for exposing a uniform view of elements and services available in a certain area of software development process. In case of OPHELIA these include: requirements management, (UML) modelling, project management (schedules), documentation management, bug tracking and repositories of other elements of the project (such as source code).

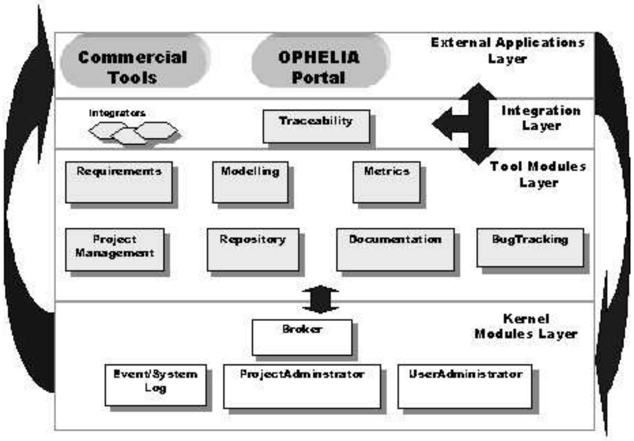

Figure 3. The OPHELIA Platform Architecture

Having established an abstraction of all available project elements, OPHELIA utilises them to provide other project-wide valuable services, such as knowledge management, semi-automatic conversion between project elements (i.e. generation of template schedules or code from an UML model), traceability (relationships among project elements), change notifications and others.

The main product of the OPHELIA project is the specification of the interfaces mentioned above called Module Interface Specifications (MIS). This architecture will therefore support the integration of a set of tools that the users choose to work with, specific tools are not mandatory (however, all the integrated tools must implement their corresponding MIS). Another product of the OPHELIA project is a prototype implementation of this architecture, called Orpheus, involving several Open Source development tools available on the market together with a deployment environment. Orpheus is a proof of concept to test the interface definitions and the platform architecture. An illustration of the OPHELIA platform architecture is provided in Figure 3.

4 Key differences, similarities and complementary points

Although the two projects are addressing the common goal of developing an environment to support collaborative software engineering, it has been instructive at recent joint meetings of staff from both projects to compare the two approaches. The two projects have taken different approaches to their support for software evolution. This is reflected in their approaches to tool integration, software process support, more significantly in the design of their repositories.

The OPHELIA project's strategy is more focused on the uniform integration of project management and system development tools through a common set of abstract tool services. Every kind of activity in project development is represented in a form of a *Module Interface Specification* interface, which constitutes a bridge between some particular tool and the rest of the platform. It is only with respect to configuration management that GENESIS has taken this strategy.

There are no requirements for artefact repositories for tools working as part of OPHELIA platform - every tool may have its own repository (i.e. CVS), some tools may share a repository (instance of a database), some types of tools may not even hold a repository at all (such as dynamically-generated metrics). What joins these all together is solely implementations of CORBA interfaces specified by the platform. Project elements are acquired not from a common repository, but via requests made to each individual module (type of tool). This also applies to meta-data, such as version information, events generated by objects and others.

The model of relationships in OPHELIA seems to be fairly similar to that found in the GENESIS project's OSCAR, though it stores relationships as intrinsic parts of the artefacts they link while OPHELIA has a separate service for managing them, which utilises object abstraction (the links only point to where an element can be found and are stored separately).

To the user, the two systems will appear very similar although the efficiency and dependability of the two approaches is an area for further investigation once both platforms are in use.

Rather than the GENESIS approach of explicit coordination of activities via Work Flow Management, one of key concepts for OPHELIA is traceability. This is determined by events at the application tool level and predefined or automatically determined relationships amongst objects, i.e. outputs of application tools. For example, from the relationships established by a conversion utility from project model to source code (and possibly documentation), a change to the requirement will result in notifications sent to people responsible for source code and documentation maintenance. In GENESIS, changes to artefacts give rise to events which can be notified to co-workers, but events are also raised from within the work flow management system.

OPHELIA is a very generic architecture - the event types and notifications can be bound to any of the pre-existing, or defined types of events. In GENESIS, event notification is realised through a notification engine which provides a similarly generic service.

The focus on configuration management differs slightly. For OSCAR, configuration management is an essential part of the environment and the system will not work without access to some form of SCM system. By contrast, OPHELIA treats SCM as another client module exposing objects to the rest of the integrated tools. Indeed one possible avenue for future collaboration is that OSCAR could be integrated with relative ease into the OPHELIA platform to provide artefact management functionality!

Unlike the GENESIS project, the OPHELIA project has actually undertaken specific tool development. An open source requirements management tool has been developed and integrated (via MIS) with ORPHEUS. Initial work has also been done in integrating different types of development tools using MIS specifications. For example, an integration of metrics generation based on data acquired from modelling MIS (with underlying Argo/UML), or an integration of modelling MIS with project management MIS (Argo/UML with MS Project). Work on higher level services such as documentation generation and cross-module object traceability has been started as well.

OPHELIA also differs from GENESIS in the level of integration. Tools integrated with OPHELIA need to be in contact the system when performing any operation on the data (such as load/save). Tools used to create data for use in GENESIS do not need to be modified [1], nor to have contact with OSCAR when working on the data. Much of OSCAR's integration will rely on internal transformation of data to extract appropriate meta-data from files under its control whilst OPHELIA relies on modification of the client tools.

Both projects introduce unique object addresses (in the form of URLs). In GENESIS these addresses may point to any artefact, as well as any specific version of an artefact. In OPHELIA only elements (objects) can be addressed. Versioning is therefore not part of the generic object definition in OPHELIA.

5 Identified areas for GENESIS/OPHELIA collaboration

Currently both projects are completing the first releases of their platforms; and as these undergo further development, they will be also be trialled by the respective projects' industrial partners and possibly within the open source development community. Both projects intent to instrument their developments and collect usage and performance data. In order to make comparisons between these platforms in use, a common set of basic measures and monitoring procedures will be agreed and implemented. It is intended that these will allow joint studies on efficiency and dependability. The two projects also intend to investigate the potential for uniting both platforms in a multi-organisation project.

[1] As long as they operate on standard files, if they do not then they must be integrated with OSCAR.

Acknowledgements

We would like to acknowledge our colleagues who have contributed to the development of the research discussed here and in the development of the GENESIS and OPHELIA research programmes.

References

[1] Aversano, Lerina; Cimitile, Aniello; Gallucci, Pierpaolo; Villani, Maria Luisa (2002), “FlowManager: a workflow management system based on Petrie nets”, in the Proceedings of the 26th Annual International Computer Software and Applications Conference, COMPSAC02, IEEE Computer Press, pp. 1054-1059.

[2] Boldyreff, Cornelia; Nutter, David and Rank, Stephen (2002), “Active Artefact Management for Distributed Software Engineering”, in the Proceedings of the 26th Annual International Computer Software and Applications Conference, COMPSAC02, IEEE Computer Press, pp. 1081-1086.

[3] CORBA (2002) [@:] http://www.corba.org

[4] Dewar, RG; MacKinnon, LM; Pooley, RJ; Smith, AD; Smith, MJ & Wilcox, PA (2002); “The OPHELIA Project: Supporting Software Development in a Distributed Environment”, IADIS WWW/Internet 2002

[5] Gaeta, Matteo & Ritrovato, Pierluigi (2002), “Generalised Environment for Process Management in Cooperative Software Engineering”, Proceedings of the Workshop on Cooperative Supports for Distributed Software Engineering Processes, in the Proceedings of the 26th IEEE Annual International Computer Software and Application Conference, August 2002, pp. 1049-1053.

[6] Genesis project website (2002) [@:] http://www.genesis-ist.org/

[7] Hapke, M.; Jaszkiewicz, A. & Perani, S. (2001); “OPHELIA – Open Platform and metHodologies for devELopment tools IntegrAtion in a distributed environment”, Proceedings of 3rd National Conference on Software Engineering, Otwock/Warsaw, pp. 189-198.

[8] Kowalczykiewicz K., Weiss D. (2002) “Traceability: Taming uncontrolled change in software development”, Proceedings of IV National Software Engineering Conference, Tarnowo Podgorne, Poland, 10 pages.

[9] Ophelia project website (2002) [@:] http://www.opheliadev.org