An Artefact Repository to Support Distributed Software Engineering

David Nutter Cornelia Boldyreff Stephen Rank
Research Institute for Software Evolution
Department of Computer Science
University Of Durham, UK
{*cornelia.boldyreff,david.nutter,stephen.rank*}
@durham.ac.uk

Abstract

The Open Source Component Artefact Repository (OSCAR) system is a component of the GENESIS platform designed to non-invasively inter-operate with work-flow management systems, development tools and existing repository systems to support a distributed software engineering team working collaboratively. Every artefact possesses a collection of associated meta-data, both standard and domain-specific presented as an XML document. Within OSCAR, artefacts are made aware of changes to related artefacts using notifications, allowing them to modify their own meta-data actively in contrast to other software repositories where users must perform all and any modifications, however trivial.

This recording of events, including user interactions provides a complete picture of an artefact's life from creation to (eventual) retirement with the intention of supporting collaboration both amongst the members of the software engineering team and agents acting on their behalf.

1 Introduction

The GENESIS platform [5] provides an Open Source solution for modelling and enacting work-flow processes and managing both planned and unplanned work products. Process enactment is distributed over multiple physical sites coordinated by a global process. Both local and global processes are managed by GOSPEL (GENESIS Open-Source Process Enactment Language). Similarly the work products managed by OSCAR are visible in a similarly global name-space composed of multiple OSCAR repositories. While the work-flow and artefact management components are intended to work together and with other applications, as shown in figure 1, either may be used alone. Therefore, OSCAR has several capabilities beyond those required by the work-flow management system.

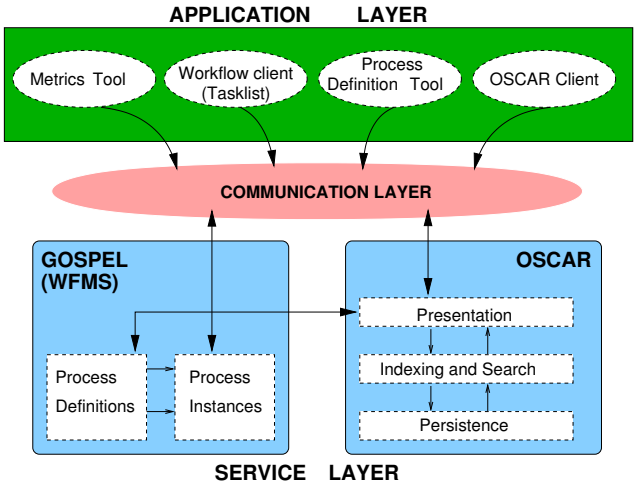

Figure 1. Overview of the GENESIS platform architecture

Firstly, OSCAR represents all data under its control as an artefact, including users of the system, the process models and instances they are using, tools and services that the system employs directly, output from metrics and other monitoring procedures that act on the repository as well as data created by tools not directly integrated with OSCAR. To support this "unified name-space", similar in intent to the UNIX file system and active directory services, a set of types with a common ancestor has been defined in earlier work [11, 2]. Secondly, since non-invasivity is a design goal of the GENESIS platform, many of OSCAR's basic functions are provided by incorporating existing Open Source applications which are in widespread use. Most notably in the current development version, version control and configuration management functionality are provided by a CVS[1] plugin and meta-data is recorded and queried in a

[1] http://www.cvshome.org

PostgreSQL[2] database accessed via a plugin. OSCAR provides default plugins of this type but has well defined APIs for plugin introspection, loading and initialisation allowing new systems for to be integrated easily, providing OSCAR with greater interoperability and capabilities. Finally, the complex and highly mutable nature of artefact meta-data makes dependability a concern. Consequently the current (accurate) meta-data is verified and stored in the SCM system alongside the artefact data every time a new version of an artefact is stored. Should the meta-data store in the database become corrupt, a previous valid version of the meta-data may be restored transparently. These two separate data stores are presented to the rest of OSCAR as a single "persistence" layer.

Regardless of the underlying data storage systems, OSCAR artefacts are presented as an XML document containing both the meta-data and the artefact data. This document is then used as a flyweight for an object; the properties of which reflect the state of the document allowing an update to the document to be visible in the object properties and vice versa. Using this model, applications merely processing the artefact data can access the document while more complex behaviour can either be implemented directly in the object, or in a wrapper class. Figure 2 shows each layer of the artefact.

1.1 Related work

OSCAR's meta-data has much in common with that stored by knowledge management systems. The Standardised Content Archive Management (SCAM) [10] system manages collections of information structured using IMS [6] meta-data and Dublin Core [3] for descriptive meta-data alongside their own application specific meta-data. IMS is intended for a specific kind of repository, that used in learning environments and is therefore not directly useful to us, however the design of SCAM is similar to that of OSCAR.

Persistent storage software such as OCEANStore [8] is also relevant to OSCAR. However, OCEAN concentrates on providing dependable distributed file-system storage for all types of file data while OSCAR concentrates on providing distributed access to a collection of software artefacts each being potentially modified by a team. Therefore OSCAR focuses on exposing configuration management functionality and meta-data while OCEAN focuses on providing an efficient and dependable file-system. OCEAN could conceivably be used in a future OSCAR persistence layer to allow a "roving" OSCAR installation or as an innate distribution mechanism for OSCAR itself.

[2]http://www.postgresql.org

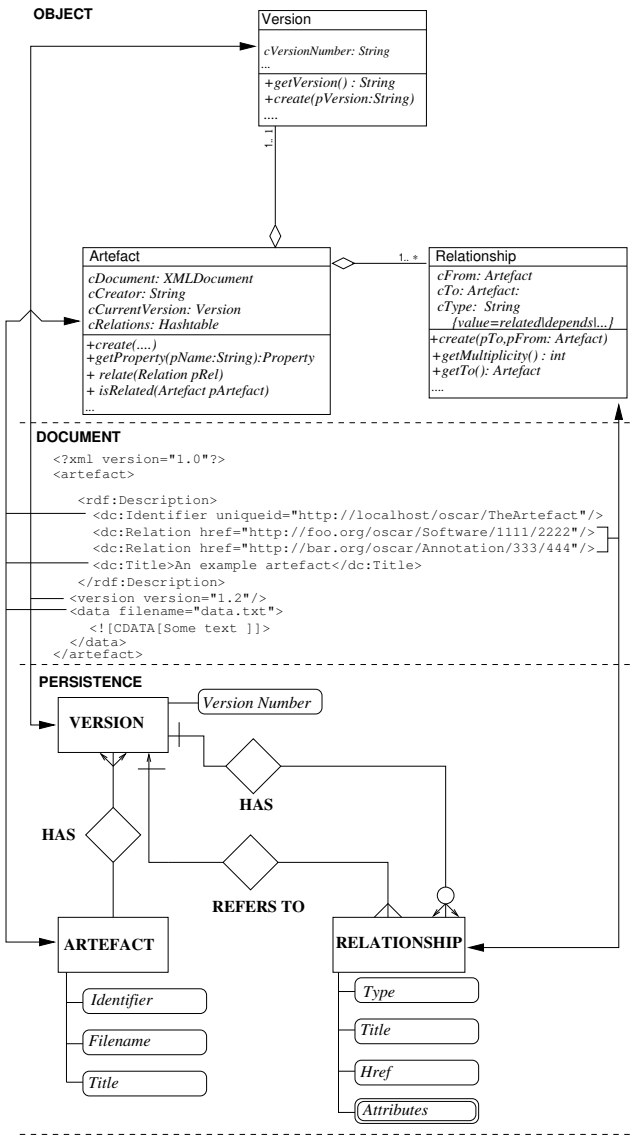

Figure 2. The Three Layer Model Of Artefacts

2 Artefacts

Within the three layer model of artefacts described earlier, much of the customisation takes place at the document level. In particular, the different meta-data models are incorporated at this level. For each artefact type, a DTD or schema fragment extends the base DTD to add the necessary extra functionality. Therefore conformance to a particular DTD indicates that an artefact is of a specific type. For example, an artefact containing a UML model may include XMI information alongside the Dublin Core meta-data associated with every artefact. The DTD describing this artefact will append the XMI specification to the meta-data model. However, the artefact will still conform to the basic artefact DTD allowing applications and parts of OSCAR that do not understand XMI to continue working with the artefact.

At the object level, customisation takes place by extending the classes as usual. Much of the low-level document manipulation is performed by the abstract base class and any extending class should be able to extend the behaviour without needing too much XML manipulation knowledge. At most, a new "artefact type" consists of three small extensions to OSCAR subsystems:

1. The DTD/schema extension to the base artefact model (or another type if that type permits extension).
2. The new artefact class (optional).
3. A piece of transformation code to transfer the XML meta-data into the meta-data storage system and retrieve the artefact (optional).

OSCAR itself provides several built-in types and some suitable defaults to store any type of artefact meta-data (though not necessarily in the most efficient way). It is hoped that these types will prove sufficient for most of the purposes new users have in mind. These types include:

Software Used to contain bits of software, including requirements and design documentation as well as code.

Annotation Used to contain informal comments or notes about artefacts.

HumanResource Contains either a link to the relevant information in the GENESIS Resource Management systems or a complete set of resource information in any format, though meta-data key/value pairs are suggested. Either way, this artefact is used as part of "created by" or "modified by" relationships belonging to other artefacts.

Project Like **HumanResource**, this represents a link to active project information in the resource management system, or if required a piece of user data describing the project. Relationships may then be used to associate other artefacts with the project.

Default The simplest type of artefact, this is used when no others will "fit". It contains a piece of data and information describing who put it there, nothing more.

OSCAR's base meta-data model is designed to be in conformance with the Semantic Web [1] by employing open standards with tool support such as RDF [9] and (specialised) Dublin Core meta-data. Additionally the prototype system is internet-ready by default; every unique artefact identifier is a URI which, in conjunction with a version number if a historical version is desired, can be used to retrieve a copy of an artefact. Ideally, accessing the URI directly will yield a definitive copy of the artefact from the location where it was created but since OSCAR systems may be transient this may not always be possible.

2.1 Event Notification

To support the "active" behaviours described earlier, every operation on an artefact generates events within OSCAR. These events may be propagated by an event monitor to artefacts deemed to be "interested" in such events by the relationships they have with the event generating artefact. For example, an artefact which has a dependency upon another will be interested in change events emanating from that artefact and will record such events in its meta-data, whilst an artefact merely related in some ad-hoc way will only be interested in events (such as delete) that make it impossible to maintain the relationship. Certain events may also be propagated outside of OSCAR in the form of notifications to members of the software engineering team, agents active on their behalf, or as inputs to a metric calculation engine.

This technique will be first be used to inject events from OSCAR into the Java Messaging Service (JMS) based event notification system used by the rest of the GENESIS platform. To do this a custom message handling extension to OSCAR will be written to transform the very lightweight OSCAR events into a form acceptable to JMS and deal with any problems arising from the JMS connection. It is not intended that OSCAR should receive direct events from outside as the existing client interfaces provide sufficiently flexible event generation through the existing artefact operations without needing clients to access OSCAR's internal workings directly.

2.2 Version Control: CVS

OSCAR defines an abstract interface for versioned storage of artefacts. Currently, the only implementation of this

interface is based on `pserver` access to a remote CVS repository using the jCVS[3] software. We intend to provide at least one more implementation of this interface using the Perforce Version Control System (as requested by our industrial partners).

Though it is possible to use the same repository with standard CVS and OSCAR at the same time, this approach cannot be recommended as the CVS plugin in OSCAR cannot take into accommodate changes made to the repository outside of OSCAR. This is an unfortunate limitation but due to the differing implementation technologies of OSCAR and CVS (Java and C respectively) make any interaction beyond that provided by the `pserver` protocol difficult.

2.3 Queries and Transformation

A key part of repositories of all kinds is the searching and indexing functionality which has two main requirements. Firstly, they must take user-amenable queries and translate them into a form that can be applied to the data in the repository. Secondly, they must extract and index appropriate information from the meta-data associated with the repository contents to ensure searches are efficient. OSCAR shall initially support two main types of query. Firstly, a simple keyword search familiar to users of Internet search engines will be provided. Secondly, a form of similarity matching shall be available where a "template" artefact is prepared containing the properties the user requires. The best matches will then be returned, subject to ordering and pruning by the client software.

Several options for autonomous indexing are being explored, the most promising currently is the use of a self-organising map [7] to prepare an ordering over the artefacts stored in OSCAR. At present however searches must be enacted in a meta-data storage back end specific way (using SQL), a method both inflexible and non-portable.

When delivering query results and located artefacts to clients, the quantity of information available (especially for older artefacts with large numbers of recorded events in the meta-data) may be problematic. Though some filtering will occur at the client level, a significant amount of server-side transformation should take place on the artefacts to remove meta-data obviously unnecessary for the tasks clients are performing on the artefact. We intend to implement a tool to dynamically assemble a "pipeline" of transformations based on the document types of artefact. Figure 3 shows an example pipeline. The artefact representations have a type (in bold), initially the source artefact type. Additionally the document-layer data type (initially XML) is indicated in monospace. To start with, a transformation that doesn't affect the representation type or document-layer type but merely prunes meta-data is performed[4]. Then, two potential transforms can be applied to the artefact representations, one outputting a HTML representation and another a PDF representation. Finally, it is possible to further transform the HTML into plain text.

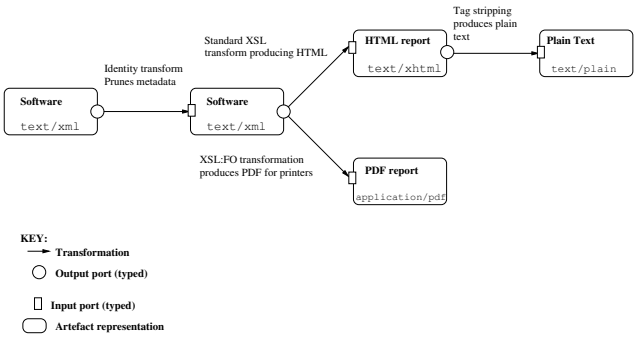

Figure 3. Example Transformation Pipeline

With a suitably large pool of transformers to integrate into pipelines it is expected that most client and context requirements will be fulfilled without the need to write explicit transformations for each task and client combination. Client requirements are defined as the limitations or capabilities of particular applications (such as display capability) and context requirements as the information necessary to perform a particular task. For example, editing a document requires different meta-data and delivery format than merely preparing a document for web publication.

3 Plans for Future Collaboration Support

Currently, OSCAR is a system designed to be deployed on a corporate intranet or on the Internet rather than more ordered environments such as learning networks or conversely less ordered environments such as Open Source development efforts. To address these concerns and make the system more flexible, planned modifications to the system will combine the current "tight" (RMI) with "light" (web-based interface) access methods into a service-based approach, combining the best of both worlds.

A service-based approach to computing is currently timely due to the rise of grid computing. In this context OSCAR at least would initially occupy a niche identified by Foster et al [4], namely that of the code repository, a specialised form of Grid connected database [12] closely aligned to the XML repository. However, OSCAR's various extra capabilities mean it has relevance in the *Collective* layer of the "Hourglass Model" of Grid subsystems as well as the *Fabric* layer where the code repository is located. Aside from this widespread application of service-

[3] http://www.jcvs.org

[4] This should take parameters identifying the meta-data to be pruned.

based computing, OSCAR will continue to support the original methods of access by wrapping the services in the familiar interfaces.

It is intended that artefacts (in XML document form) will provide a happy middle-ground between rigid database schemas and free-form files by associating a common structure and set of meta-data with ordinary file data. With the addition of the object built on the XML, integrating artefacts with work-flow systems or any application that requires an API rather than a document to process will be relatively easy. Indeed, our experiences with integrating OSCAR with the work-flow component in the GENESIS platform should provide useful experience here.

Whether artefacts are stored locally or remotely does not matter as OSCAR may propagate events generated by artefacts within the local system to clients within that local system or, via an external messaging service, to remote systems not directly connected to the system. Current plans are to integrate OSCAR's internal event system with the JMS-based GENESIS notification system.

In order to support collaboration, OSCAR will supply the following facilities to clients:

- Provision of a set of resources uniquely identified by URIs with associated meta-data to facilitate discovery. Each resource may have multiple versions with associated historical meta-data and relationships between them
- Using these resources, maintenance of an archive of all captured discourse regarding particular artefacts leading to a complete picture of a particular artefact's development.
- Presentation of the artefacts to different clients in an appropriate form by assembling transformation pipelines.

The first facility prevents the decay of relationships between resources over time, for though the purpose and human-readable title of an artefact may change over time any relationships will still point to the correct item. The archive which relies on the first facility provides awareness of both the history and current context of a particular resource since no discourse will have been discarded at archive creation time; instead the final facility will prevent spurious information from reaching the user.

4 Conclusions

The GENESIS platform provides process-based support for distributed software engineering over the internet. Within the GENESIS platform, OSCAR provides sophisticated facilities for the management and control of artefacts. While GOSPEL provides support for the project management, OSCAR supports general software engineering tasks. The GENESIS platform is intended to be lightweight, in that it should be possible to install the software in an organisation without the requirement to adapt that organisation's software processes. In order to support this, it will be possible to install the components of the platform incrementally, to allow the gradual adoption of the technology. The process modelling language used by GOSPEL allows each project to use the GENESIS platform with their current practices: the tools adapt to fit the users, rather than vice versa.

OSCAR itself aims to provide basic collaboration services to software engineers such as configuration management, awareness of change, meta-data storage and search while the rest of the GENESIS platform aims towards an integrated solution for process-driven software development.

The GENESIS tools will be released under an open-source license, which allows adopters freedom to adapt the platform. The tools use and can operate with various open-source software which provides database services, persistence, instant messaging, and so on.

References

[1] T. Berners-Lee. Semantic web road map. http://www.w3.org/DesignIssues/Semantic.html, September 1998.

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

[3] Dublin Core Consortium. Dublin Core Metadata Elements Set: Version 1.1. http://dublincore.org/documents/1999/07/02/dces/, July 1999.

[4] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organisations. *International Journal of Supercomputer Applications*, 15(3), 2001.

[5] M. Gaeta and 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] IMS Global Learning Consortium. IMS learning resource meta-data XML binding: Version 1.2.1. http://www.imsglobal.org/metadata/imsmdv1p2p1/imsmd_bindv1p2p1.html, September 2001.

[7] T. Kohonen, S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transaction on Neural Networks*, 11(3):574–585, May 2000.

[8] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon,

W. Weimer, C. Wells, and B. Zhao. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of ACM APSLOS*. ACM, November 2000.

[9] O. Lassila and R. R. Swick. Resource description framework (RDF) model and syntax specification, February 1999.

[10] M. Nillson, M. Palmer, and A. Naeve. Semantic web metadata for e-Learning: Some architectural guidelines. In *Proceedings of the 11th World Wide Web Conference*. Royal Institute Of Technology, Stockholm, 2002.

[11] D. Nutter, S. Rank, and C. Boldyreff. Architectural requirements for an Open Source Component and Artefact Repository System within GENESIS. In *Proceedings of the Open Source Software Development Workshop*, pages 176–196. University Of Newcastle, February 2002.

[12] P. Watson. Databases and the Grid. Technical Report UKeS-2002-01, National e-Science Centre, Department Of Computer Science, University of Newcastle, 2002.