Historical Awareness Support and Its Evaluation in Collaborative Software Engineering

David Nutter and Cornelia Boldyreff
Department Of Computer Science
University Of Durham
{*david.nutter,cornelia.boldyreff*}*@durham.ac.uk*

Abstract

The types of awareness relevant to collaborative software engineering are identified and an additional type, "historical awareness" is proposed. This new type of awareness is the knowledge of how software artefacts resulting from collaboration have evolved in the course of their development.

The types of awareness that different software engineering environment architectures can support are discussed. A way to add awareness support to our existing OSCAR system, a component of the GENESIS software engineering platform, is proposed. Finally ways of instrumenting and evaluating the awareness support offered by the modified system are outlined.

Keywords: Awareness, OSCAR, collaboration, workspaces

1 Introduction

Collaborative software engineering environments require awareness support to ensure users of the environments know of the activities of others in order to coordinate their work, identify potential problems and prevent conflict. One definition of awareness that is especially relevant in this domain is

> "An understanding of the activities of others which provides a context for your own activity".[8]

Where a number of software artefacts are being developed, often concurrently, one way to provide awareness is to build it into a common repository, such as the Open Source Component Artefact Repository (OSCAR[18]), monitor changes made to artefacts and use this data as the basis for awareness provision. However, care must be taken to ensure that data collection and awareness provision do not disrupt activities themselves.

To ease adoption by industry and the Open Source community, a key design goal in the development of OSCAR has been *non-invasiveness*. OSCAR is process aware but does not require its adopters to change their existing working practices and tools, and it may be adopted on its own without the rest of of the Generalised Environment for Process Management in Collaborative Software Engineering (GENESIS[11]) platform. Similarly awareness support within OSCAR must be non-invasive.

Building awareness into OSCAR will provide minimally invasive support for indirect coordination and communication by alerting collaborators to the changes made to artefacts. In the longer term, analysis of the raw data that the awareness support uses will allow studies of development projects using OSCAR; and through these, we shall evaluate the support that OSCAR provides in order to build better support for collaborative software engineering in future.

Our initial motivation to provide support for awareness is based on our experience and that of our partners in developing the GENESIS platform. Top down, rigidly imposed methods of collaboration largely failed when working on software artefacts consisting of many files (such as code) though they succeeded with monolithic artefacts such as Word documents. As the project moves towards self-hosting its own development on GENESIS/OSCAR, integrated awareness support will allow us to collaborate during the development of software artefacts without spending excessive amounts of time reading change logs and e-mail archives all managed by separate tools to understand the history of a particular artefact.

Section 2 describes the existing OSCAR system, section 3 describes related work, existing systems and common architectures for awareness support, section 4 proposes the addition of awareness support to OSCAR, section 5 discusses the evaluation of our improved environment and section 6 identifies future work.

2 OSCAR

OSCAR is a client/server application designed to manage and deliver XML based artefacts for both human users, i.e. software engineers, and programs which access OSCAR via its API, i.e other components of the GENESIS platform including work-flow[2] and resource management.

Figure 1 is a simplified architecture diagram of a single OSCAR system. The **StorageManager** components may be stub objects that call a remote OSCAR , allowing (explicitly configured) federation. Each OSCAR installation has a CVS repository to store the data component of artefacts and a relational database to store the meta-data component. Users see the two unified as XML documents or Java objects, depending on their requirements. Interacting with artefacts on the server generates events which OSCAR can either ignore or propagate to interested parties, including other artefacts. This allows artefacts to respond "actively" to changes in related artefacts. All data stored in the system including source code, documents, user and project information is represented as artefacts with common meta-data such as creator, contributor, modification time etc.

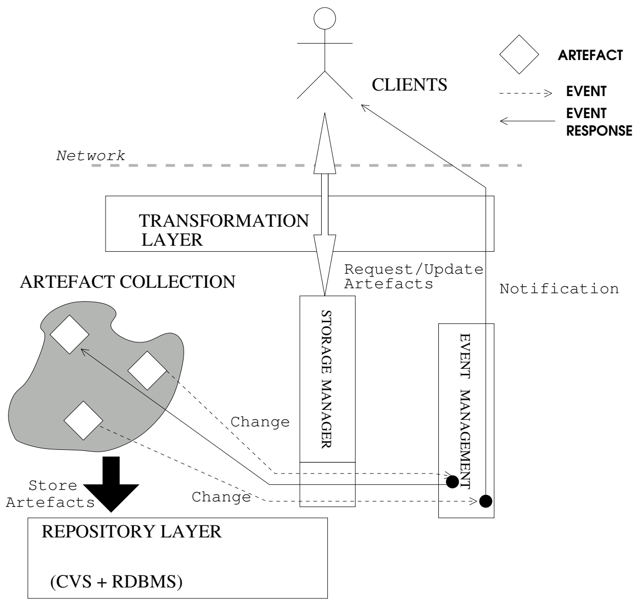

Figure 1. OSCAR architecture

The system is intended to support all software life cycle phases by storing artefacts produced either as the result of the GENESIS work-flow processes or by users directly with their existing tools. The architecture relies on a plugin system which encourages integration with existing software such as SCM to allow minimally invasive system set-up. OSCAR does not require users to migrate wholesale from their existing software in order to benefit from OSCAR's features.

3 Related Work

Several informal definitions of distinct awareness types have been collected by Drury et al[9] from previous works. Table 1 describes the relevance of those types useful in an environment like OSCAR.[1]

Table 1. Awareness Types and Purpose

Type	Definition
informal awareness	*"The general sense of who is around and what others are up to"* This is important as a prerequisite for "group structural awareness" and a baseline for informal collaboration.
peripheral awareness(2)	*"Where people know what others are doing"* Full direct awareness is not possible with a system like OSCAR, so such support is necessarily limited to interactions with the OSCAR system rather than interactions with a tool. However, since the point when changes will affect other users of the OSCAR system are when changes are sent to the central system, this is sufficient.
social awareness(2)	*"information about the presence and activities of people in a shared environment"* Presence information promotes collaboration while activity based information prevents conflicts of interest over artefacts
group structural awareness	*"Knowledge of roles and responsibilities, their positions on an issue and process information"* OSCAR's relationships model can provide such information; such awareness will be used to prevent conflicts of interest over certain artefacts.
workspace awareness(2)	*"Who is working on what"* Relating the human and machine actors to specific software artefacts is an important function of OSCAR. As before, such knowledge will be used to prevent conflicts of interest.

Several kinds of awareness identified by Drury are important, yet omitted from table 1. These are the *synchronous* awareness types that rely on timely propagation of information between users of systems such as Rear View Mirror[4].

[1]Numbers in brackets indicate the definitions order in Drury et al

Since OSCAR's clients have limited integration with tools and are not permanently connected to OSCAR, supporting synchronous awareness cannot be guaranteed. Additionally, *task* and *concept* oriented awareness cannot be supported by OSCAR; however, the GENESIS platform's work-flow management system can provide this support if desired.

Awareness types that that fit with OSCAR's minimally-invasive philosophy are largely *asynchronous*, *informal* and *workspace awareness(2)* limited to operations on artefacts within OSCAR. Complete awareness for all OSCAR session participants is not a goal due to the large overhead incurred.

3.1 Existing systems

One prerequisite for asynchronous awareness is ensuring that each artefact possesses a rich history describing what happened to it and when[15]. This information is the basis of retrospective awareness such as "source code wear"[16], a visualisation of source code change history. This type of awareness is closely related to *peripheral awareness*.

MITRE evaluated two multiple-participant collaboration systems[7] focusing on synchronous collaboration (though replay was possible). In contrast to the "wear" system this experiment instrumented the collaboration tools along with change tracking, providing a richer interaction record. Several different visualisations were employed such as a time-line of data accesses during a session, providing full *workspace* and *social awareness*.

InterLocus[17] implements an interesting form of *workspace awareness*, without needing shared workspaces. Instead a series of snapshots the user's files are taken and the change details used to generate awareness information. This approach allows the use of any tool but requires synchronously connected clients and fast communication links. A simpler approach to up-to-the-minute workspace awareness is provided by Radar Views[14] which provide *presence* and simple *action* awareness (via telepointers) rather than detailed knowledge of changes.

SPE/JViews-based environments[12] manage inconsistencies during software development, supporting round trip engineering, inconsistencies visualisation, *peripheral*, *social* and *workspace awareness* using "Change Objects". The developers of SPE/JViews discovered that their users liked uncluttered graphical awareness views but needed detailed text views too; much the same was said of Radar Views.

Shared editing environments are a special form of shared workspace, relying on synchronous collaboration to produce shared documents and diagrams. Lessons learned from ShrEdit[8] indicate that prior, static assignment of roles is not a successful way of providing *group structural awareness* as roles change frequently. Awareness data must be obtained at no cost to information providers as otherwise the amount of awareness obtained will be minimal. The ShrEdit system did not provide full *workspace awareness*; an acknowledged limitation. Shared diagram development[6] has divergent requirements to text editing and consequently different awareness needs. As with ShrEdit, for small sessions social norms supported by *informal awareness* are good enough, but for more permanent diagrams formal and enforced task/concept awareness is required.

In light of this work, we propose a new type of awareness for collections of artefacts: "historical awareness". Like the retrospective awareness discussed earlier, historical action information is presented to the user but unlike retrospective awareness, historical awareness deals with a collection of heterogeneous artefacts allowing the user to view the complete context of an artefact's creation and change into its present form rather than a contextless view of changes to a single artefact.

Historical awareness is superficially similar to changelogs and history views provided by SCM systems but, unlike these systems, provides information that has not been explicitly requested by the user.

3.2 Common architectures for awareness

A simple architecture for awareness support is the "reflector" model. Each client transmits activity details to the reflector, which distributes this information to other clients, allowing awareness displays on each one. Though simple and capable of providing a complete awareness picture, the number of clients is limited by the resources available to the reflector system. Such systems include WhitePine CU-SeeMe and Microsoft Netmeeting.

Peer-to-peer awareness architectures do not have a reflector; instead each client knows of several peers with which it exchanges information[3]. Clients in a peer-to-peer awareness network cannot assume that they know of all activities as they may not be linked to all peers, nor that they are receiving timely information. The size of the network is limited by the capacity of communications links between peers and the presence or absence of a peer discovery system.

Another enabling technology for awareness is *publish-subscribe*. Events are generated by the system and clients need only receive those they are interested in. They indicate interesting event types by making a *subscription*. The MOTION[10] system uses publish/subscribe to support mobile teamwork, awareness support. Two problems exist with publish/subscribe:

- finding the correct subscriptions to ensure a client receives all pertinent events and

- preventing information overload through over-zealous subscription!

Complex filtering regimes require fast communication (if implemented on the client) or can slow event delivery (if implemented on the server). Campailla et al[5] describe a filtering system intended to deliver messages quickly to clients with complex subscriptions.

Finally, "shared view" systems have enjoyed a renaissance with tools such as Virtual Network Computing[1] used with awareness-capable tools such as shared editors which incorporate presence awareness and shared interaction with a single application.

4 Adding awareness support to OSCAR

Table 2 lists some activities OSCAR supports and the types of awareness they require. The first five are specific features of OSCAR; the remaining are two example activities that OSCAR will be used for in the GENESIS project evaluation.

Activity	**Awareness Types**
Edit artefact meta-data and data	Social/presence, workspace
Joint development of artefacts	group structural, workspace, *historical*
Relate artefacts by project, creator etc	group structural, *historical*
Re-using existing artefacts	workspace, group structural, *historical*
Annotation of artefacts	informal, conversational
Support software testing	task/concept, workspace, conversational, group structural
Development of web application	task/concept, workspace, group structural

Table 2. Activities supported by OSCAR

The most basic features of OSCAR are creating and editing the stored artefacts. The awareness types useful here are simple: presence awareness so users know who is around and workspace awareness to describe what is happening to the artefacts of interest. Joint development has additional requirements as users must be aware of group information in order to collaborate, alongside the workspace awareness necessary for preventing conflict. This group awareness information can be derived from relationship meta-data associated with each artefact, which includes information such as creator, contributors, any projects the artefact is related to etc.

Group structural awareness can also be used to discover which artefacts are used elsewhere by other users, indicating potential re-use candidates. Annotation artefacts exist to contain supplementary information (such as case studies, communication records etc.) and can provide a form of conversational awareness: "who is talking about which artefacts" related to workspace awareness.

Both the evaluation tasks in the table rely on task/concept awareness which cannot easily be supported by OSCAR without the GENESIS work-flow system. A discussion of such integration is beyond the scope of this paper. Both tasks rely on workspace and group structural awareness indicating that these types are the most useful for the GENESIS project evaluation. The evaluation of software testing will also benefit from conversational awareness, indicating to users what others say about test-case results and the software they test.

4.1 Approach

In order to support awareness, another subsystem must be added to OSCAR which keeps track of clients connected to OSCAR both directly or via any distribution mechanism. The types of awareness those clients request will determine the information sent to them; a simple way to avoid sending all the information. Furthermore the event management system must be connected to an extensible awareness information generator to create the requested information. On the client side, a generic container for the awareness display must be available to support multiple awareness views.

The revised architecture is shown in figure 2.

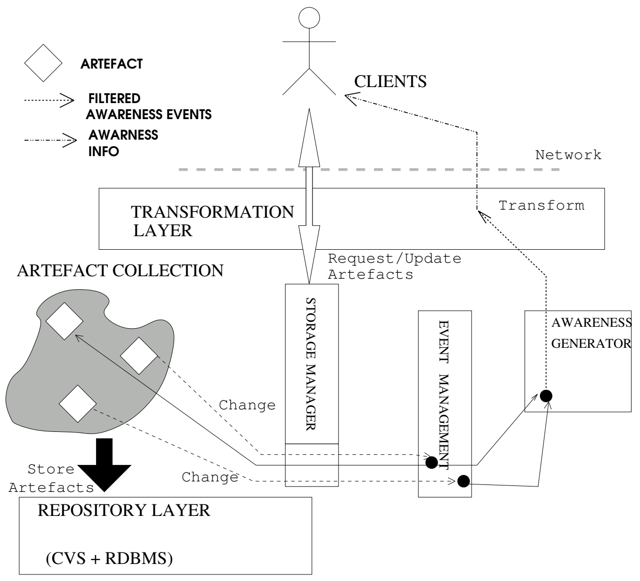

Figure 2. Revised OSCAR Architecture

OSCAR's existing event management subsystem will be used to collect events containing useful information which will then be passed to an awareness information generator where initial filtering and sorting of the events will take place. The organised collections of events will then pass to the transformation layer which shall perform final filtering and turn the events into a form that the clients can render.

The proposed prototype approach is oriented to timeline awareness display of artefact changes. Figure 3 shows an example timeline including a key for the symbols shown. The proximity of the lines indicates how close the system believes the two users' activities are to conflicting. Timelines may be drawn for artefacts and the actions of users. The conversion of raw OSCAR events into timeline information will take place in the existing transformation subsystem of OSCAR and the resulting output transmitted to the clients for rendering. This approach has the advantage that timelines can be applied to past versions of artefacts or applied in soft real time with variable granularity of the displayed events achieved by disregarding events deemed unimportant. This variation in granularity allows the awareness display to cope when events are missing, merely resulting in a gap on the timeline instead of system failure.

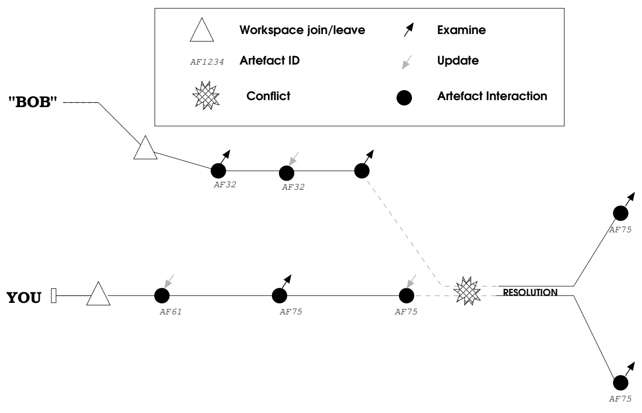

Figure 3. Example Timeline

4.2 Instrumentation

The presence of centralised event management and dispatching in both forms of OSCAR's architecture allow system instrumentation by recording these events at the server side, irrespective of whether they are subsequently sent to the clients. These events may then be used as a session record or to calculate metrics about OSCAR sessions, supporting the evaluation of OSCAR described in section 5.

Though collecting all information from instrumented clients would yield more raw data, establishing the temporal order of events between clients would be difficult, also clients may lose information due to operator error. Since client actions are only "important" in historical awareness terms when they can affect other artefacts, i.e when they act on artefacts held in OSCAR rather than local copies, a server-only approach is sufficient for our needs. Additionally, "server-only" instrumentation is easier to deploy for production environment studies than multiple, potentially distributed, clients which need to "phone home" on a regular basis to return their event logs.

5 Evaluation

To ensure that the support we are developing addresses the motivating concerns outlined in section 1 we must evaluate the awareness support provided by OSCAR. Since OSCAR is intended for all phases of everyday software engineering, studying the actions of users by analysing event data and session logs collected transparently will minimise interference by the experiment with the user activities.

For example, to find out how our historical awareness and timeline views affects the incidence of conflicts between users of OSCAR, two similar problems based on existing collections of artefacts could be given to the users: one problem to be solved with the assistance of the awareness support and one without. Comparing the metrics output from the two sessions, and the types of artefacts created and modified to solve the problem will show how the addition of awareness to OSCAR affects the users of the system and the quality of artefacts they develop. The intention is to compare code quality, both by simple automatic metrics such as defect level and by inspection of the finished systems to obtain a qualitative judgement of their soundness. As a second stage, the GENESIS project's industrial partners will be asked to examine the awareness extensions for application in their work.

6 Summary, Open Issues, and Future Work

An approach to adding asynchronous workspace awareness to an existing software artefact repository has been proposed. The elements of "OSCAR Awareness" from the Workspace Awareness Framework[13] are *Presence*, *Activity Level*, *Actions*, *Changes*, *Objects*. The other elements of this framework rely either on exclusively synchronous shared workspace sessions or on process information which OSCAR does not directly support.

Many existing "workspace awareness" systems such as TUKAN[19], ShrEdit and SPE/JViews, including the current prototype intended for OSCAR rely on the system designers anticipating every way of changing the data and providing for it. They do not cope well with deus-ex-machina changes to the contents of the workspace. In OSCAR's case, it is possible to manipulate the CVS repository directly using a standard CVS client, potentially confusing the OSCAR system which at present operates on the assumption that all actions on artefact data held in CVS will be performed by it alone. Addressing this requires two changes, firstly that OSCAR itself can cope to a certain extent with data modifications performed by other means and secondly that awareness collection is not affected by unexpected data changes. "Informal awareness" systems like the Radar View do not have such issues as they do not deal in specific actions. This flexibility must be balanced against the loss of the complete "awareness picture" available in more restrictive systems.

A possible way of providing task/concept awareness by using the GENESIS platform's work-flow system was mentioned above; however other approaches that do not rely on any external tool must be examined as a part of any future work. One such awareness model is implemented in the TUKAN system which provides task awareness in the software development domain with the intention of finding suitable development partners to work with. Task awareness derived from a work-flow system will be limited to tasks directly managed by that system and will not include awareness about tasks taking place outside a predefined process or during deviations from a predefined process.

Alongside the architectural changes proposed in the approach discussion, we must find a suitable distribution model which supports awareness across a network of OSCAR systems without excessive communications overhead or data loss.

References

[1] Virtual network computing. [@:] http://www.realvnc.org.

[2] L. Aversano, C. Aniello, P. Gallucci, and M. L. Villani. Flowmanager: a workflow management system based on petri nets. In *Proc. of the 26th Annual International Computer Software and Applications Conference, COMPSAC02*, pages 1054–1059, Oxford, England, August 2002. IEEE Computer Press.

[3] S. Bowen and F. Maurer. Designing a distributed software development support system using a peer-to-peer architecture. In *Workshop on Cooperative Supports for Distributed Software Engineering Processes (with COMPSAC 2002)*, pages 1087–1092, Oxford, England, August 2002. IEEE Computer Society Press.

[4] D. G. Boyer, M. Cortes, J. Herbsleb, and M. J. Handel. Virtual community presence awareness. *ACM SIGGROUP Bulletin*, 19(3):11–14, 1998.

[5] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith. Efficient filtering in publish-subscribe system. In *Proc. of the International Conference on Software Engineering (ICSE01)*, pages 443–452. ACM Press, New York City, May 2001.

[6] J. D. Campbell. Characteristics of group development of diagrams. In *Proc. of the Eleventh International Workshops on Enabling Technologies for Collaborative Enterprises (WETICE02)*, pages 29–34, Carnegie Mellon University, Pittsburgh, June 2002. IEEE Computer Society Press.

[7] L. E. Damianos, J. Drury, T. Fanderclai, L. Hirschmann, J. Kurtz, and B. Oshika. Evaluating multi-party multimodal systems. In *Proc. Of the 2nd International Conference on Language Resources and Evaluation (LREC2000), Athens*, MITRE Corporation, 202 Burlington Road, Bedford, MA01730 USA, May 2000.

[8] P. Dourish and V. Belotti. Awareness and coordination in shared workspaces. In *ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, pages 107–114, Toronto, Ontario, November 1992. ACM Press, New York City.

[9] J. Drury and M. G. Williams. A framework for role-based specification and evaluation of awareness support in synchronous collaborative applications. In *Proceedings of the 11th International Workshops on Enabling Technologies for Collaborative Enterprises (WETICE02)*, pages 12–17, Carnegie Mellon University, Pittsburgh, June 2002. IEEE Computer Society Press.

[10] P. Fenjam, E. Kirda, S. Dustdar, H. Gall, and G. Reif. Evaluation of a publish/subscribe system for collaborative and mobile working. In *Proc. of the Eleventh International Workshops on Enabling Technologies for Collaborative Enterprises (WETICE02)*, pages 23–28. IEEE Computer Society Press, June 2002.

[11] M. Gaeta and P. Ritrovato. Generalised environment for process management in cooperative software engineering. In *Workshop on Cooperative Supports for Distributed Software Engineering Processes (with COMPSAC 2002)*, pages 1049–1053, Oxford, England, August 2002. IEEE Computer Society Press.

[12] J. Grundy, J. Hosking, and W. B. Mugridge. Inconsistency management for multiple view software development environments. *IEEE Transactions On Software Engineering*, 24(11):960–981, November 1998.

[13] C. Gutwin and S. Greenberg. Workspace awareness for groupware. In *CHI Conference Companion*, pages 208–209, 1996.

[14] C. Gutwin, S. Greenberg, and M. Roseman. Workspace awareness support with radar views. In *CHI Conference Companion*, pages 210–211, 1996.

[15] W. C. Hill and J. D. Hollan. History-enriched digital objects. In *Proc. of the ACM Conference on Computers, Freedom and Privacy CFP'93*, pages 9.16–9.20. ACM Press, New York City, 1993.

[16] W. C. Hill and J. D. Hollan. History-enriched source code. Unpublished manuscript, August 1993.

[17] T. Nomura, K. Hayashi, T. Hazama, and S. Gudmundson. Interlocus: Workspace configuration mechanisms for activity awareness. In *Proc. of the ACM Conference on Computer Supported Cooperative Work (CSCW'98)*, pages 19–28. ACM Press, New York City, November 1998.

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

[19] T. Schümmer. Lost and found in software space. In *The 34th Annual Hawaii Internationl Confernece on System Sciences (HICSS01)*. GMD-IPSI, IEEE Computer Society Press, 2001.