Sixth International Workshop on Collaborative Editing Systems, CSCW 2004, Chicago, November 6, 2004.

Adding Group Awareness to Design Tools using a Plug-in, Web Service-based Approach

Akhil Mehra[1], John Grundy[1, 2], John Hosking[1]
Department of Computer Science[1], Department of Electrical and Computer Engineering[2]
Private Bag 92019, University of Auckland, Auckland, New Zealand
ameh010@ec.auckland.ac.nz, {john-g, john}@cs.auckland.ac.nz

ABSTRACT
Group awareness provides members of a team with relevant knowledge of other's activities to enable them to make informed decisions with regards to their future work. Providing group awareness in collaborative editing tools is important to enhance a team's efficiency and effectiveness. We describe a new approach to adding group awareness facilities to an existing single-user editing application using plug-in components and Web Services. We have added group awareness facilities as a component based plug-in to Pounamu, a meta-CASE tool. We have used this plug-in, web service based system to enhance Pounamu's existing collaborative editing facilities. We describe the architecture of our approach, key design and implementation issues, illustrate its feasibility and evaluate its effectiveness

Keywords
CSCW, group awareness, collaborative design, software plug-in, Web services-based architecture, meta-CASE tool, visual design environments.

1. INTRODUCTION
Group awareness can be defined as "an understanding of the activities of others, which provides a context for your own activity" [3]. Group awareness is of great importance when collaboratively editing documents. Knowledge of one another's presence, actions and intentions vastly improves efficiency while developing documents collaboratively.

Computer-Aided Software Engineering (CASE) tools are frequently used to automate, administer and simplify the development process for software [1]. We believe that it is important to add collaborative editing and group awareness facilities to CASE tools in order to provide efficient collaboration during the administration of a project. We aim to add group awareness facilities to the Pounamu meta-CASE tool which enables a user to specify and generate multi-view visual design tools[18]. Using Pounamu a user is able to create visual tool specifications and then use these tools to model existing projects. Pounamu was developed as a single-user application but provides collaborative editing facilities via plug-in components [10]. However it does not provide any group-awareness facilities.

One of our major goals for the Pounamu meta-CASE tool is to enable groups of people to work effectively when collaborating on development of artifacts. In the work presented we wanted to add group awareness facilities to Pounamu in order to make users more knowledgeable of each others identity, actions, and intentions. Group awareness is provided for both synchronous and asynchronous collaboration in Pounamu. Visual cues are used to indicate a user's presence, actions and intensions. Group awareness in synchronous mode is achieved by plugging in the group awareness component into an existing collaborative editing environment developed for Pounamu. Group awareness in asynchronous mode is facilitated by integrating a version control system, a visual differencing tool and the group awareness component developed. Each user is expected to check in his/her work into a version control repository. With the help of the visual differencing tool users are able to visually differentiate their work with existing versions of design diagrams in the repository and see the comparison.

In pursuing this work we have developed a proof-of-concept approach to support the development of group awareness applications using Web services-based technology. This explores the feasibility of using a Service Oriented Architecture and Web services for realising a range of group awareness facilities in CASE tools.

We first present the motivation for our work and discuss related research. We then outline our approach of using Web Services to provide collaborative editing in Pounamu. We describe key design and implementation issues and provide an evaluation of our approach. We conclude with a summary of the contributions of our work and directions for future research.

2. MOTIVATION
Adding collaborative work support to an application includes the ability of a user to collaboratively edit artifacts, version edited information for asynchronous work support and provide "group awareness" i.e. providing information to other users so that they can make informed decisions with regards to their future work [8]. Pounamu already possess collaborative editing facilities via a set of plug-in components [10]. In order to provide a whole set of tools for collaborative work support for Pounamu we have enhanced Pounamu by providing a version control mechanism and group awareness facilities.

A number of tools have been built that support group awareness for collaborative editing systems. Examples include Quilt[6], GROVE[4], PREP[12], SASSE[2], Calliope[11] and Alliance[15] collaborative writing systems. Most of these tools have been built using either ad-hoc approaches (no specialized tool kit or component) or using a specialized groupware tool kit. The tools thus have fixed, hard-coded group awareness support. Most such applications suffer from a lack of extensible groupware facilities and lack of ability of users to configure these facilities at run-time [8]. Ideally users should be able to determine the kinds of groupware facilities they want to use while a tool is in use and have these incrementally added to their design environment on-demand.

Most existing groupware applications use TCP/IP or remote object technologies such as CORBA to achieve data transfer [13,

16] We have found that these technologies make building groupware infrastructure challenging and difficult to evolve [10], as the remote object interfaces and data protocols are not amenable to extension and evolution. They are also not easy to use to build dynamic, run-time discoverable and deployable distributed work facilities. Web services technologies offer more extensible and dynamic run-time support mechanisms for collaborative work infrastructure[9, 14]. Initial prototypes of collaborative editing tools have been successfully developed with web services [10, 17], and we wanted to apply the same technique to group awareness facilities.

We have been developing Pounamu, a meta-CASE tool for specification and generation of multiple view visual tools [18]. The tool enables a user to rapidly specify "visual notational elements, underlying tool information model requirements, visual editors, the relationship between notational and model elements, and behavioral components" [18]. Tools are generated on the fly and can be used for modeling immediately. Figure 1 illustrates a Pounamu-developed UML tool in use.

We wanted to provide group awareness facilities for any such Pounamu tool in a plug-in component based manner using a Service Oriented Architecture. Additionally we wanted the group awareness component to support both synchronous and asynchronous awareness.

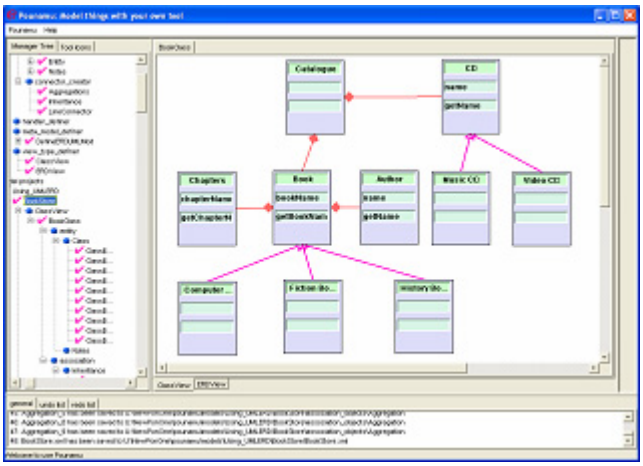

Figure 1. Example of Pounamu-built UML tool in use.

3. OUR APPROACH

The main aims for group awareness extensions to Pounamu include:

- Adding group awareness feature using a dynamic plug-in mechanism [8].
- Appropriate visual cues to effectively provide group awareness information in a non intrusive way in a meta-CASE tool environment
- Approaches to effectively provide group awareness in a multi-view application
- The viability of using a CVS repository to provide effective asynchronous group awareness information

In order to realize all of our group awareness facilities we have extended an existing Web Service based architecture that utilizes the principles of Service Oriented Architecture. The primary reason for reusing the above mentioned approach is the success when using this approach while developing our collaborative editing component [10]. Web services allow run-time discovery of multiple collaboration services, dynamic integration with the services, and use of XML-based technologies to facilitate data exchange between tools.

Figure 2 illustrates our approach to building a group awareness component. Firstly we developed a group awareness component that was plugged into our existing collaborative editing infrastructure. The group awareness component is responsible for providing group awareness related information for a Pounamu command. Pounamu commands are a set of API's that describe all editing events that take place in Pounamu model project. All user interaction with a specific Pounamu model project can be formulated into a Pounamu command. A user's interaction with a Pounamu model project generates Pounamu events (1). These events are captured by a collaborative editing component and appropriate web service calls are generated to notify remote collaborators of these events (2). The web service passes on these events to the remote collaborative editing component with the help of RMI calls (3, 4). The collaborative editing component is responsible for generating appropriate Pounamu commands that describe these events and then passing these commands to the group awareness component (5). The group awareness component decorates the commands with appropriate group awareness information e.g. user, time/date, the view the edit was performed in, etc. These commands are then executed and displayed in the remote user's environment with appropriate group awareness decoration.

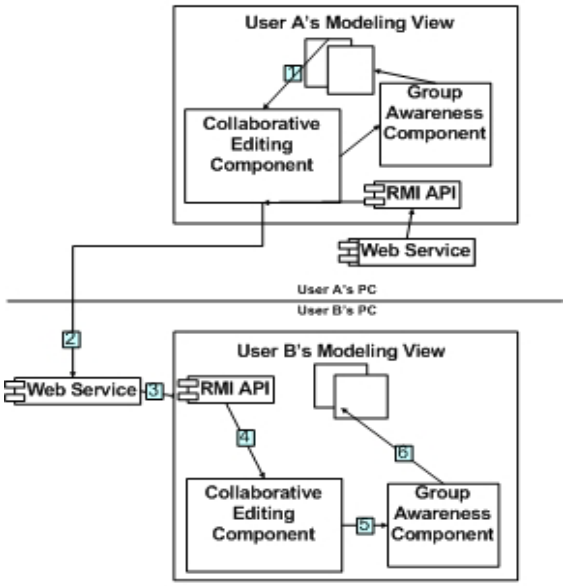

Figure 2 Overview of our approach.

We further extended Pounamu to access a version control system, Concurrent Versions System (CVS) [7], using a Web Service interface. Additionally we developed a visual differencing tool for Pounamu to help distinguish between two different versions of a project. Figure 3 shows how the visual differencing and access to a version control system integrates into our existing system.

The Pounamu differencing tool component compares an existing model project with an earlier version accessed from a CVS repository and generates appropriate Pounamu Commands to

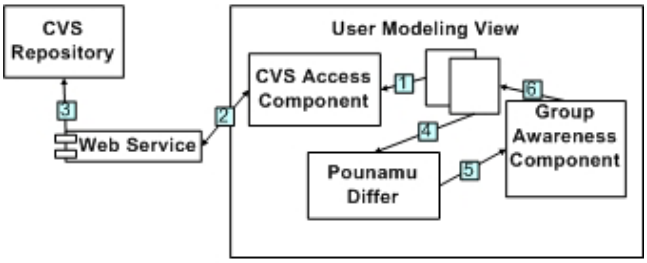

Figure 3 Our Approach to adding asynchronous group awareness.

update(4). These commands are then passed on to the group awareness component (5) to be decorated with appropriate visual cues and displayed to the user (6).

4. ARCHITECTURE

In previous work we developed a collaborative editing plug-in to provide collaborative editing support for any Pounamu specified tool. The plug-in may be discovered and loaded at runtime when the user decides they want collaborative editing support. It uses web services technology for discovery and communication. This collaborative editing component uses a peer-to-peer architecture as show in Figure 3. One of the main goals while developing our new group awareness component was to develop the component in a similar plug-in manner to support runtime loading and unloading of the component. The group awareness component is responsible for decorating Pounamu commands with appropriate group awareness information.

We have further added CVS repository access by developing a Web Service interface to access the repository. The Web Service interface to CVS uses Web Service attachments to achieve the required functionality. In order for a user to be able to view differences in various versions of a Pounamu model project checked in we developed a visual differ component for Pounamu. Both collaborative editing component and the visual differencing tool use the group awareness component to display differences. As User A edits a model project he/she is working on, appropriate events are propagated to the collaborative editing component (1). User A's collaborative editing component calls appropriate remote web services hosted by User B(2) to notify User B of editing event. User B's Web Services with the help of RMI call notifies its corresponding collaborative editing component of remote editing events (3). The collaborative editing component generates Pounamu commands and passes these on to the group awareness component (4). The group awareness component decorates these with awareness information and displays them in the Pounamu Meta tool (5).

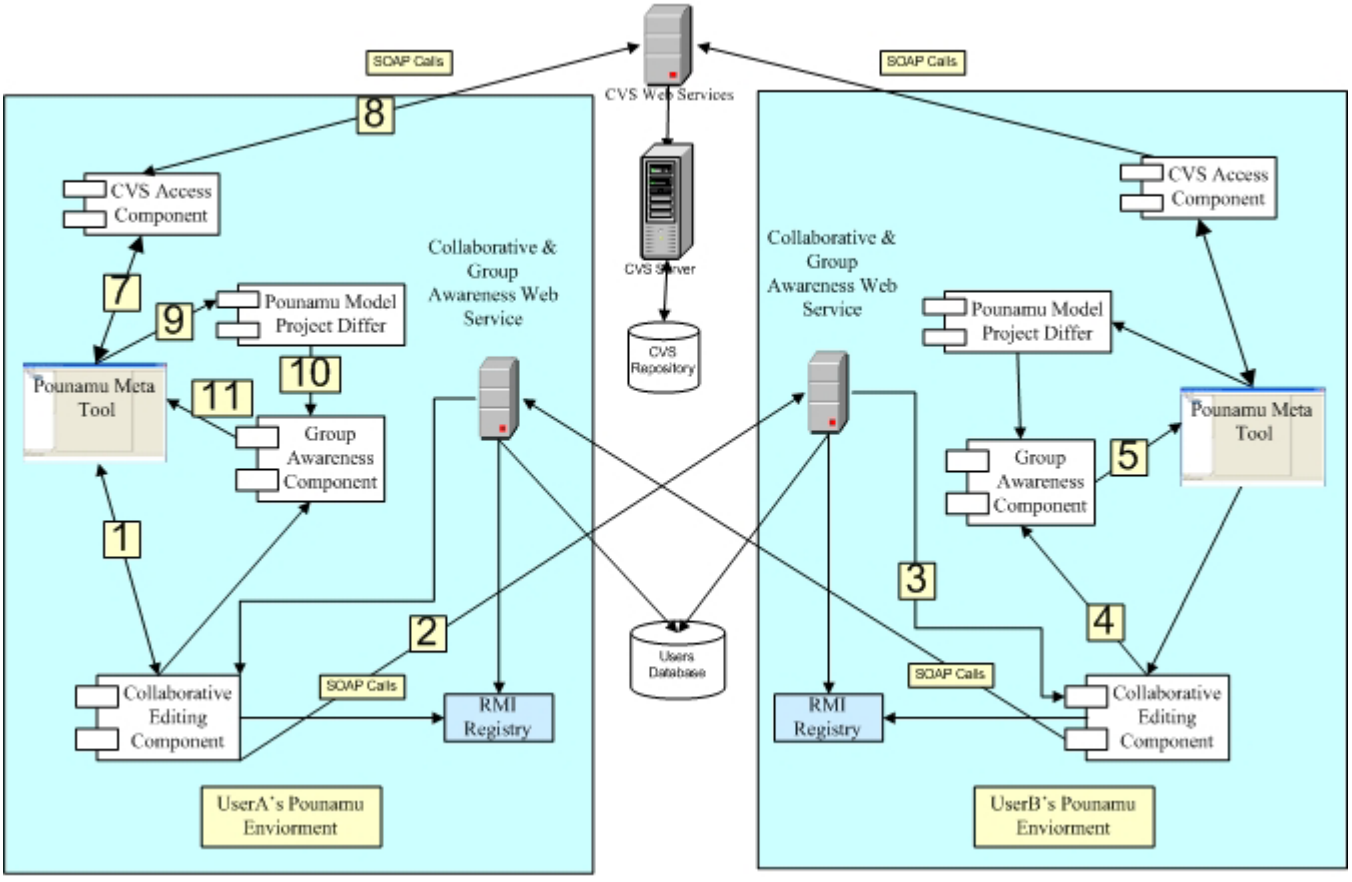

Figure 4 Overview of our Web Service Based architecture

The main components that are used when retrieving and differentiating project with earlier versions of the project include the Pounamu Model Differ component, CVS Access component and the CVS web service. Suppose UserA wishes to retrieve an earlier version of a UML model project that he and his colleagues have been working on. UserA will specify his wish to retrieve an older version of a Pounamu Model project. This request will be passed on to the CVS Access component (7). The CVS access component will use the CVS web service to retrieve an earlier version of the project (8). Once the model project has been retrieved the Pounamu Model Project differ will be called to differentiate between the retrieved version and the current version (9). The Pounamu differ will generate appropriate Pounamu commands that signify model project differences. These commands will be passed on to the Pounamu Model Project differ and then will be passed on to the Group Awareness Component (10) that will decorate the commands appropriately. Change will be displayed to the user (11).

5. EXAMPLE USAGE

In this section we describe an example usage of our Pounamu plug-in group awareness component. Consider the following scenario. Three colleagues John, Tim and Tom are working on an UML (Unified Modeling Language) based model project created in Pounamu from geographically disparate locations. They have all enabled collaborative editing in synchronous mode and are thus able to see each others changes in real time.

All users have the option of enabling an awareness panel. The panel gives users information about other online users and a history of collaborative editing events that have take place. Considering Pounamu is a multi-view visual tool we considered it important to display inter-view awareness. One way of achieving this is by providing users with an indication of other users intentions. This was achieved by coloring the user names of online users with respect to the project and the view they are working on. For example: a user name colored in red denotes that a user is working in the same view as we are. Where as blue denotes that a user is working on a different project.

Let us suppose that John decides to add two classes and an inheritance connector between them. Figure 5 (a) illustrates group awareness information presented to Tim and Tom as a result of John's actions. Both remote users see the added classes and connectors highlighted. This helps users distinguish between remotely added shapes and connectors and locally added ones.

We believe that in synchronous mode remotely added shapes and connectors should merge into existing diagrams over a period of time if a remote user does not explicitly reject the shape. This is done by "fading away" group awareness information. For example, in Figure 5 (a) the highlighting of one shape has become lighter than the other. Over a five minute

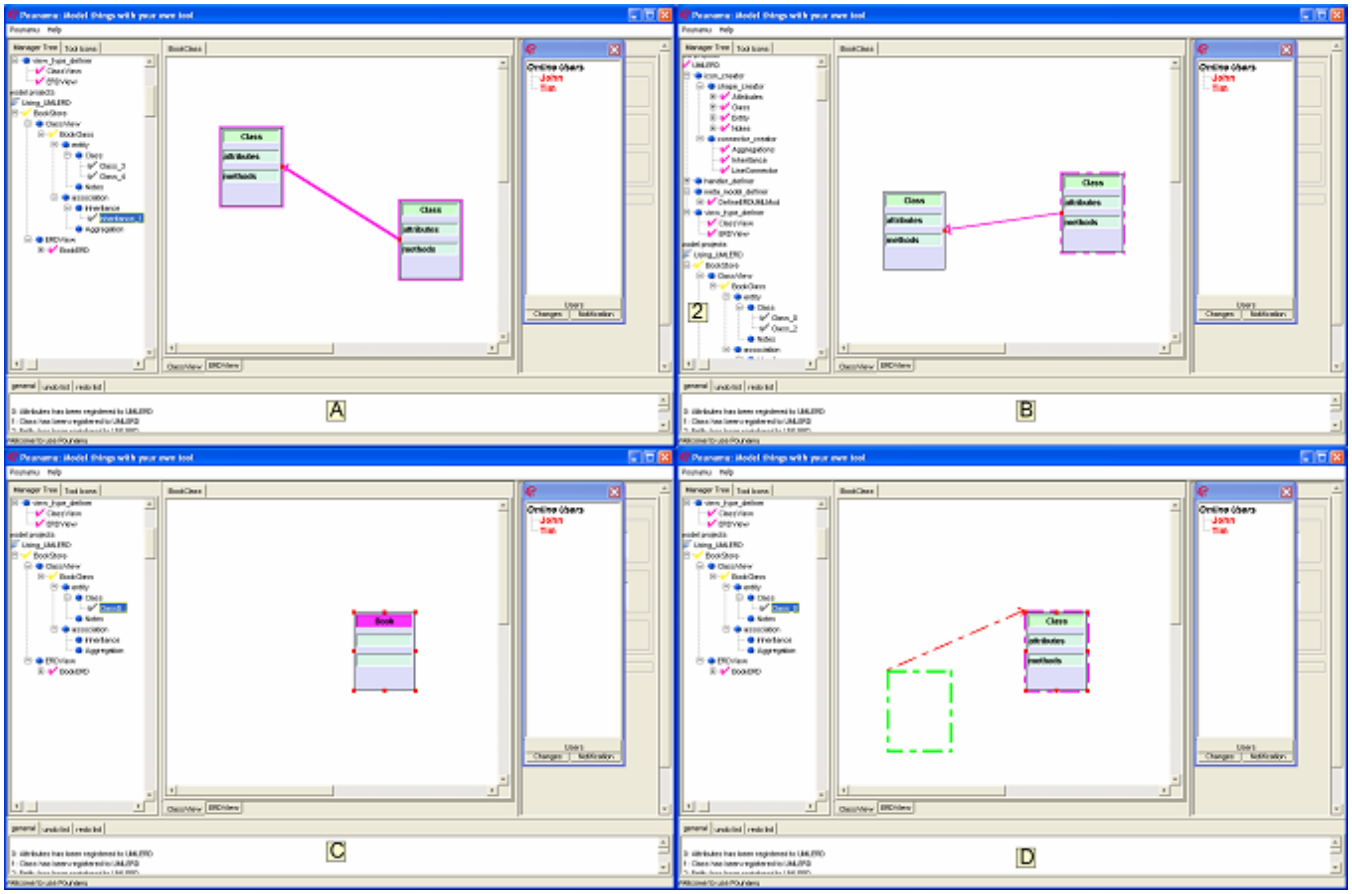

Figure 5 Group Awareness Examples.

period remotely added shapes automatically merge into the existing diagram and this is depicted to users by total fading away of group awareness information. Figure 4 also give us examples of other type of group awareness information presented to remote users such as deletion (Figure 5 b), movement (Figure 5 d) or changes in properties (Figure 5 c). Changes in properties are shown by highlighting the property changed.

Now let's consider a scenario where John, Tim, and Tom cannot continue to edit the document synchronously due to individual commitments or network failure/unavailability. Let us suppose that Tim and Tom leave the joint editing session while John continues to edit artifacts in a UML Model project. After John is done he checks his version into a CVS repository. Tim later wishes to update his model project by differentiating between his version and the version checked in by John.

This can be achieved in Pounamu with the help of the visual differencing tool we have provided, again using a plug-in component architecture. Tim logs on to the CVS server and gets a list of prior versions checked in. This is done with the help of the CVS configuration dialogue box a shown in Figure 6.

Using this dialog box a user is able to view various versions of model project checked in and differentiate there versions with these. Tim can select a version checked in by John and press the differentiate button. This will show Tim differences in a visual manner as show in Figure 7. Tim may wish to keep some changes while rejecting others. Once done Tim can check in his new version and others are able to view his changes. Development may continent in an iterative manner.

The collaborative editing facilities added with group awareness features in Pounamu enable user to collaborate easily and effectively. By providing a user both synchronous and asynchronous group awareness facilities we have provide users with a rich set of cues to understand user intention while modeling using the Pounamu meta-tool.

Figure 6 CVS configuration Dialogue Box.

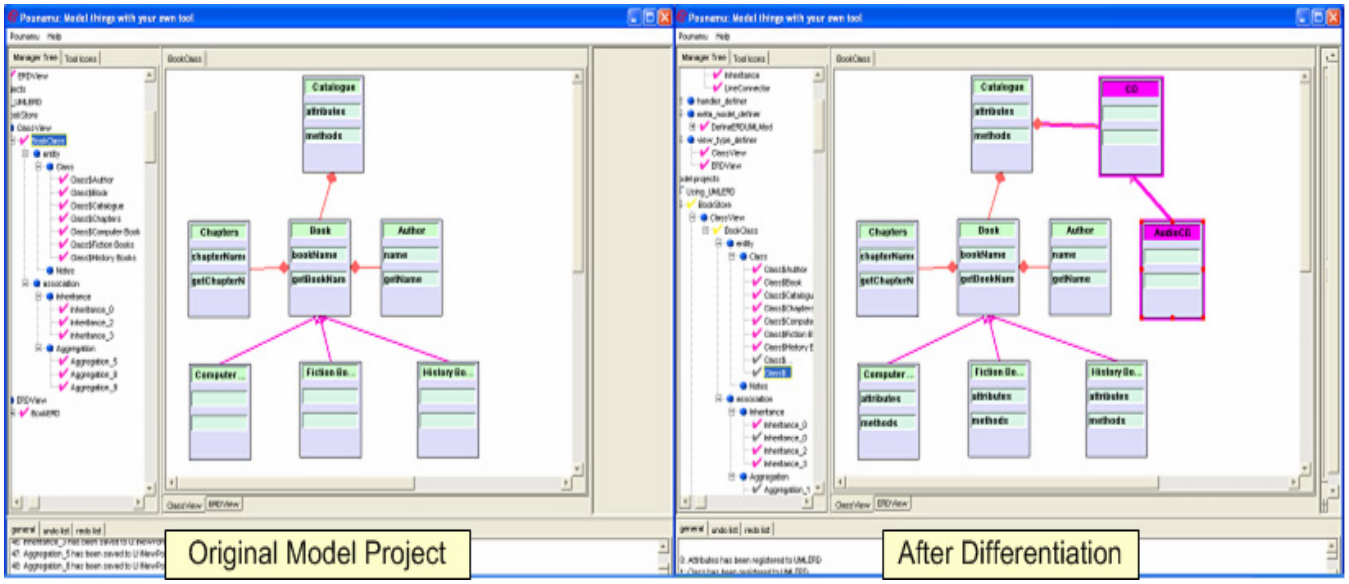

Figure 7 CVS Visual Differentiation Example.

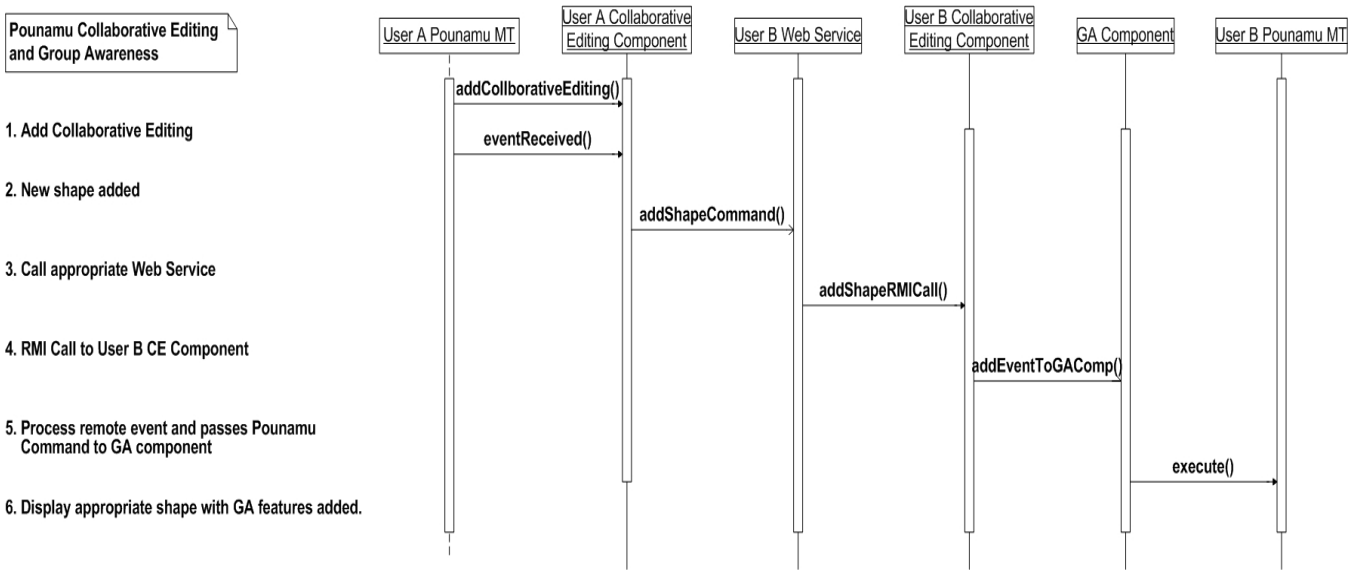

Figure 8 Sequence Diagram for a typical interaction with a Group Awareness Component.

6. DESIGN AND IMPLMENTATION

One of the major design goals was to make no changes to the existing single-user Pounamu code when adding collaborative editing and group awareness capabilities. The group awareness component was added in as a plug-in [8]. Additionally we expanded our existing Service Oriented Architecture to include a version control system and provided a Web Service interface for CVS. Doing this made the access to version control integration and evolution consistent with the approach used for synchronous exchange of editing events.

Providing a Web Service interface for a CVS sever turned out to be quite straight forward. The web service developed is an adaptor to an existing java CVS client. In order to send and receive files from the CVS repository via web services we are using Web Service attachments. The web services developed take CVS related parameters, CVS commands and appropriate files if required. The web service returns the status of execution that has taken place.

The sequence diagram in Figure 8 describes the typical flow of events between various objects in a collaborative editing situation. Typically we will have two users, User A, User B. Let assume that User A initiates collaborative editing. This will result in the instantiation of the collaborative editing component. Now lets assume that User A adds a shape to his model project. This will result an AddShapeEvent being generated and passed to the collaborative editing component via the eventReceived() method. The Pounamu Command will call UserB's web service and pass on the details of the event via the addShapeCommand(). UserB's web service will make an RMI call to UserB's collaborative editing component using the addShapeRMICall() method. UserB will generate an AddShapeCommand and pass it on to the group awareness component via the addEventToGAComp() method. The group awareness component will decorate the component with appropriate group awareness information and execute the command. The shape will be added to UserB's Pounamu model project.

The Pounamu meta-CASE tool was developed using the Java platform, thus we chose Java and related technologies to implement our group awareness and collaborative editing features. JAX-RPC (Java API for XML-based Remote Procedure calls), JAVA Web Service attachments and Java RMI are some of the technologies used to implement our group awareness features. Additionally we used Javacvs [5]a CVS client built in Java to help with the development of a Web Service interface for CVS. Currently we are using CVSNT as our CVS server.

7. DISCUSSION & CONCLUSION

We have provided groupware support for Pounamu a meta-CASE tool in the form of collaborative editing and group awareness plug-in capabilities. The group awareness plug-in developed provides both online and offline awareness for any Pounamu-specified visual modeling tools. This has been used to date on ER diagramming tools, UML tools and semantic modeling tools. The group awareness component designed is extremely flexible in that it provides group awareness for any tool that can be defined with the Pounamu meta-CASE tool. The flexibility of the group awareness component is further highlighted by the fact that it provides awareness for both synchronous collaborative editing as well as asynchronous collaboration based on version control changes.

The construction of a Web Service for CVS access has enabled us to evolve our group service enhancement in a Service Oriented manner. The main advantage of providing a Web Service based interface for CVS is the ability for a heterogeneous set of application to interact with a version control system in a consistent manner. We envision a number of varied applications accessing Pounamu files vie a CVS repository.

Key future research includes improving the configurability of the group awareness components by end users. In addition, we want to make the different group awareness capabilities currently

supported discrete separate plug-ins themselves. This will allow users to choose between a range of different awareness capabilities and have each individually integrated within a Pounamu tool as required. We are investigating using AI techniques to provide more informative group awareness information in a non-intrusive manner by examining event histories and presenting other users with summaries of another's intensions. We would also like to apply our group awareness capabilities to the web based diagramming plug-in component that we have also developed for Pounamu.

8. REFERENCES

[1] CASE Webopedia, Jupitermedia Corporation, Darien, CT, 2003, Webopedia is a free online dictionary for words, phrases and abbreviations.

[2] Baecker, R.M., Nastos, D., Posner, I.R. and Mawby, K.L. The user-centered iterative design of collaborative writing software. Human Factors in Computing Systems. INTERCHI '93. IOS Press. 1993. 399-405.

[3] Dourish, P. and Bly, S. Portholes: supporting awareness in a distributed work group. CHI *'92 Conference Proceedings. ACM Conference on Human Factors in Computing Systems*. Striking a Balance. ACM Press. 1992. 541-547.

[4] Ellis, C.A., Gibbs, S.J. and Rein, G.L. Groupware: some issues and experiences. Communications of the ACM, 34 (1). 39-58.

[5] Entlicher, M. Javacvs, 2001, Javacvs module for Netbeans.

[6] Fish, R.S., Kraut, R.E., Leland, M.D.P. and Cohen, M. Quilt: a collaborative tool for cooperative writing. Sigois Bulletin, 9 (2-3). 30-37.

[7] Fogel, K., Bar, M. and ebrary Inc. Open source development with CVS. Coriolis Group Books, Scottsdale, AZ, 2001.

[8] Grundy, J. and Hosking, J. Engineering plug-in software components to support collaborative work. Software-Practice & Experience, 32 (10). 983-1013.

[9] Kafeza, E., Chiu, D. and Cheung, S.C. Alert-Driven Process Integration in a Web Services Environment. *In Proceedings of the 1st International Conference on Web Services*, Las Vegas, USA, June 23-26 2003.

[10] Mehra, A.G., J.C. and Hosking, J.G. Supporting Collaborative Software Design with a Plug-in, Web Services-based Architecture. *In Proceedings of the ICSE 2004 Workshop on Directions in Software Engineering Environments*, Edinburgh, Scotland, IEE Press.

[11] Mitchell, A. Communication and Shared Understanding in Collaborative Writing, University of Toronto, Toronto, Canada, 1996.

[12] Neuwirth, C.M., Kaufer, D.S., Chandhok, R. and Morris, J.H. Issues in the design of computer support for co-authoring and commenting. CSCW 90 Los Angeles. Proceedings of the Conference on Computer-Supported Cooperative Work. ACM. 1990. 183-195.

[13] Pendergast, M.O. and Vogel, D. Design and implementation of a PC/LAN-based multi-user text editor, *In Proceedings of IFIP WG 8.4 Conf. on Multi-User Interfaces and Applications*, North-Holland, September 1990, 195–206.

[14] Pokraev, S., Koolwaaij, J. and Wibbels, M. Extending UDDI with Context-Aware Features Based on Semantic Service Descriptions. *Proceedings of the 1st International Conference on Web Services*, Las Vegas, USA, June 23-26 2003.

[15] Salcedo, M.R. and Decouchant, D. Structured cooperative authoring for the World Wide Web. Computer Supported Cooperative Work, 6 (2-3). 157-174.

[16] ter Hofte, G.H.a.H.J.v.d.L. CoCoDoc : A framework for collaborative compound document editing based on OpenDoc and CORBA. Open distributed processing and distributed platforms: *Proceedings of the IFIP/IEEE international conference on open distributed processing and distributed platforms*, Toronto, Canada, May 26-30, 1997. Chapman & Hall, 1997, pp. 15-33.

[17] Younas, M.a.I., R. Developing Collaborative Editing Applications using Web Services. *Proc. 5th Int. Workshop on Collaborative Editing, Helsinki*, Finland, Sept 15, 2003.

[18] Zhu, N., Grundy, J.C. and Hosking, J.G., Pounamu: a meta-tool for multi-view visual language environment construction, *In Proceedings of the 2004 International Conference on Visual Languages and Human-Centric Computing*, Rome, Italy, 25-29 September 2004, IEEE CS Press.