April 2007

Improving Collaboration with On-Line Meetings

Kerri Leigh Theriault
Worcester Polytechnic Institute

Sarah J. Pickett
Worcester Polytechnic Institute

Project Number: GFP0603

Improving Collaboration with On-Line Meetings

A Major Qualifying Project Report:

Submitted to the faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by:

_____

Kerri Edlund

_____

Sarah Pickett

Thursday, April 26, 2007

Approved:

_____

Professor Gary F. Pollice, Major Advisor

1. Eclipse Plug-in
2. ECF
3. Webfoot
4. Virtual Meeting

Abstract

With the rapid growth of collaborative development within the software engineering field, finding ways for teams to stay connected while working on a project is becoming more important. While there are many useful tools for these teams to utilize, it is often the case that one specific tool does not include all of the necessary features. This project builds upon the Eclipse Communication Framework (ECF) to add SourceForge capabilities useful to teams while in a virtual meeting.

Acknowledgements

There are many people that helped to make this project possible. First and foremost, we would like to thank Professor Gary Pollice, our advisor. Without his wisdom and guidance, this project may not have been possible. We thank him for his continuous support and input throughout the project. His comments and suggestions proved to be invaluable and a great aid to our end product.

We also wish to thank the other students working on Webfoot projects. Their outside view of our project allowed us to think outside the realm of a programmer. Their unremitting contribution of suggestions helped us achieve the best end product that we could.

Finally, we would like to thank the members of the Eclipse Communication Framework team for their support and interest in our project. We thank them for giving us the opportunity to add greater functionality to their product. The members, especially Scott Lewis, have been very helpful toward us when we had questions on the code that already existed and for his help we are grateful.

Table of Contents

List of Illustrations

1. Introduction

Within educational and industrial settings, people often need to work as a team to solve a common problem. Teams have difficulty maintaining clear, open lines of communication and assign tasks within the group. Collaboration tools, such as SourceForge, help manage group interactions. SourceForge provides groups with discussion boards, source code repositories, and various other ways to manage projects. Users can also create file releases without having to worry about version control. The discussion forum is helpful but does not provide a real-time atmosphere that group members can use to hold meetings when they are unable to meet in person. Although SourceForge is an exceptional organizational tool, it was not designed with the concept of a virtual meeting in mind. By expanding SourceForge's capabilities and integrating them into an existing chat room environment, this allows for real-time interaction between users that many development teams depend upon for communication.

Eclipse is a well known software development environment that allows programmers develop software in a variety of programming languages. The Eclipse Communication Framework was developed to provide a generic chat client communication platform, that lets Eclipse users hold virtual meetings between team members who are unable to meet in person. With the advent of this framework, teams are now provided with a reliable way to communicate and remain organized when the team is physically unable to meet.

WPI's Environment Built for Object-Oriented Teams (Webfoot) is an open source, multi-year project established to execute the integration of Eclipse with such tools as SourceForge, allowing easier communication and maintenance among groups. Through

Webfoot, Eclipse users can log into their SourceForge accounts, and access their projects with greater ease due to less context switches. After doing this, the user is provided with a multitude of options, allowing the user to update, view, and maintain the various aspects of their project. Previously completed Webfoot projects provide capabilities for users that include the ability to view, create, and delete tasks, tracker items, and user stories (a special tracker item) from within Eclipse. Our work in Eclipse contributes new capabilities to Webfoot.

The problem we sought to address was integrating SourceForge and Eclipse so that teams could communicate and stay organized at the same time. We incorporated tools that help users communicate in a chat room environment, capture minutes, and create action items (tasks) from the chat room. These capabilities included allowing a user to save the current chat to the discussion board of their project, open a chat that was previously posted on the discussion board, and create a new task directly from the current chat. When groups are able to manage task assignments and communication within a single program, they can reduce the amount of confusion, time spent, and delays in getting results. There is no longer a need to hold a meeting, only to go back later and create the tasks and type up the meeting minutes. By holding a "virtual meeting", i.e. meeting in a chat room, all group members do not have to be in the same place at the same time. No important thoughts are lost in translation, since the full transcript of the chat meeting can be saved to the discussion board and viewed at a later date.

The remainder of this report contains sections on background, methodology, results and analysis, future work and conclusions, and any necessary appendices. The background is a description of Eclipse, Webfoot, and collaborative development. The

methodology contains a detailed description of our approach and a discussion of some of the problems we encountered. Our final results and analysis of our project comes next. We conclude with our suggestions of future work that might improve the Meeting Manager.

2. Background

Software tools have been around since early programmers in the 1950s began to utilize them to create, debug and maintain other applications and programs. Early software tools included linkers and loaders. UNIX helped tools become famous within the software development community in the 1970s with the creation of tools such as “grep”, “awk” and “make”. Tools were originally intended to be simple and lightweight; however, as time has progressed they have become part of more powerful environments known as Integrated Development Environments (IDEs). Eclipse is an IDE that can interface with other software and collaboration tools, including SourceForge, to aid in the growing field of collaborative software development. (Software Tools 2007)

2.1 Development Environments & Java

Java, like other programming languages, can be written using many different environments. It can be written using a simple editor, such as Emacs, or using an integrated development environment. Each environment has its advantages and disadvantages and the debate over which is more useful is often heated. Simple editors such as Emacs, JEdit, and Textpad aid users by providing parentheses matching, color coding and indentation. Unfortunately, simple editors are not particularly robust. They do not write code, integrate tightly with the compiler, or provide graphical java development tools. On the other hand, Integrated Development Environments (IDEs) are much more robust, with most allowing for visual Java development. They tightly integrate with the compiler or application server, providing debugging tools, allowing refactoring of code and handling version control. (Hall & Krasner)

2.2 Eclipse

Eclipse is an IDE that provides an environment enabling a user to utilize several development tools simultaneously. It provides teams with team-based tools allowing several team members to manage, work and organize a single project simultaneously. Eclipse can do this because it provides architecture and frameworks that support the creation of integrated tools. The architecture and frameworks allow users to create their own Eclipse tools which are known as plug-ins. (D'Anjou, et al.)

While Eclipse provides a solid framework for team development, it lacks a collaborative environment. The Eclipse Communication Framework (ECF) seeks to provide a framework that allows for the development of Eclipse-based tools which provide asynchronous point-to-point and publish-and-subscribe messaging. (ECF Website) Our project seeks to extend and build upon ECF to allow development teams to hold a virtual meeting, create SourceForge tasks, and save and manage meeting transcripts.

2.3 Collaborative Development

With development teams becoming further spread across continents, it is progressively more difficult to hold meetings. In response to this growing concern, the topics of collaborative software development (CSD) and open-source communities have begun to take shape. CSD and open-source overcome several limitations of traditional software engineering, providing a software engineering team with the following benefits:

- Instant project creation;
- Rapid requirement definition;
- Online design details;

- Small, agile teams for coding;
- Online reviews with instantaneous feedback.

Each of these elements is bound together by fact that every phase of development, from concept to release, is done by two or more individuals working together. Without the establishment of several Internet technologies, CSD and open-source communities would be nearly impossible. Internet tools, from email to network aware code repositories, have helped remove barriers that hinder the development process. As a result, developers are able to create better code more quickly.

As development teams became highly communicative and distributed an unintentional drift towards agile development methods occurred. As a general rule, open source development projects tend to "grow through small, incremental changes defined and executed by compact and highly communicative development teams." (Augustin et al.) However, with open source projects online, collaboration replaces the need for the direct interaction advocated by several agile methods, most notably the eXtreme programming principles.

CSD and open-source communities have several comparisons with the principles promoted by agile methods. Proponents of the agile methods often state "that software development is as much people-oriented as process-oriented." (Augustin et al.) The tools that are used with CSD were developed specifically to match the personality of the developers; the creation of those tools, which coincided with "the people-oriented aspect of CSD," is what generated the greatest amount of benefits.

2.4 SourceForge

SourceForge is web-based software developed and maintained by VA Software that allows users to set up projects and create central code repositories providing a centralized way to manage projects. When SourceForge was developed, it aimed to:

- Minimize administrative work;
- Maximize communication and collaboration;
- Preserve project knowledge;
- Make it easy to establish projects and recruit experts to them;
- Find and leverage existing code;
- Apply these concepts on a global scale.

The preceding goals were developed based upon extensive research conducted by VA Software that focused on the nature of open-source development communities and collaborative software development. VA Software aimed to enhance Internet based collaboration by providing SourceForge as a tool to accelerate and simplify the development process. SourceForge, unlike a loosely integrated development environment, provides Web-based assets and activity management tools supported by a centralized repository. As a whole, SourceForge allows a team to "improve collaboration, break down information silos and deliver solutions to customers faster via better information management and asset reuse." (VA Software 2005)

3. Methodology

The project focuses on understanding the problems that arise when a programming team is geographically diverse. Once these issues were understood, a solution was developed and implemented. Contained within Eclipse is the Eclipse Communication Framework (ECF), and although this is still under development, it is quickly becoming an essential tool for teams working on projects within Eclipse. More information about this tool can be found in Appendix A. This project is based on the concept of a virtual meeting, which is usually held in a chat client to enable project members to join from remote geographic areas. Through the integration of some SourceForge capabilities into Eclipse, users can access the necessary features without the use of a web browser.

A working knowledge of the underlying technologies is an integral part of the plug-in building process. Knowing the ins and outs of the existing framework which the project builds upon, and studying the SourceForge API are the key elements to the completion of this project. Before improving an existing plug-in, it is essential to learn exactly how the plug-in functions and how all of its pieces fit together. Since the goal of the project was to enhance the existing chat room environment, the first step was to learn to the existing framework. This learning process proved to be the majority of the time spent on the project.

3.1 Learning Existing Technologies

When building upon existing technologies and using existing technologies within a project, it is important to thoroughly understand the function and layout of the existing

framework or software. For this project, the SourceForge API and the existing Eclipse Communication Framework play an integral part in the integration of SourceForge capabilities into the framework. It was necessary to fully understand the framework before beginning work. Finding where to put the code for the features to be added can be a tricky task and can take a long time. The time spent learning the existing technologies can sometimes far outdo the number of hours spent on actually implementing the features to be added; this is what we discovered with our project.

3.1.1 Learning ECF

When adding features to an existing project, learning where to put the code for the features can prove to be a difficult task. Learning how the code works and how all of the pieces fit together is the first step that should be taken when building upon something that already exists. As the team found, this step can take far longer than implementing the additional features. The code added for the features could be minimal and take only a couple of days whereas learning where to put it and how to fit it into everything could take several weeks. Due to the extensiveness of the Eclipse Communication Framework, there were many pieces that were not necessary for the development of the new features. Learning which pieces were essential to the project took much time and effort since the entire framework needed to be understood before particular pieces could begin to be extracted. Once the necessary elements were extracted, the correct place in which to add the new features still had to be found.

After extracting the necessary plug-ins and projects of the framework, learning, more extensively, the way in which they worked was an essential part in finding where to place the code for the new features. Although time may have been lost on implementation due

to the time it took to find the correct class in which to place the new code, it was time well spent. Without taking the time to first locate the correct class, the code may have been placed in the incorrect class which would have resulted in a greater loss of time. As most know, trial and error is not usually the most efficient way at completing a task.

3.1.2 Learning the SourceForge API

Learning an existing API is not an easy task. Depending on the extensiveness of the software, APIs can be extremely intricate. However, they often contain very useful methods and extension points. There are many elements that need to be taken into consideration which may or may not be used when the new features are added. Not having a full understanding of the API can result in problems later in the project which may not have been accounted for when considering the amount of time the project would take to complete. Existing APIs can sometimes still be under development and can therefore be difficult to use. Using these APIs can be problematic, as the team found during development. This problem is discussed further in section 3.4.

To aid in the completion of the project, many of the functions provided by the SourceForge API were utilized. This made the integration of SourceForge capabilities into ECF nearly seamless. Learning the API and the functions which were needed took some time. The process did not take as much time as planned, however it was still time away from accomplishing the actual integration. Naturally, the knowledge of the API was necessary to facilitate the complete incorporation of the new features.

3.2 Design Decisions

Design decisions are an important part of all software projects. These decisions are primarily made in the beginning, but more refined decisions are made during the process of completing the project. Design decisions are difficult to make as both the programmer and end user need to be taken into consideration. Making the correct decision in the beginning can save a great deal of time throughout the development process. Development should not begin until a clear-cut design has been decided upon with the aim of minimizing changes that would have to be made during the development process. Although it may seem that time is lost in the beginning discussing all of the design issues, time will be saved in the long run since the design of the project will be straightforward at the beginning of the development phase.

During the span of the project, there were many design considerations that needed to be addressed. The problem of how and where to open an existing chat sparked much debate both among the group members and other members of the Webfoot development team. Whether or not to build a UI was a design decision that needed to be made before beginning the developmental stage of the project. The dilemma of where to save the current chat transcript was another design decision that was important to make prior to implementation.

3.2.1 Open Existing Chat

If an existing chat were to be opened in a new view, only the user who wished to open the chat would be able to see it. This does provide any benefit if the user wanted to open a specific chat to point out or comment on something that was said in a virtual meeting with other members. For this reason, it was decided that when a user wants to

open an existing chat, this chat will be opened within the current chat. The opened chat is displayed with the inclusion of a header and footer.

3.2.2 New Task Dialog

Another design decision that needed to be rethought was the creation a user interface (UI) enabling the users to create a SourceForge task directly from the chat. Although this feature is still a part of the end product, it was not designed by the team members. Since there was another group working on a Tasks View within Eclipse, they needed a way to create and edit tasks. To do this, they created a useful UI that was reused for the use of the Team Meeting Manager.

3.2.3 Save Chat

The original idea of saving the chat to the users' local workspace was not the design that was implemented. Doing so would result in other members of the team not being able to later view and/or comment on the meeting that took place. If a member or members were not able to attend the virtual meeting, they would not be able to later review the meeting that they missed since it would only be available in the workspaces of the users who were present and saved the chat. The ultimate decision was to save the chat to the SourceForge project as a new topic on the discussion board. This way, users can log into their SourceForge account, lookup the meeting on the Discussion Board, and review and comment on the virtual meeting. The contents of this post will be the exact transcript of the meeting, so nothing can be lost as would be the case if a single participant were to take meeting minutes.

3.3 Practices

The way in which the project is completed is also important to the timely and complete release of the product. When beginning a project, the time frame, features to incorporate and how to go about doing the implementation are important factors to discuss and lay out before any progress is to be made. Without defined goals and time frames in which to complete the goals, projects can easily get out of hand and become more stressful than necessary. If clear goals and deadlines are established before each stage of the project, it is more likely that the project will incorporate all of the requirements which were established and be completed on schedule.

3.3.1 Project Vision

After careful deliberation and several design reviews, a list of the capabilities to include in the project plug-in was created. The team decided that within the final deliverable product, users would be able to save the current chat, open a previously existing chat within the current chat, and create a task from the current chat. For more information on the requirements, a copy of the use cases can be found in Appendix B.

Although the functionality for the chat room and project selection were already incorporated into the Eclipse Communication Framework and SourceForge respectively, we still needed to implement all of the capabilities that were to become features and to determine how they would function once incorporated.

Considering the limited amount of time given to complete this project, it was difficult to include the extensive collection of features that may have been beneficial to other group members. The initial features that were thought to be useful to a team working on a project during a virtual meeting are the features included in the final

product. There was no need for refinement of the features to be included since the initial list was concise, well defined, and could be completed within the scope of the project.

3.3.2 Reuse

”Reuse of software assets, processes, knowledge, or code is desirable so that development time can be reduced and quality increased” (White). This project has encompassed this philosophy. Another important practice of software development and object-oriented design is encapsulation. By reusing code, this practice has also been adopted. These practices are important in that if something were to change, it only has to be changed in one particular place in the code rather than in multiple places where the code would be repetitive. This results in fewer mistakes and missing pieces in the final product.

Another important reason for reuse is to reduce the amount of implementation time. Since most all projects work on a deadline, time spent on the project is extremely valuable. To waste time rewriting a piece of code that already exists would be wasteful and could result in requirements or specifications left unimplemented or incomplete. By reusing elements or pieces of code already in existence, time is saved and can be spent on being sure that the final product is as complete as possible in the time frame given.

3.3.3 Iterative Development

Iterative development is a critical principle in the software development process. The “emphasis [is] on building releasable software in short time periods” (“Agile Software Development”). This principle is extremely important to projects where the time in which the project needs to be completed is short. This principle was closely

followed so as to not move on to a task before the previous one was completed. By following this convention, the team would be able to deliver a working product in the end, even if not all of the requirements had been included. Since no new task is started before the previous one is complete, this allows for less chance of incompleteness in the final product which helps to eliminate the likelihood of non-functional features within the software.

Using the tasks in SourceForge greatly helps in keeping with iterative development. By assigning specific tasks to team members to be accomplished within a particular iteration, staying on track with the project becomes simpler. Following and updating the tasks when in progress or complete allows the other team member to know where the project stands. This way, all members are on the same page of what is done, what needs to be done, and what someone working on completing a task may need assistance with to get the task completed by the end of the specified iteration.

3.4 Problems

Although the team was constantly testing the project, a number of obstacles were encountered along the way. The Eclipse Communication Framework that was being built upon was under continuous improvement. When the project began, the stable build was version 0.9.1. A month or two into the implementation, version 0.9.2 was released. With this new version, a number of packages had been refactored, many classes had been moved around, and even others were deleted. New stable builds were released at a faster rate than could be kept up with due to the scope of the project. In the end, version 0.9.3 was chosen as the version to in which the new features would be added.

Another problem encountered during testing was that the plug-in seemed to hang for a reason that was unbeknownst at the time. It was constantly "hanging" whenever the list of discussion forums associated with the user's currently selected project needed to be retrieved. This retrieval needed to occur in order to prompt the user to select the forum in which they wished to save their chat. This error had both the team and the advisor bewildered. Finding the problem took approximately two to three weeks. Once the problem was discovered, the confusion still continued. A call was made to a class called the IDiscussionAppSoap which is part of the SourceForge API (Application Programming Interface). For some reason, the plug-in did not like the instantiation of a new object of the type IDiscussionAppSoap in the class where it was called. Simply extracting the method from the class and moving it to another class solved the problem and everything worked fine.

3.5 Testing

Due to the short amount of time spent on a project at WPI, there are often things that need to be left out that one may feel are essential to the project, but simply cannot be done within the scope of the project. Although the best way to test a plug-in is with a JUnit test plug-in, one was not built for this project. A test plug-in had begun to be built, however, focus needed to be brought back to the coding of the features being added to ECF.

However, throughout the project, tests were run at every stage. Interacting with a plug-in is often a good way to be sure that it is working as expected. This method of testing was done whenever a feature was completed; sometimes even when the feature was thought to be working to a certain extent to be sure that development was heading

down the correct path. During this testing stage is where many mistakes were caught. There were times when the user was prompted to do something and even though they clicked cancel, they would be prompted to complete the next sequential task. Through these interactions, it was possible to discover that certain checks within the methods needed to be added to be sure that the user did make a selection and did not click cancel.

The plug-in was often run to ensure that items were happening in the correct sequential order. For instance, it would not make sense to prompt the user to select a project without checking and possibly prompted them to log into SourceForge first. There were times when it was realized that the user was being prompted for information out of sequential order.

This method of testing helped the team to think as not only the programmer, but also as a user. Asking questions, such as the following, about what the user would expect from the program proved invaluable during both the design and development stages of the project. How would the user expect the program to work? Is the plug-in intuitive? Does each of the functions do what the user would expect them to do? By taking on this double role of both user and programmer, the team feels that they were able to truly grasp the look and feel of the end product. Throughout the integration, weekly meetings were held with the advisor and other groups working on Webfoot projects. These meetings proved very helpful; they allowed for feedback on developmental issues from other people who would most likely be end users.

4. Results and Analysis

The final product of this project is not a fully working plug-in. At this point, the team is unable to produce a stable build of the project. There are errors that seem to appear spontaneously. The project is, however, functionally complete. The capabilities are fully implemented and work as the user would expect. Aside from implementing the project, a help manual has been provided as support to the user.

4.1 The Implemented Plug-in

In all, the end product is what was expected to be produced, aside from the errors. All of the functionalities that were originally planned to be included are incorporated in the end plug-in. Although some of the capabilities were not implemented in the way that was originally thought, after all of the design decisions that were made, the end result was implemented far better than the team imagined it would be in the beginning. As discussed earlier, in section 3.2, implementing the features in the way originally planned would not make sense or work out well for the end user.

As this project is a part of the Webfoot project, connection to the Meeting Manager has been included under the Webfoot Menu. Preferences have also been included as part of the Webfoot → SourceForge Preferences page. The end product is intuitive and easy to use. The user can find everything he or she needs within the right click menu of the chat room. The user is prompted from there to complete all of the steps necessary in order to complete the task that they set out to do. The order in which the user completes the steps is instinctive and sequential. They were put in the specific order

which they are in because it is the order in which the user would complete the steps on SourceForge to complete the same task.

All of the features that have been included in the implementation are found in the right-click menu while within the chat. The menu is shown in Figure 1. The last three menu items are the capabilities that have been added by this project. To accomplish this, action items were added to the already existing menu. Functionality was then added to the action items so that when the user clicks an option, the desired action occurs.

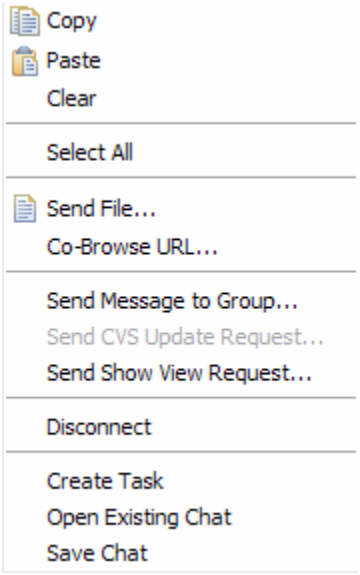


Figure 1: Chat room right-click menu

All of the capabilities added require the user to be logged into their SourceForge account and to have a project selected. Therefore, when a user selects an action from the newly added capabilities and is not logged into SourceForge or does not have a project selected, the plug-in prompts them to do so before they can continue. The check to see whether the user is logged in and has a project selected or not was easily implemented. The methods to do so were already provided in the Webfoot code, allowing the creation

of new instances of the objects and checking the particular settings of the object. The code can be seen below.

```
if (!SFConnector.getTheSFConnector().isConnected())

SFProject project = WebfootSFCorePlugin.getDefault()
            .getCurrentProject();
                  if(project == null)
```

Once the user enters the chat room, logs into their account, and selects a project, they can then use all of the features that we provide in our project. The team determined that it would be useful for users to be able to create tasks directly from the virtual meeting. If a person in the meeting had an idea of something that should be included in the project, someone who is logged into SourceForge via Eclipse can immediately create the task reducing the possibility of it being forgotten. When the user is in a virtual meeting, he or she can use the right click menu to create a task. The user will be prompted to select the Task Group that they wish their new task to become associated with. The selection can be made from a list of the Task Groups that exist within the selected project

Once the user selects the group that they want the new task to go into, the system presents a dialog to fill in all of the necessary information for their new task. The project utilized the dialog box that was created by another group working on an MQP.

Another feature of the final product is the ability for the user to save the chat. This is useful when keeping a log of all meetings held, since meeting minutes do not have to be taken because the meeting transcript can be saved directly. It is also useful to save an entire meeting transcript when someone cannot attend the virtual meeting. The saved meeting is entered into the SourceForge Discussion Board as a new topic and the missing member can read of the entire meeting and make comments as he or she sees fit.

When the user selects to save the current meeting, he or she will do so from the right click menu as show in Figure 1. As before, if the user is not logged into their SourceForge account or they do not have a project currently selected, they will be prompted to do so. Otherwise, they will be prompted to select the Discussion Forum which they wish the chat to be put into. Initially, the team wanted to give the user the option to create a new forum, but the current SourceForge API does not support this action. Once the user has selected the forum, he or she will be given the opportunity to enter the title for their new post. Error checking was included here to ensure that the user cannot submit a blank title or one that begins with a white space character. Below is the code that was included to ensure that the title the user entered is valid.

```
public void makePost()
{
	//Ask the user for a name for their topic

	InputDialog createTopic = new InputDialog(shell, "Create Topic",
	"Please enter a topic title for your post", "", null);

	createTopic.open();
	t = createTopic.getValue();


	//Be sure that the topic name entered is valid.  That it does not
	start with a space or is null.

	if(t.startsWith(" ") || t.equals(""))
	{
		MessageDialog.openError(shell, "Invalid Title", "The title
		you entered is not valid, please enter a new title.");

		if(currentForum != null)
		    makePost();
	}

}
```

This function is called recursively so that if the user enters a title that is not valid, the function is called again giving the user another chance to enter a title.

The final feature of this project was the ability for the user to open an existing chat. This is particularly useful when a user wishes to mention something from a previous meeting that is relevant to the current meeting. The way this feature is implemented this was to have the selected chat open in the current chat. The opened chat will be sent as a message to the chat room via the user who chose to open the chat. This may look something like the screen shot shown here in Figure 2.

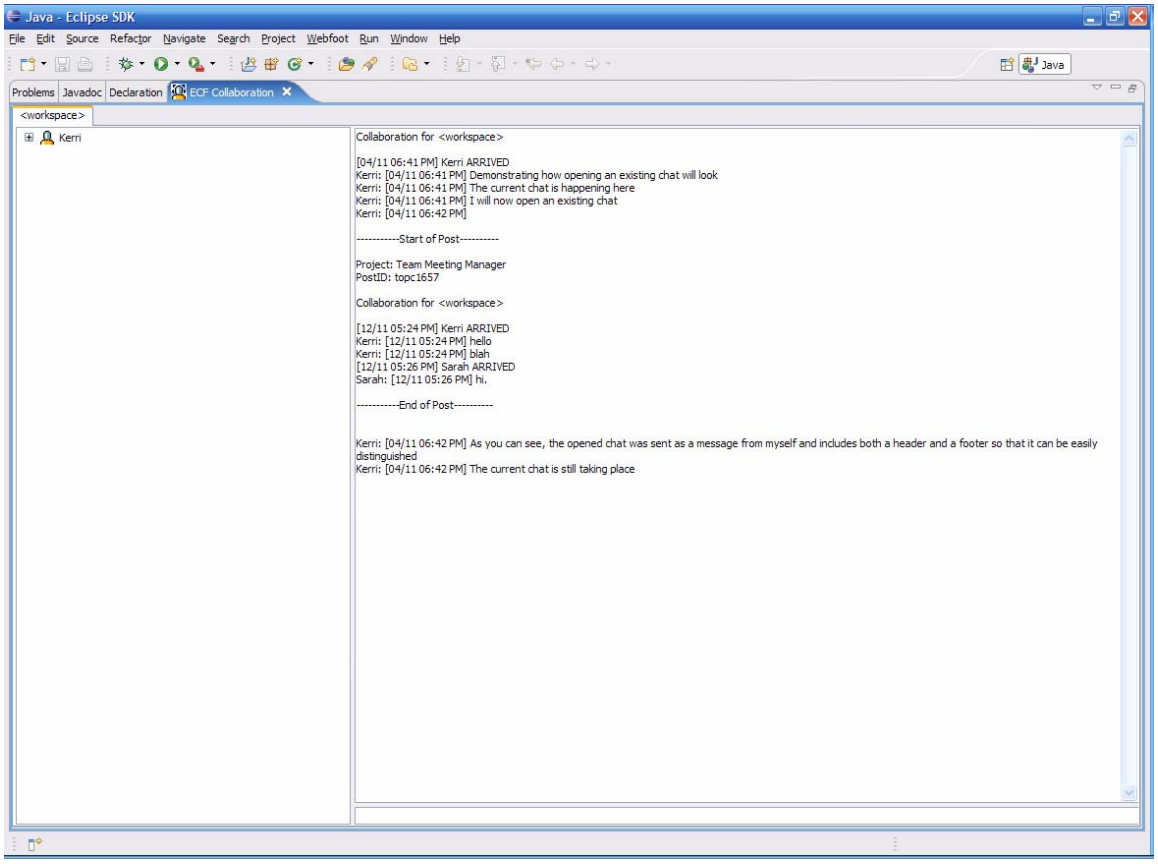


Figure 2: Opening an existing chat

With all of the features this project has added to Eclipse, teams will find it easier to work together and stay organized. With these capabilities, users will be able to keep track of their meetings in an organized fashion as opposed to the “old” way of doing things. With the completion of this project, full meeting transcripts will be saved from the virtual meeting, whereas if the users were to get together and meet, only the meeting

minutes would be available as an artifact of the meeting. Additionally, the option of opening an existing meeting within an ongoing meeting could prove to be extremely helpful. Giving the user the ability to create a task directly from the meeting reduces the chances of a miscommunication or forgetfulness. Since the user can create a task immediately after a good idea is introduced, there will be less no chance to forget between when the idea was manifested and the time when the user is back at his or her computer to record the idea.

4.2 Team Meeting Manager Help Section within Webfoot Help

The help manual has been included as a subset of the Webfoot help manual. In the Meeting Manager section of the manual the user will find guidance on performing all of the functions that have been added as a result of this project. The manual helps guide the user through connecting to the Meeting Manager, creating a task from the chat, saving the current chat to the discussion forum and also opening a chat within the current chat.

4.3 Plug-in UML

In the end, the code that needed to be incorporated into the existing ECF code was minimal. Three new classes were added to the existing project to add the features desired by this project. These new classes allowed the features to be added to the right-click menu and actions to occur when a selection from the menu is made. The methods themselves were quite minimal as they mainly called the methods that were created within the Webfoot SourceForge Core.

The final incorporation of the features into the Webfoot project can be summarized by the UML diagram shown in Appendix C. The packages included in the

diagram are those of the Webfoot SourceForge Core plug-in; they include the preferences, events and ui packages. New classes have been added to each of these packages to aid in the completion of this project.

The classes and methods within the preferences package add a Webfoot Preferences Page to the Eclipse Preferences so that the user may set their default preferences such as the SourceForge URL and their user name. The team has added new methods and constants to the classes in this package to enable the user to define a default forum in which to save their meeting transcripts. This default forum is also the default from which to open the existing chat. Although the user can define a default forum, it is not required. If one is defined, it does not have to be the forum used. When the user is prompted to select the forum to save to or open from, the default forum, if specified, is the forum which is selected in the list dialog. However, the user can choose to keep or change this selection before clicking OK.

The actions package is where the majority of the code for this project was added. Originally, the methods to prompt the user to select their active project were included within this package. For this project, classes and methods were added to prompt the user to select a Discussion Forum when both saving and opening a Meeting Transcript and to provide a Topic Title when they are saving their Meeting Transcripts. The classes which prompt the user to select the discussion forum contain methods to retrieve the list of forums from the user's selected project. For this reason, the user is first prompted to select an active project if they have not already done so. The classes which prompt the user for a topic title also contain methods for error checking. This, as earlier discussed, does not allow the user to submit a title which is blank or begins with a whitespace character.

The other package, the ui package, was not expanded by the completion of this project. However, the ui package contains methods which are pertinent to the success of the Webfoot project. The package contains methods which prompt the user to connect to SourceForge via a

login dialog so that they may log into their account. By logging into the account via Eclipse, the user can then have access to all of the elements which are part of their project within SourceForge. Once logged in, the user can then select the project which they wish to work. Once the user has logged into SourceForge and selected a project, all of the features added by this project are available. If the user attempts to use one of the added features without first logging in or selecting a project, they will be prompted to do so before the selected task can be completed. The reason for this sequence of actions is that the discussion board and tasks needed for the added features are specific to the project selected.

4.4 Metrics

The project ended up totaling approximately 700 lines of code. This is not much since as the capabilities were built upon an already existing chat room framework. The project had to only add the methods and actions to implement our features. Although the amount of code is not much and easily could have been written in a short amount of time, a large part the time spent on this project was learning the existing code and determining where and how to incorporate the new features. As a result, the team has created or added code to 14 classes which reside within 4 packages total.

5. Future Work and Conclusions

Although the project is complete, there several things that could be done in the future to further improve the framework and SourceForge integration. The first and most pertinent issue is that the meeting manager is not fully integrated with Webfoot. An extension of this project could be to fully integrate the plug-in with Webfoot and update the ECF framework to the most current version. There are several other extensions or new ideas that could be done to aid the developing field of collaborative development.

Though a specific way to integrate the "Open Existing Chat" functionality into an Eclipse view could not be determined during the scope of the project, it would be a great addition to the project. The hardest part of this addition would be to find a way to open a view in every chat participant's Eclipse. However, by opening the chat in a separate view, it allows the users chat to stay clutter free and easier to visually parse. Opening the existing chat in a separate view also allows users to scan the chat while still continuing the virtual meeting without the nuisance of continually scrolling back up to the opened chat in order to reference what was said.

A good addition to the meeting manager plug-in would be to allow for a shared whiteboard to aid in the initial phase of development. This could be added to the meeting manager by figuring a way to have Eclipse simultaneously open a "Whiteboard View" that could be shared by the users of the chat. Utilizing a whiteboard or some other visually shared workspace is often an important aspect of meetings. By utilizing a whiteboard, meeting members could easily visualize what the person writing or drawing

on the board is trying to get across without trying to understand what the person means when they try to explain it in the chat.

As collaborative development continues to increase due to the increasing number of teams working on a project being geographically or temporally separated, new features for these teams to utilize will constantly be in demand. The final product of this project and other projects completed as part of Webfoot has only begun to scratch the surface of the tools which will be needed by these teams. As new technologies and features are added, teams will find working together, although separated, to be easier and more comfortable. Being able to view the same things at the same time, while separated by distance, will help all members to feel connected and less divided.

Appendix A

About ECF

The Eclipse Communication Framework (ECF) is a framework built to support the development of distributed Eclipse-based tools and applications. It can be utilized to build new plug-ins, tools or full Eclipse applications which require point-to-point or publish-and-subscribe messaging. ECF also provides users with an open distributed component model, along with an expanding amount of extensible communication components. ECF has not yet released a final product, but currently in the stages of refactoring for release.

Appendix B

Use Cases

Creating a Task from Ongoing Chat

1. Meeting Facilitator/Project Leader logs into SourceForge via Eclipse
2. Meeting Facilitator/Project Leader initiates a chat
3. Members of a SourceForge Project group join
4. Discussion begins within chat
5. Meeting Facilitator right clicks within the chat pane and selects Create Task…
6. The user is prompted to select the Task Group folder which they want the new task placed into
7. User is prompted to fill in all necessary task information
8. User clicks OK and is returned to Eclipse

Saving a Chat

1. User right clicks on the chat pane once a chat is completed
2. User selects “Save Chat”
3. The user is prompted to log into SourceForge and select the project if they have not already done so
4. The user is prompted to select the forum in which they want the chat to be saved
5. The user is then prompted to type in a title for their post (cannot be blank or start with white space)
6. The user clicks OK
7. A dialog box is displayed to the user alerting them that their topic has been created.
8. The user clicks OK and is returned to Eclipse

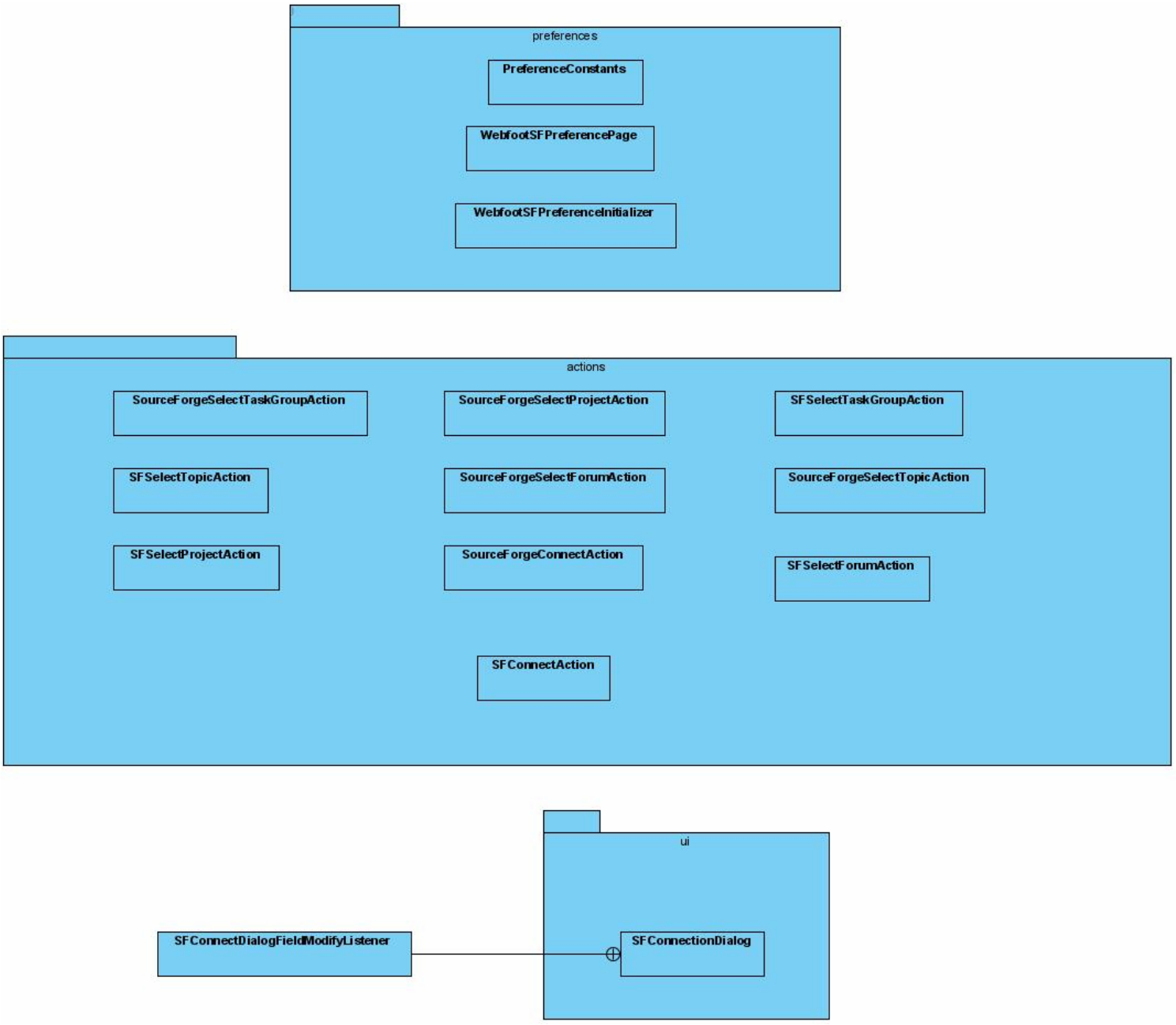
Opening a Saved Chat

1. User right clicks on the chat pane once a chat is in progress
2. User selects “Open Chat”
3. The user is prompted to log into SourceForge and select the project if they have not already done so
4. The user is then prompted to select the Forum in which the topic they wish to open resides
5. The user then selects the topic the wish to open

The transcript of the selected topic is the sent to the chat as a message from the user (including a header and footer) so that all other users of the chat can view the opened chat

Appendix C

UML Diagram



Glossary

API Application Programming Interface

ECF Eclipse Communication Framework

IDE Integrated Development Environment

SourceForge The world's largest Open Source software development web site with a centralized resource for managing projects, issues, communications, and code

UI User Interface

UML Unified Modeling Language

WPI Worcester Polytechnic Institute

References

"Agile Software Development." Wikipedia, The Free Encyclopedia. 24 April 2007. Retrieved April 25, 2007, from < http://en.wikipedia.org/wiki/Agile_software_development>.

Augustin, L., Bressler, D., Smith, G. "Accelerating Software Development Through Collaboration" 2002. 559-563.

D'Anjou, J., Fairbrother, S., et al. Java Developer's Guide to Eclipse. Boston: Addison-Wesley, 2004.

Hall, M., Krasner, S. "Java Integrated Development Environments (IDEs) and Editors". Retrieved April 4, 2007, from <http://www.apl.jhu.edu/~hall/java/IDEs.html>.

"Software Tools." Wikipedia, The Free Encyclopedia. 31 March 2007. Retrieved April 4, 2007, from <http://en.wikipedia.org/wiki/Software_tools>.

"VA Software: SourceForge Product Introduction." VA Software Corporation. 2005. Retrieved April 4, 2007, from <http://www.vasoftware.com/sourceforge/index.php>.

White, S.A., C. Lemus-Olalde. "Architectural Reuse in Software Development." 1998. Retrieved April 23, 2007, from <http://nas.cl.uh.edu/whites/webpapers.dir/etce98.pdf>, pg. 1.