Integrating COTS Search Engines into Eclipse: Google Desktop Case Study

Denys Poshyvanyk, Maksym Petrenko, Andrian Marcus

Department of Computer Science
Wayne State University
Detroit, Michigan USA 48202
denys, max, amarcus@wayne.edu

Abstract

The paper presents an integration of the Google Desktop Search (GDS) engine into the Eclipse development environment. The resulting tool, namely Google Eclipse Search (GES), provides enhanced searching in Eclipse software projects.

The paper advocates for a COTS component-based approach to develop useful and reliable research prototypes, which support various software maintenance tasks. The development effort for such the tools is reduced, while customization and flexibility, to fully support the needs of developers, is maintained. The proposed solution takes advantages of the power of GDS for quick and accurate searching and of Eclipse for great extensibility. The paper outlines our experiences of integrating GDS engine into Eclipse as well as possible extensions and applications of the proposed tool.

1. Introduction

Incorporating reliable commercial-off-the-shelf (COTS) components into software systems is desirable and not uncommon as there are many successful stories attesting such practices. For example, a web designer would rarely build a web-server from scratch, as there are many COTS software components for building web-based systems [6].

Using COTS software components to build research prototypes in academia obviously has many benefits as well. Software systems developed as research prototypes or as proof-of-concept tools often suffer from problems which prevent their wide-spread adoption among researchers or practitioners from industry. However, such tools have yet to make it in the mainstream of the software development practice. As with most research prototypes, some of these tools might suffer from limited interaction with potential users or financial support to maintain those tools, thus delaying their wide acceptance.

In order to mitigate some of the problems associated with research prototypes, we advocate in this paper an approach, which allows us developing useful and reliable tools by leveraging the advantages of existing COTS components. We present a particular case of implementing such a tool. The tool combines an existing off-the-shelf component for searching, namely Google Desktop Search[1], with the Eclipse[2] development environment. The tool is named Google Eclipse Search (GES) and it leverages the strengths of GDS, the extensibility of Eclipse, their popularity, and thus promises a wide-spread use among developers. The paper discusses some of the advantages of the solution, which we did not anticipate at the moment of incorporating GDS into Eclipse. The situation is not unique, as it was observed before by others: "innovative ways of integrating COTS into software systems usually unimagined by their creators" [5].

The next section presents some background information and our motivation for building GES using an existing COTS component. In section 3, we provide more details on the actual integration of GDS into Eclipse. The section 4 discusses some of the possible applications of GES.

2. Background and motivation

Recently, we have been working on developing a new methodology to support searching and browsing activities of software developers in the source code [8]. Since *searching* has been recently redefined by the internet search engines, most of them being based on information retrieval (IR) techniques, we applied a similar approach for searching in the source code of software projects and proposed a new methodology based on indexing of the source code using advanced IR techniques [8]. In order to bring the technology to the fingertips of developers, we interoperate our tools with MS Visual Studio [10] and Eclipse [9]. However, our tools may still suffer from the same problems as the majority of the research prototypes do, especially in terms of computational efficiency and the online re-indexing of the large-scale software as it changes during maintenance and evolution. In order to bring the technology closer to adoption among the

[1] http://desktop.google.com/
[2] http://www.eclipse.org/

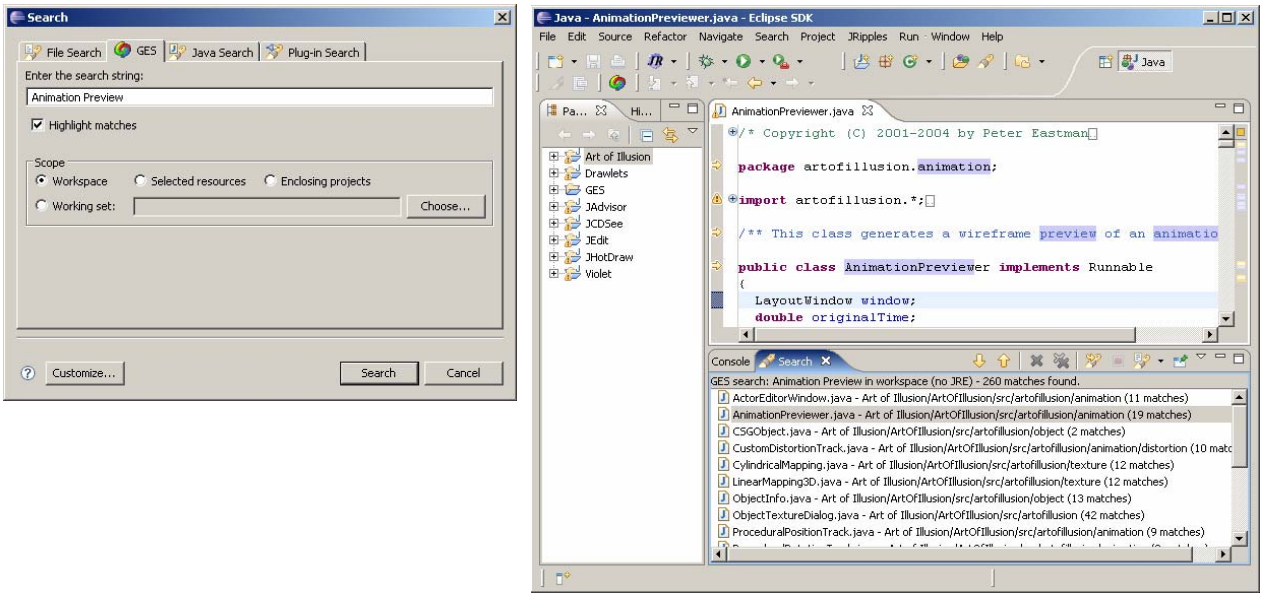

Figure 1. The GES plug-in for Eclipse (left) showing the results for the query "animation preview" (right) while searching in the source code of Art of Illusion software system

software developers, we needed to solve the problems highlighted above.

Lately, Google released its technologies for searching desktops in Google Desktop Search (GDS) [3]. One of the aspects that set GDS apart is its COTS-based architecture, which allows incorporating GDS into other applications via Google Desktop SDK[3]. Using this SDK, Google Desktop can be configured to index various types of applications and files, including the source code files.

In addition, GDS has many other features that were missing in our previous efforts, such as efficiency and the facility to unobtrusively index and re-index the source code files as they change during maintenance and evolution.

GDS can be used to search the files of a software project as it is via an Internet browser. However, such a use might be uncomfortable in some situations, since it would break the work flow of the developers as they would have to constantly switch between the IDE and the browser.

Incorporating GDS into Eclipse environment will provide the following supplementary features for searching in source code of software projects: on-the-fly preprocessing and indexing of the context; developer-friendly search methods; rapid indexing of specified objects in specified locations; persistent indexing, which maintains and updates content location changes for more accurate results; background indexing, lenient to user's CPU usage; quick response to developer's search queries; history of searches; and ranking of the results by relevance or date.

Finally, incorporating GDS into Eclipse has other advantages over existing solutions for searching software projects:

- *multiple* term queries, which is a specific feature of IR-base searching, as well as ranking of the results of the search;
- *robustness* and *reliability* of search engine component (i.e., GDS), which is important for large file repositories such as large scale software systems;
- access to the results of the search within Eclipse's IDE using *native interfaces* that provide direct links between the search results and their respective positions in the source code editor.

3. Integrating Google Desktop into Eclipse

GES is implemented as a plug-in for the Eclipse development environment (see Figure 1) to be used for searching within projects or within a custom working set of files using natural language (or multiple word) queries.

The GES search dialog is displayed in the standard Eclipse search dialog panel and the search results are presented through the standard search results presentation view (see Figure 1).

The GES search experience is similar to Eclipse's *File Search*. In order to perform a search using GES, the user has to type a query into the GES search dialog and specify the scope of the search (i.e., workspace, selected resources, enclosing projects, or working set). After the execution of the query, the search results are displayed

[3] http://desktop.google.com/developer.html

within the GES search results tab, similar to the one of regular Eclipse search (see Figure 1). The results can be easily explored by simply browsing the files in the editor. When source code files are in the scope of the search, the terms from the original query that are found in the java file are highlighted with colors (see Figure 1). Note that the terms from the query need not be in immediate vicinity of each other in the source code.

Through GES, the user can take advantage of all the intrinsic features of GDS, including searching using a set of terms, exact phrases, queries with Boolean operators, or restricting the search results to specific file types (i.e., by using the "filetype:" modifier).

3.1. Implementation details

To be generally accepted, GDS exposes its API through HTTP communication and XML, which adds some programming burden for the clients using GDS. Fortunately enough, the interface, supplied by one of the GDS plug-ins, the GDS Java API[4], hides all the implementation details so that clients can access GDS from any java application. The implementation of the GDS Java API is based on JAXB (Java architecture for XML binding[5] which maps semi-structural XML elements for flat-structural objects), thus users can formulate queries to GDS just by calling provided functions and traverse search results as easy as traversing elements in a simple Java list.

In order to maintain common look and feel of Eclipse search tools, we decided to reuse Eclipse search components. Being extremely extensible environment, Eclipse provides means to extend virtually every possible part of its GUI and search dialogs with no exception. Therefore, we decided to use the extension points of org.eclipse.seach group – searchPages to provide search dialog GUI and searchResultViewPages to provide search results GUI. There are also two additional extension points in the search group - textSearchEngine and textSearchQueryProvider, - which should ease creation of the text-based search engines, but were of no help in our project.

While extending searchPages were simple and involved implementation of a simple dialog window, searchResultViewPages demanded implementation of the ISearchResultPage interface which, as a parameter, accepts search results formatted accordingly to the ISearchResult interface. In addition to implementation, those two interfaces demanded implementation of chain of auxiliary interface classes to be either passed as function parameters or produced as function results.

As the search tools are not what developers typically extend in Eclipse, we found no documentation on the suggested search framework besides basic API documentation on the mentioned interfaces. Therefore, we decided to reverse engineer available sources of Eclipse search tools in org.eclipse.jdt.internal.ui.search package.

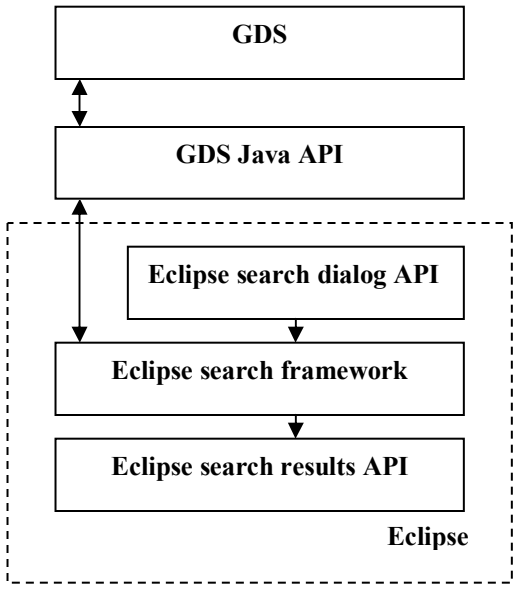

Figure 2. Logical structure of GES tool

After all, we were able to reuse 7 classes from this package with only minor modifications and 2 with more advanced changes. One class was modified to call GDS and obtain search results, whilst another class was modified to highlight occurrences of the search terms in found java files. It is also worth mentioning that in inspected framework, search engine was called in the middle of the delegation chain, formed by Eclipse search framework classes between search dialog and search result classes, thus making it hard to find the actual place GDS had to be called from.

We also had to copy 5 additional classes (like Messages class that provides common search results messages) from the same package without any modifications as they had internal visibility scope (see the package name) and were not available for the direct use.

Described approach saved us a lot of time in that we did not have to learn the framework and implement set of many unfamiliar interfaces, but rather modify several classes to use GDS as the source of search results. However, even with this strategy the effort was quite substantial to identify those couple classes within the available Eclipse packages. Final diagram with the logical structure of GES tool is presented in Figure 2. We made the source code of GES available to the research community, so the interested reader may study integration

[4] http://desktop.google.com/plugins/javaapi.html
[5] http://java.sun.com/xml/jaxb/

in more details by downloading GES's source code from sourceforge[6].

3.2. Formulating search queries and processing search results.

As it was mentioned, we used GDS Java API to communicate with GDS. However, in order to make successful searches within Eclipse resources, we had to solve several problems.

The major problem was restriction of the GDS search results to the scope, selected by a user. In its basic version, GDS searches for provided terms in the whole hard drive of the user. However, as we need to search only within projects, loaded into Eclipse environment, we need the restrict GDS search results to the files of those projects. Furthermore, user may want to restrict search scope even more to the particular Eclipse entities as those available in standard Eclipse's search dialog.

As there is no direct way to restrict GDS search scope, we investigated couple indirect "tweak" methods. The simplest way is to allow GDS to search the whole hard drive and then to filter the results; however, this method is clearly inefficient as it involves processing a lot of irrelevant information. Another possibility is to modify undocumented Windows registry keys settings of GDS, which can be used to set up GDS to index only those folders that relate to the scope, chosen by the user. In this case penalty is the time which GDS takes to re-index folders after the registry keys are modified.

Finally, we discovered the method that solved the problems of previously mentioned solutions: if the fully-qualified file or folder name is added as a part of the search request, the search will be limited to that file or folder. Therefore, we used this fact to convert list of files and folders within Eclipse search scope into the appropriate GDS query. Also, in recent GDS releases, Google introduces special tag words to specify a folder (but not a file) to search within, which enabled us to optimize our searches even further.

The other problem is that GDS provides only the list of files with the search terms, but not the locations of those terms within the files. As Eclipse search tools typically highlight found terms in the code, we had to implement the similar feature. Currently, we simply open every file, returned by GDS, and perform a plain text search within those files for the requested terms. However, as current GDS has a capability of highlighting search terms in the cached version of the files, we hope that in future releases GDS API will include means for determining position of search terms in the text of the found files.

[6] http://ges.sourceforge.net/

3.3. Additional issues

Since GDS is not an open-source application, the only possible way to customize it currently is through the available GDS SDK and undocumented Windows registry keys. This issue raises several challenges in building and using GES.

One of the major issues is the GDS' background indexing. By default, GDS indexes (and re-indexes) the user files only when the user's computer is idle; thus, to be able to initially use it, the user typically needs to wait until GDS completes the (re-)indexing of the files. Unfortunately, currently this problem can not be addressed using GDS preferences or GDS API calls. Ideally, we would like to allow the user the option to choose when and how the files to be (re-)indexed.

4. Applications of GES

Originally, the tool was presented in [11], however after that GES has been applied and shown to be useful in the set of case studies [12]. In addition, there are possible applications of this tool which we discuss in this section.

For example, GES can be used in its current form to index not only source code files, but also project-related external documentation in various formats. Concept location and program comprehension can be improved by searching within the external documentation in addition to the source code.

Also, GES can be extended with proxy server classes[7] to be used as a server for indexing source code repositories and handling queries from multiple clients, which will allow searching remote machines. With such an extension, GES could provide support for various collaborative tools like [2] and [4]. In this context, several versions of the software, extracted from repositories could be indexed together or separately. This requires some additional implementation effort, which we are currently undertaking.

GES could be successfully used as an complementary search feature within other source code exploration tools like the Aspect Browser [13], Creole [7], or JRipples [1] etc.

Moreover, the experience of integrating GDS into the Eclipse environment allows us to repeat the effort with other IDEs and/or search engines. In other words, the search engine may be seen as a service provider while the IDE may be the service consumer. For example, we could extend GES to manage several other external search engines that provide extensions via SDK, like Copernic, and implement the same plug-in for MS Visual Studio or CodeWarrior.

One important issue we are working on is to modify the storage of the source code, such that GES could index

[7] http://www.projectcomputing.com/resources/desktopProxy/

and return results at different granularity levels than files (e.g., classes, methods, etc.). GES is available as open-source application and other researchers modified it for their purposes [12].

In future versions, GES will give users more direct control over additional advanced features of GDS. In addition, we will investigate the benefits of integrating GES with other Eclipse software browsing plug-ins.

5. Conclusions and future work

Incorporating GDS into Eclipse is a COTS-based solution to improve source code searching and produce an easier to adopt approach to this problem. GES allows Eclipse software developers to perform searches in the source code and associated documentation of a software system, using most features offered by GDS.

In addition, this COTS-based combination has one important advantage – whenever a new version of Google Desktop is released, the programmer does not have to implement any changes to the tool, but rather install new version of GDS and use new features available in that version without extra work.

6. Availability

GES is registered as official Google gadget, available at http://desktop.google.com/plugins/i/eclipse_search.html. The source code is also available at http://ges.sourceforge.net/

7. Acknowledgements

This research was supported in part by grants from the National Science Foundation (CCF-0438970 and a 2006 IBM Eclipse Innovation Award).

8. References

[1] Buckner, J., Buchta, J., Petrenko, M., and Rajlich, V., "JRipples: A Tool for Program Comprehension during Incremental Change", in Proceedings of 13th IEEE International Workshop on Program Comprehension (IWPC'05), May 15-16 2005, pp. 149-152.

[2] Cheng, L.-T., Hupfer, S., Ross, S., and Patterson, J., "Jazzing up Eclipse with collaborative tools", in Proceedings of OOPSLA workshop on eclipse technology eXchange, 2003, pp. 45-49.

[3] Cole, B., "Search engines tackle the desktop", in *IEEE Computer*, vol. 38, 2005, pp. 14-17.

[4] Cubranic, D., Murphy, G. C., Singer, J., and Booth, K. S., "Hipikat: A Project Memory for Software Development", *IEEE Transactions on Software Engineering*, vol. 31, no. 6, June 2005, pp. 446-465.

[5] Egyed, A., Müller, H., and Perry, D., "Integrating COTS into the Development Process", in *IEEE Software*, vol. July/August, 2005, pp. 16-19.

[6] Johann, S. and Egyed, A., "State Consistency Strategies for COTS Integration", in Proceedings of 1st International Workshop on Incorporating COTS Software into Software Systems (IWICSS'04), Redondo Beach, CA, 2004, pp. 33-38.

[7] Lintern, R., Michaud, J., Storey, M. A., and Wu, X., "Plugging-in Visualization: Experiences Integrating a Visualization Tool with Eclipse", in Proceedings of ACM Symposium on Software Visualization (SoftViz'03), 2003, pp. 47 - 57.

[8] Marcus, A., Sergeyev, A., Rajlich, V., and Maletic, J., "An Information Retrieval Approach to Concept Location in Source Code", in Proceedings of 11th IEEE Working Conference on Reverse Engineering (WCRE'04), Delft, The Netherlands, November 9-12 2004, pp. 214-223.

[9] Poshyvanyk, D., Marcus, A., and Dong, Y., "JIRiSS - an Eclipse plug-in for Source Code Exploration", in Proceedings of 14th IEEE International Conference on Program Comprehension (ICPC'06), Athens, Greece, June 14-17 2006, pp. 252-255.

[10] Poshyvanyk, D., Marcus, A., Dong, Y., and Sergeyev, A., "IRiSS - A Source Code Exploration Tool", in Proceedings of 21st IEEE International Conference on Software Maintenance (ICSM'05), Budapest, Hungary, September 25-30 2005, pp. 69-72.

[11] Poshyvanyk, D., Petrenko, M., Marcus, A., Xie, X., and Liu, D., "Source Code Exploration with Google ", in Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM'06), Philadelphia, PA, 2006, pp. 334 - 338.

[12] Shepherd, D., Fry, Z., Gibson, E., Pollock, L., and Vijay-Shanker, K., "Using Natural Language Program Analysis to Locate and Understand Action-Oriented Concerns", in Proceedings of International Conference on Aspect Oriented Software Development (AOSD'07), 2007, to appear.

[13] Shonle, M., Neddenriep, J., and Griswold, W., "AspectBrowser for Eclipse: a case-study in plug-in retargeting", in Proceedings of OOPSLA workshop on eclipse technology eXchange, 2004, pp. 78-82.