AutoDoc: A Search and Navigation Tool for Web-Based Program Documentation

Richard Wheeldon, Mark Levene and Nadav Zin
Department of Computer Science
Birkbeck College, University of London
London WC1E 7HX, U.K.
{richard,mark,nadav}@dcs.bbk.ac.uk

1 Abstract

We present a search and navigation tool for use with automatically generated program documentation, which builds trails in the information space. Three user interfaces are suggested, which show the web pages in context, and hence better explain the structure of the code.

2 Introduction

The documentation process is an important part of any software development. Extensive documentation of any program is essential if it is to be maintainted and enhanced. To meet this demand, many companies and organizations are using tools capable of producing Hypertext documentation, typically in HTML format suitable for placing on a project web site or company intranet.

Such documentation is typically generated from special markup in the code. One of the best known examples of such a system is the Javadoc [2] tool shipped with all versions of the Java Development Kit. Exploration of the Google index reveals over 800 examples of documentation sets created with Javadoc with over 200000 HTML pages between them. Given that so many such archives are firewalled or restricted from robots, the true number is most likely to be much higher.

Similar systems also exist for other languages, with Object Oriented languages such as C++, C# and Java gaining particular benefit from the mapping between classes and web pages. However, none of the tools for creating such documentation provide search facilities or automated navigational support for the user. We have devloped a new search and navigation tool, which we call AutoDoc, to address these problems. An example of AutoDoc using the Javadocs for Sun's Java Development Kit v1.4 is available at www.navigationzone.net

3 The Search and Navigation Challenge

Visitors to a Web Site often "get lost in hyperspace" when they lose the context in which they are browsing, and are unsure how to proceed in terms of satisfying their original goal [5]. The unresolved problem in Web site usability, of assisting users in finding their way, is termed the *navigation problem* [3]. In order to help tackle this problem, we developed a navigation system which builds information *trails* (or navigation paths) in response to a user's query. The basic architecture of this system and the first user interface, which presents the results in a tree-like

structure that users can interact with, was discussed in [4]. The trails presented in the tree structure were generated from the link structure of the hypertext, but could sometimes be confused with a hierarchy or directory structure. This is a problem we have attempted to address with our new TrailSearch and GraphSearch interfaces.

A similar problem also exists in automatically-generated corpuses such as program documentation. Program documentation, like other online support systems, provides the opportunity for enhanced productivity. However, users often take longer finding the required information in such systems than they would in conventianal paper-based documentation [6].

Such corpuses are often highly interlinked. Links are created in JavaDocs due to package structure, class inheritance and type references. This is of great benefit if the information can be harvested correctly, as a link exists in most cases where the relationship between the classes suggests one should. Hence, a greater number of potential trails also exist. Filtering this increased density of link information provides the challenge. We achieve this by combining our Best Trail algorithm with a set of heuristics customized for on-line documentation.

Interesting questions may be asked of such corpuses. Should links based on the usage of objects in code be treated in the same way as those based upon human judgements? Are these created with the same distributions of preferential linking as found in web sites? How does this affect the performance of metrics which assume that links denote quality or authority, such as Kleinberg's Hubs & Authorities (HITS) or Brin and Page's PageRank?

4 The AutoDoc Solution

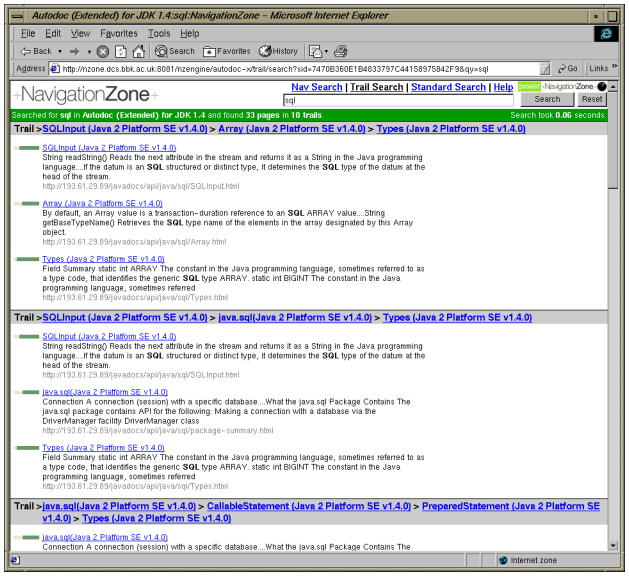
The AutoDoc tool is based on the same technology as our website navigation tool, SiteNavigator. The trail-finding and information retrieval subsystems remain the same, as do the major components of the user interfaces. AutoDoc, like SiteNavigator, provides full-text indexing of the corpus. Our trail finding algorithm uses probabilistic expansion of a search tree to select candidate trails quickly, with the results returned in an XML format that can be adapted using XSLT StyleSheets to any format required. Three user interfaces are provided, each with their own advantages and restrictions.

4.1 Flat TrailSearch User Interface

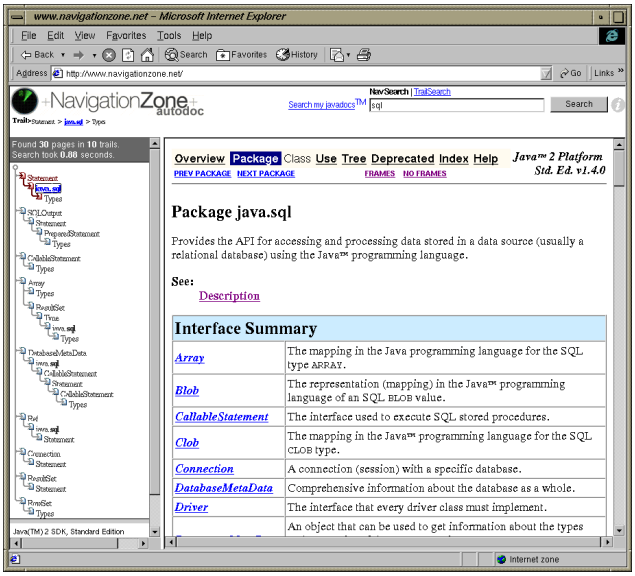
The flat TrailSearch user interface appears very similar to that of a traditional search engine. Each trail appears sequentially and users can follow any link they choose. Such an interface is the best choice for introducing new users who are already familiar with search engines to the ideas of a returned trail. However, it is still difficult to see the context of a node or the structure of a site in such a display, nor is there adequate support during the navigation session.

4.2 Improved NavSearch User Interface

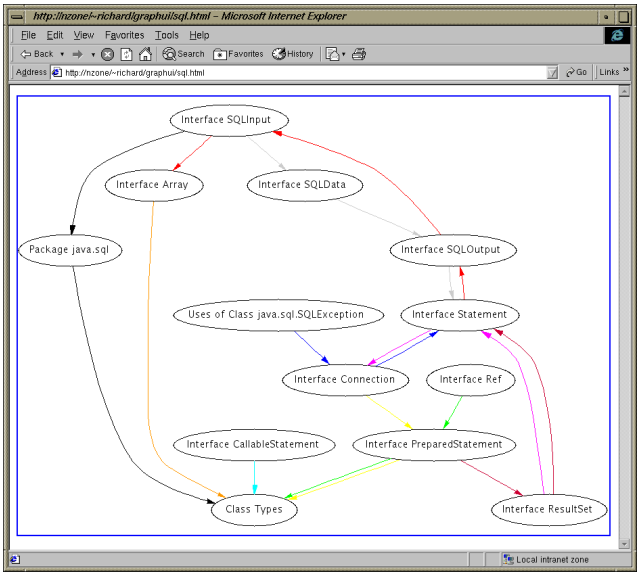
The NavSearch user interface appears at first glance to be identical to that presented in [4]. The two main elements are a *navigation tool bar* comprising of a sequence of URLs (the "best trail") and a *navigation tree window* with the rest of the trails. Improvements provide faster responses to queries and better filtering of results to remove duplicate entries and redundant information.



(a) TrailSearch



(b) NavSearch



(c) GraphSearch

Figure 1: Three user interfaces for AutoDoc, showing results for the query "sql". The results show the classes of JDBC (Java's software for connecting to SQL databases). By examining the links between pages on the trails, we can see the connections between the classes.

4.3 Clickable GraphSearch User Interface

We have developed a prototype interface, using the GraphViz program [1] which displays the results in the form of a graph, where each trail is indicated by a different colour. The trail set output is simply piped to the clickable image map generator.

AutoDoc allows increased productivity while programming, being quicker and more up-to-date than using a book or manually navigating the on-line documentation. It is thus ideal for experienced users who want answers quickly, as well as a providing a smooth introduction to the language for novice programmers. It also helps to expand developer's knowledge by providing contextual information such as relevant classes in the hierarchy, implemented interfaces and external specifications, giving pedagogic value as a teaching tool.

5 Future Work

Following the release of AutoDoc for Java documentation, we are currently extending the coverage for C++, C# and other languages. We intend to further develop the GraphSearch user interface to production quality, with pop-up windows containing useful information similar to those in the NavSearch interface, including document summaries and metadata. Finally, we intend to allow personalization so that programmers working on a particular field have query results tailored to their particular needs.

References

[1] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem-Phong Vo. A technique for drawing directed graphs. *IEEE Trans. Software Engineering 19*, pages 214–230, 1993.

[2] Sun Microsystems Inc. Javadoc tool home page, 2001. http://java.sun.com/j2se/javadoc/.

[3] M. Levene and G. Loizou. Web interaction and the navigation problem in hypertext. In A. Kent, J.G. Williams, and C.M. Hall, editors, *Encyclopedia of Microcomputers*. Marcel Dekker, New York, NY, 2002. To appear.

[4] M. Levene and R. Wheeldon. A Web site navigation engine. In *Poster Proceedings of International World Wide Web Conference*, Hong Kong, 2001.

[5] J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Indianapolis, Indiana, 2000.

[6] Martin D. Tomasi and Brad Mehlenbacher. Re-engineering online documentation: Designing examples-based online support systems. *Technical Communication, 46*, pages 55–66, 1999.