

Analyzing Potential Year 2000 Code Problems with Source-Navigator

This article describes how to use Cygnus Solutions' Source-Navigator code-comprehension tool to detect and fix potential Year 2000 (Y2K) problems in existing code. Though not originally designed for Y2K applications, Source-Navigator is ideal for Y2K applications.

Source-Navigator, the source code-comprehension tool from Cygnus Solutions, is very well suited for Year 2000 analysis. Its ability to automatically identify date-relevant fields in source code makes it a prerequisite for anyone with plans to make existing program code Year 2000 compliant. Cygnus Solutions' German distributor, CAS GmbH, has developed Year 2000 extensions to Source-Navigator via the Software Development Kit (SDK). The following explains how Source-Navigator and the CAS extensions can be used to analyze code and determine the extent of potential Year 2000 (Y2K) problems.

YEAR 2000 COMPONENTS

Typically Y2K problems are hidden in very small components of the total code within a given program. The following are the major groups of components.

- **Literals**
These are any hard-coded dates such as "12/25/98".
- **Containers**
These are symbols that store date or time information; such a symbol might be called ChristmasDay and store the string value "12/25/98".
- **Modifiers**
These are functions or operations that read or write date/time information such as DaysToChristmas (DateNow, ChristmasDay) that calculates the number of days until Christmas.
- **References**
These are containers and modifiers that are linked together in "referred to" or "referred by" relationships. For example, BuyGiftsNow may refer to DaysToChristmas, which in turn refers to DateNow and ChristmasDay. DaysToChristmas is referred by BuyGiftsNow.

USING SOURCE-NAVIGATOR TO ANALYZE YEAR 2000 COMPONENTS

Step 1 - Building a Symbol Database

All Source-Navigator tools are organized around a project database that holds all project-specific objects. The name and location of the project files, symbols extracted from the source, and the relationships

between symbols reside in the project database. File structure may not be pertinent when visualizing code relationships, so Source-Navigator allows the user to view and understand software structure regardless of which file contains what information. The navigational tools are the Project Editor, Symbol Browser, Editor, Class Hierarchy, Class Browser, Cross-Reference and the Include Browser.

Creating a project database simply involves pointing Source-Navigator to a set of source files to be analyzed. Source-Navigator automatically builds a project database. Once the database is completed, the Symbol Browser window appears as illustrated in Figure 1.

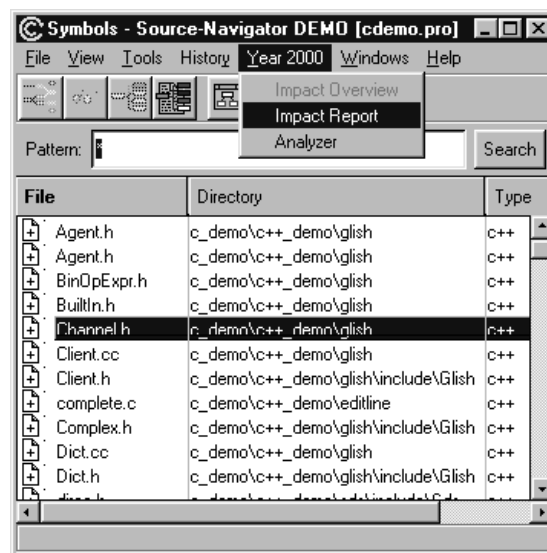


Figure 1. Building a Symbol Database

STEP 2 - SEARCHING FOR YEAR 2000 COMPONENTS

The CAS extensions add a Year 2000 menu item to the Symbol Browser. Choosing this menu item invokes the Impact Analysis main window shown in Figure 2.

The basis for the Impact Analysis is a list of search criteria for literals, containers and modifiers within the project database that contain date-related strings. The search criteria can be changed via the 'include' and

AD ESCE

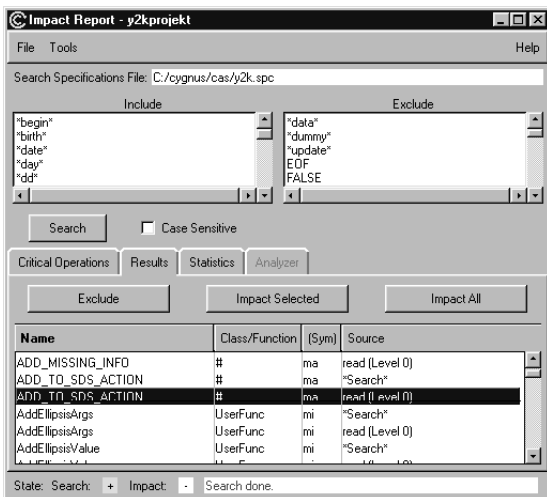


Figure 2. Searching for Year 2000 Components

'exclude' list and may be extended for specific custom criteria.

Clicking on the search button begins the analysis. Upon completion, a list of all the symbols that match the search criteria is displayed. Critical operations and exclusion criteria may filter the list.

STEP 3 - ANALYZING THE YEAR 2000 PROBLEM

There are two methods to further analyze the Year 2000 problem from the list created in the Impact Analysis window:

Method 1. From the Analyzer

Choosing the Analyzer tab from the Impact Analysis window will create a detailed description of where and how a specific symbol is used in the code in HTML for-

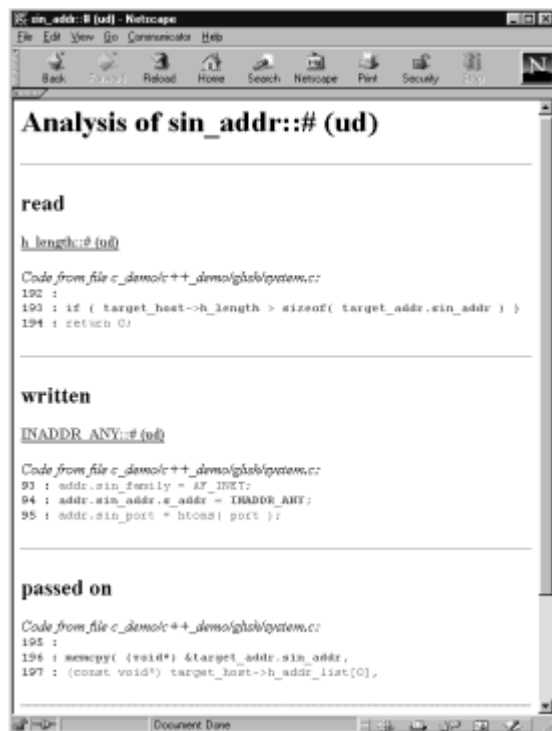


Figure 3. Analyzing the problem from the analyzer

mat. Figure 3 shows the impact of ATTNAM2 including read in, written in and passed on in. The number of lines of code displayed before and after the symbol may be selected. In addition, by double clicking on the symbol the Source-Navigator Editor will be invoked to exact location of the symbol within the source file. From this point the appropriate changes can be made in the code.

Method 2. Directly Into Source-Navigator

Double clicking on the highlighted line within the Impact Analysis list invokes the Source-Navigator Editor to the exact location of the symbol within the source files. The appropriate code changes can be made/saved in the editor, or further analysis can take place using standard features of Source-Navigator. Below is a brief summary of the standard Source-Navigator features and their applicability to the Year 2000 problem.

Editor

The Editor, the main Source-Navigator window, combines navigating with browsing. It allows for navigating through actual program text, and shows its location in the source code. The Editor highlights the basic syntax of all supported programming languages and updates the project symbol database when files are modified and saved. Consequently the project symbol database is always up-to-date with changes made during the development cycle. By using the Impact Analysis list, jumping to the symbol with a Year 2000 problem and making the necessary code changes is easy.



Figure 4. Analyzing the problem directly into Source Navigator

Symbol Browser

The Symbol Browser, which operates on the project symbol database, helps the user learn the existing program structure, and shows how and where symbols are used. From the Symbol Browser, one can quickly find the location of a symbol in the source code. If the step from Impact Analysis list to the Editor does not bring forth enough context surrounding the symbol with the Year 2000 problem, information from the Symbol Browser can be used to get a broader view of the location of the symbol within the source files.

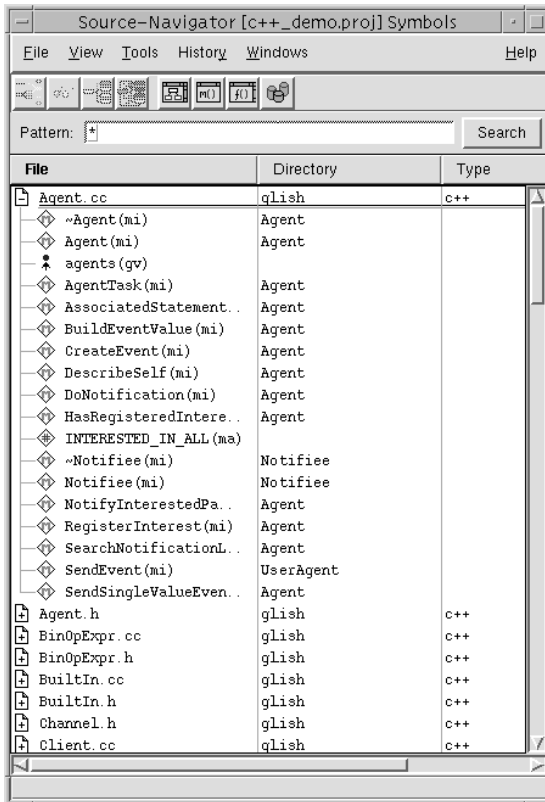


Figure 5. Symbol Browser

Class Hierarchy

The Class Hierarchy Browser helps the user to understand the inheritance relationship for a particular class. If the Year 2000 problem lies within C++ or Java source files, the Class Hierarchy Browser allows for a better understanding of the inheritance structure of the classes that may be effected.

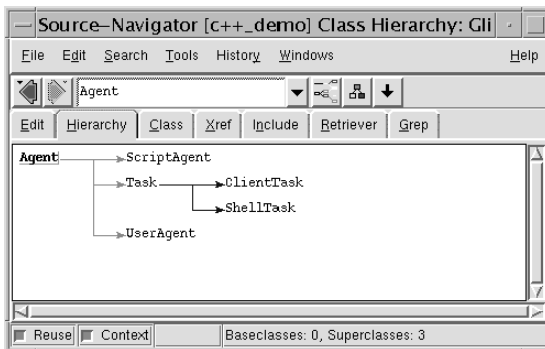


Figure 6. Class Hierarchy

Class Browser

The Source-Navigator Class Browser provides an intuitive list view of classes and their attributes. With the Class Browser, the user can analyze the project class members of a selected class based on inheritance tree, scope, member attribute, member type, and more. If the Year 2000 problem lies within C++ or Java source files, the Class Browser allows a user to better understand the inheritance structure of the classes that may be effected.



Figure 7. Class Browser

Cross-Reference

The Cross-Referencer helps the user understand complex source code by showing the relationships between components in the project. Cross-Referencer diagrams show the Refers-to relationships (drawn in blue arrows) and Referred-by relationships (drawn in red arrows) between different components in the source code. One can traverse the tree and expand or restrict the elements within the tree. The ability to transverse the tree allows a user to visually follow a specific symbol with a Year 2000 problem and understand its potential ripple effects on other source code components.



Figure 8. Cross-Reference

Include Browser

The Include Browser allows the user to display Includes and Included from relationships of source files in the same window. The Include Browser shows what other source files may need to be checked for Year 2000 problems.

Complex Views

The Editor, with its view tabs, can be used to launch a different view of a selected component. For instance, if a class name is highlighted, and the Class Hierarchy tab is selected, Source-Navigator will open a tree view of the inheritance line of that class.

With Source-Navigator's browser capability, the analysis tool allows the user to combine multiple views into a single window.



Figure 9. Include Browser

Source-Navigator can provide a lot of complex information in a number of intuitive ways. As a user navigates through a project, he may want to return to the view of a relationship that he was previously investigating. Source-Navigator stores a view history of the journey through the project. The left and right arrows in the toolbar act like previous and next buttons in popular Web browsers. For more detailed history, the History menu allows you to view a list of previous views on a per-tool basis. This allows the user to jump directly to the view desired, rather than paging through previous views. Because context is much easier to remember using complex views, related Year 2000 problems can be tracked down much faster than with traditional stand-alone text-based search tools.

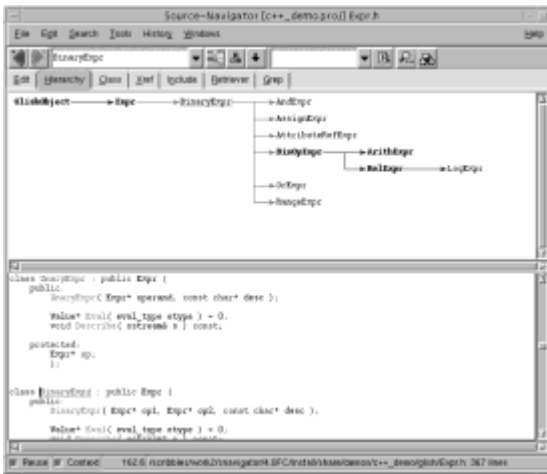


Figure 10. Complex Views

SOURCE-NAVIGATOR ENTERPRISE AND DEVELOPER EDITIONS WITH CAS EXTENSIONS

To accommodate different company sizes and project scales, Source-Navigator is available at two levels - the Developer Edition and the Enterprise Edition. Both editions provide visual representations of project source code and the ability to query and present code in different ways, such as by project, symbol, inclusion, class hierarchy and cross-reference. Both editions also include advanced browsing capability that allows for the creation of several views of a user's code base and navigation between them, while preserving the context of a selected component.

- The Developer Edition is designed for the single developer working on small to medium size pro-

jects (up to 100,000 lines of code per project). Its graphical front end to configuration management systems provides easy and quick check-in and checkout capability. Available on Windows NT/95/98, Red Hat Linux and Solaris, the Developer Edition includes a source editor and browsers that let you view a project in many different ways. This Edition also includes the C/C++, Java, tcl, FORTRAN, COBOL and assembly language parsers. CAS also provides parsers for Ada, Pascal, Perl and PL/I. Pricing for the Developer Edition is \$395US. The CAS extensions to the Developer Edition are priced at \$199US and have the following limitations:

- Impact Overview in place of the Analyzer (see Figure 11 below). This is designed to give a statistical overview of the Year 2000 problem only.
- No modifications allowed to the "exclude/include" search criteria.
- Impact Analysis can only be executed once per project.

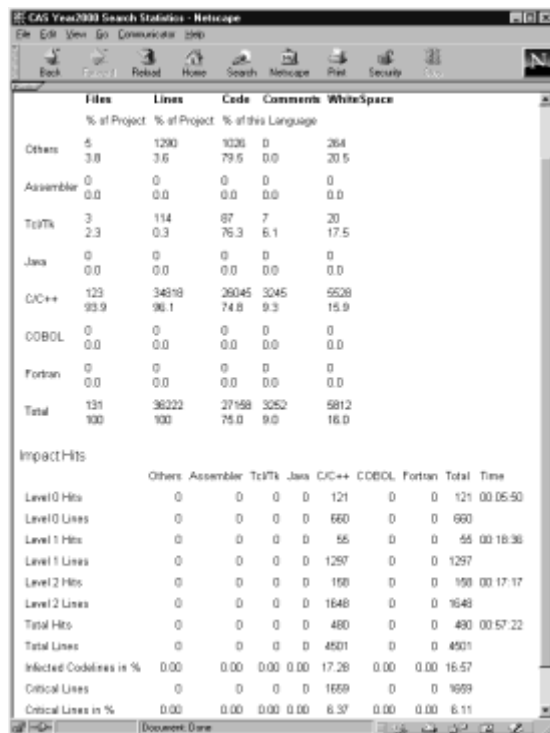


Figure 11. Source-navigator enterprise and developers editions with CAS extensions

- The Enterprise Edition includes all the capability of the Developer Edition and is designed for large-scale, complex projects - projects that require a large-scale development team and a great deal of organization. The CAS extensions function as described above in Steps 1, 2 and 3. ■

Mark Carson is Group Product Manager at Cygnus Solutions, mcarson@cygnus.com. Manfred Shlitt is an engineer at CAS GmbH in Obertshausen, Germany, y2k@cas-gmbh.de.