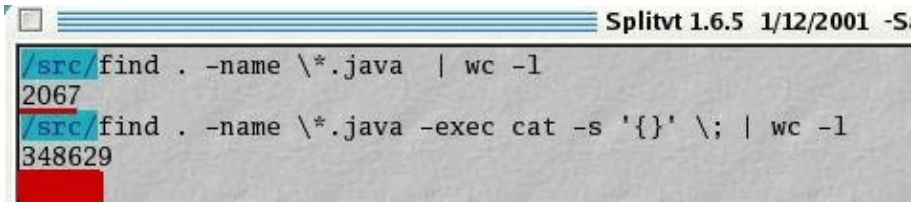
von Gerrit Renker
<gerrit.renker(at)gmx.de>

Über den Autor:
Gerrit mochte Computer überhaupt nicht, bis er es mit C und Linux versuchte.

Übersetzt ins Deutsche von:
Viktor Horvath
<ViktorHorvath/at/gmx.net>

Kreuzfahrt mit dem snavigator



Zusammenfassung:

Dieser Artikel stellt den snavigator vor, ein mächtiges Werkzeug für Code−Analyse, Querverweise und Re−Engineering, das unverzichtbar ist, um der Komplexität der Pflege von größeren Softwareprodukten und −paketen effizient zu begegnen.

Motivation

Ein altes Sprichwort sagt, daß ein Buch nicht nach seinem Umschlag beurteilt werden sollte. Etwas Ähnliches gilt für Open Source−Code. Open Source ist jedoch nicht dasselbe wie Open Documentation, und das Lesen wird immer schwieriger mit der wachsenden Anzahl und Länge der Quelldateien. Ich mußte kürzlich mit einer Software programmieren, die mit einer halben HTML−Seite Dokumentation aufwartete sowie mit über 348.000 Zeilen Open Source Java−Code, verteilt auf mehr als 2.060 Dateien (siehe Graphik). Angesichts solcher Dimensionen werden Werkzeuge für elektronische Orientierung, Reverse Engineering und Analyse unverzichtbar. Ein solches ist der *Red Hat source code navigator*, der in diesem Artikel vorgestellt wird.
Er automatisiert viele Aufgaben, die man normalerweise mit (c)tags, grep, search und replace erledigte, aber viel präziser und bequemer, in einer einfach zu benutzenden graphischen Oberfläche integriert, siehe die Screenshots unten.

Installation in Debian

Unter Debian kannst du das ganze Paket mit folgendem Einzeiler bekommen:

```
apt-get install sourcenav sourcenav-doc
```

Das besorgt gleichzeitig auch die Dokumentation. Der source navigator befindet sich dann in `/usr/lib/sourcenav/`, du kannst das Hauptprogramm mittels `/usr/lib/sourcenav/bin/snavigator` starten (siehe den Tip bezüglich eines Symlinks unten). Die Dokumentation kann unter `/usr/share/doc/sourcenav/html/` gefunden werden.

Installation aus den Quellen

Die Homepage von source navigator ist http://sourcenav.sourceforge.net/, die Downloads liegen hier (sourceforge.net/project/showfiles.php?group_id=51180). Hole den neuesten Tarball `sourcenav-xx.xx.tar.gz`. Während des Downloads kannst du dich mit etwas anderem beschäftigen – die Quellen sind 55 Megabytes groß. Das hat auch seine guten Seiten, das Paket braucht kaum etwas anderes. Obwohl es stark auf andere Bibliotheken wie Tcl/Tk, Tix und Berkeley DB zurückgreift, sind deren richtige Versionen alle mitgeliefert. Um Konflikte mit anderen Versionen von `Tcl/Tk` auf deinem System zu vermeiden, ist es eine gute Idee, die Installation in einem separaten Verzeichnis vorzunehmen, z.B. `/opt/sourcenav`. Die Anweisungen empfehlen des weiteren die Benutzung eines separaten *Kompilier–Verzeichnis (build directory)*, was wie folgt funktioniert. Im Verzeichnis, wo die entpackten Dateien liegen, mache dies:

```
mkdir snbuild; cd snbuild
../sourcenav-*/configure --prefix=/opt/sourcenav
make                              ## takes a while ...
make install                      ## might have to become root first
```

Die Option `--prefix` spezifiziert das Installations–Verzeichnis. Während `configure` läuft, bekommt man bereits eine Vorstellung von den vielen Sprachen, mit denen snavigator umgehen kann. Es ist ebenfalls möglich, weitere Parser für Sprachen deiner Wahl oder auch selbstgeschriebene hinzuzufügen. Ist die Installation via `make install` einmal beendet, ist der snavigator bereit zur Arbeit und kann als `/opt/sourcenav/bin/snavigator` aufgerufen werden. Anstatt die Umgebungsvariable PATH zu erweitern, schlage ich einen *symbolischen Link* vor, z.B. in `/usr/local/bin`.

```
ln -s /opt/sourcenav/bin/snavigator    /usr/local/bin
```

Das Hauptprogramm ist ein Shellscript, das sein Verzeichnis kennen muß. Daher wird es über einen Symlink–Aufruf verwirrt. Das kann mit der Änderung folgender Zeilen in `/opt/sourcenav/bin/snavigator` behoben werden: Ersetze

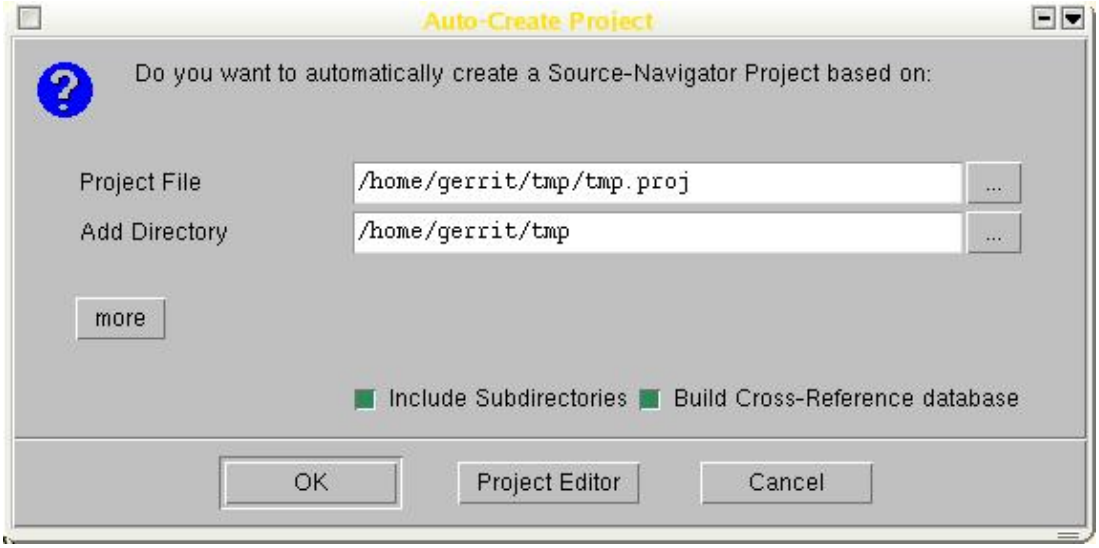
```
snbindir=`dirname $0`
```

durch

```
prog=`readlink -f $0`
snbindir=`dirname $prog`
```

Die Option `-f` von **readlink(1)** erzeugt eine kanonische Darstellung des Pfades, d.h. es funktioniert sogar, wenn auf die Datei über einen sehr langen Pfad von verschachtelten Symlinks zugegriffen wird.

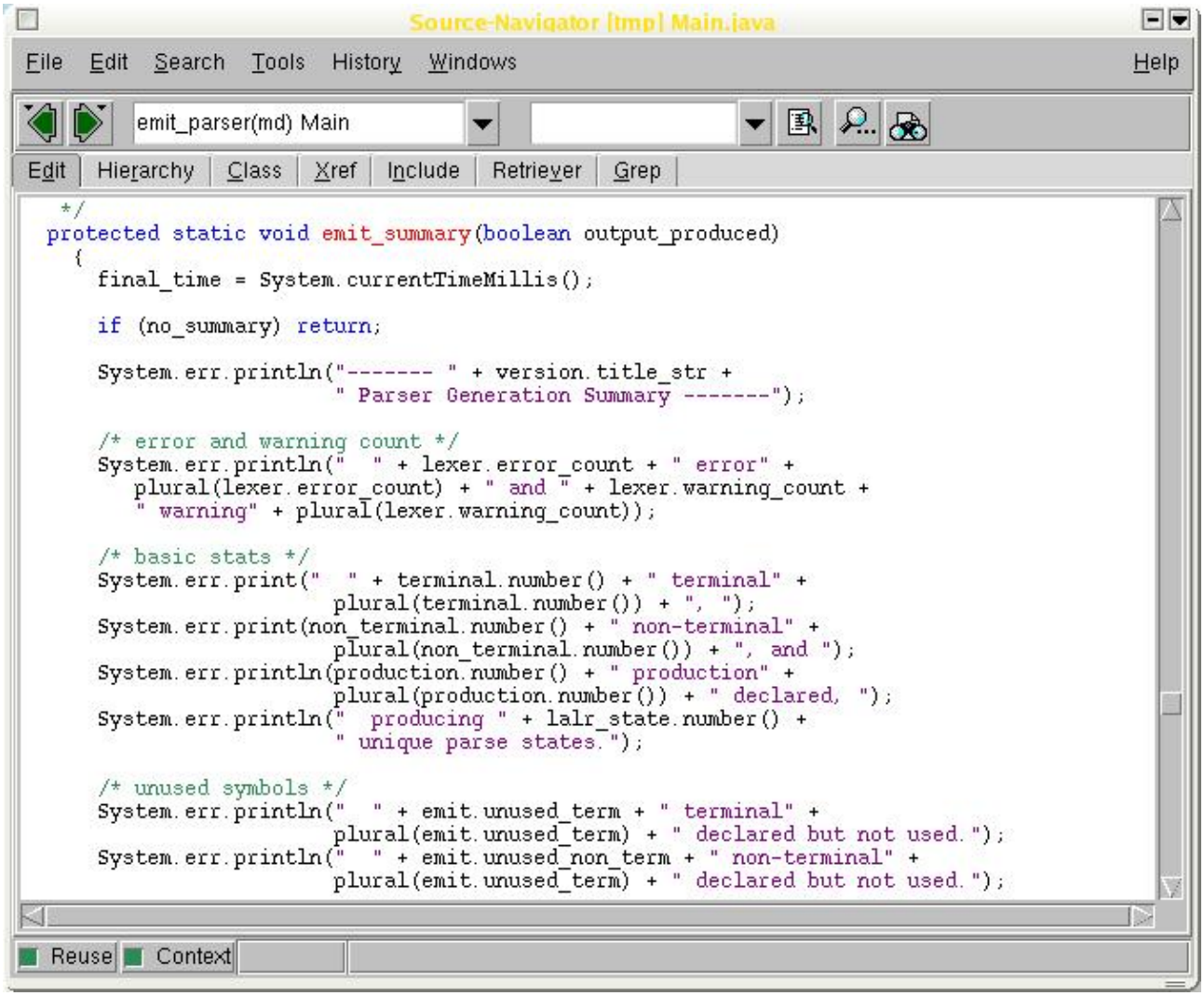
Benutzung des snavigator



Wird der snavigator zum ersten Mal aufgerufen, fragt er nach Verzeichnissen, die Quelldateien enthalten, wie der folgende Screenshot zeigt. Die unterstützen Sprachen umfassen unter anderem Java, C, C++, Tcl, Fortran, COBOL und Assemblerprogramme. Wenn der Quellcode angegeben wurde, baut er selber eine Projekt−Datenbank, die Verweis−Informationen, Klassenhierarchien, Datei−Abhängigkeiten und noch viel mehr umfaßt. Je nach Größe von deinem Projekt kann das ein bißchen dauern. Anschließend kannst du Anfragen an die Datenbank stellen und zusätzliche Informationen über den Code erfragen. Das Folgende präsentiert nur ein paar der Features, um dir eine Vorstellung zu geben. Eine illustrierte *Einleitung* und ein Referenzhandbuch befinden sich im Unterverzeichnis `html` deiner Installation.

Projekt−Management

Zum Paket gehört ein *Editor* mit Syntax−Highlighting, der auch zum ansprechenden Drucken genutzt werden kann. Der folgende Screenshot bildet das Hauptfenster des Editors ab. Er ist fast wie eine Entwicklungsumgebung, denn er wartet mit Sachen wie einer Debug−Möglichkeit, Projekt−Kompilierbefehlen, Versionskontrolle und ähnlichem auf.

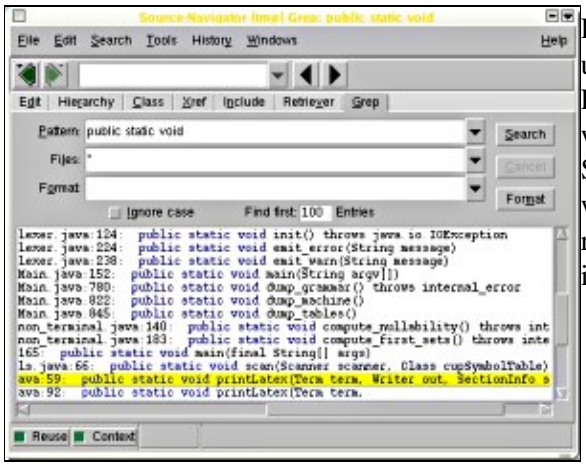


Insbesondere die großen grünen Pfeile in der Toolbar funktionieren genau wie in einem Web–Browser. Der Projekt–Editor erlaubt einem, die Datenbankinformationen zu kontrollieren (z.B. falls eine Datei gerade verändert wurde), Dateien von der Liste hinzuzufügen oder zu löschen und andere Verwaltungsaufgaben. Alle Dateien werden als ein großes Projekt behandelt. Wenn also Veränderungen geschehen, kannst du die Datenbankinformationen mit *Refresh Project* oder *Reparse Project* aktualisieren.
Im Editorfenster markiere etwas, etwa den Namen einer Funktion, so daß die markierte Region in gelb erscheint. Dann klicke mit der rechten Maustaste – du kannst entscheiden, ob du (a) die *Deklaration* von dem, was du markiert hast, sehen willst (z.B. eine Header–Datei) oder (b) die *Implementation* des markierten Symbols (z.B. eine `.cpp`–Datei) – und es gibt noch ein paar weitere nützliche Optionen.

Der Symbolbrowser

Das ist das erste Fenster, das nach dem Füllen der Projektdatenbank erscheint. Normalerweise enthält es Dateinamen, aber es kann auch Klassenmethoden, Funktionssymbole und ähnliches darstellen. Wenn auf einen Dateinamen geklickt wird, öffnet sich der Editor mit dieser Datei.

Das grep–Fenster

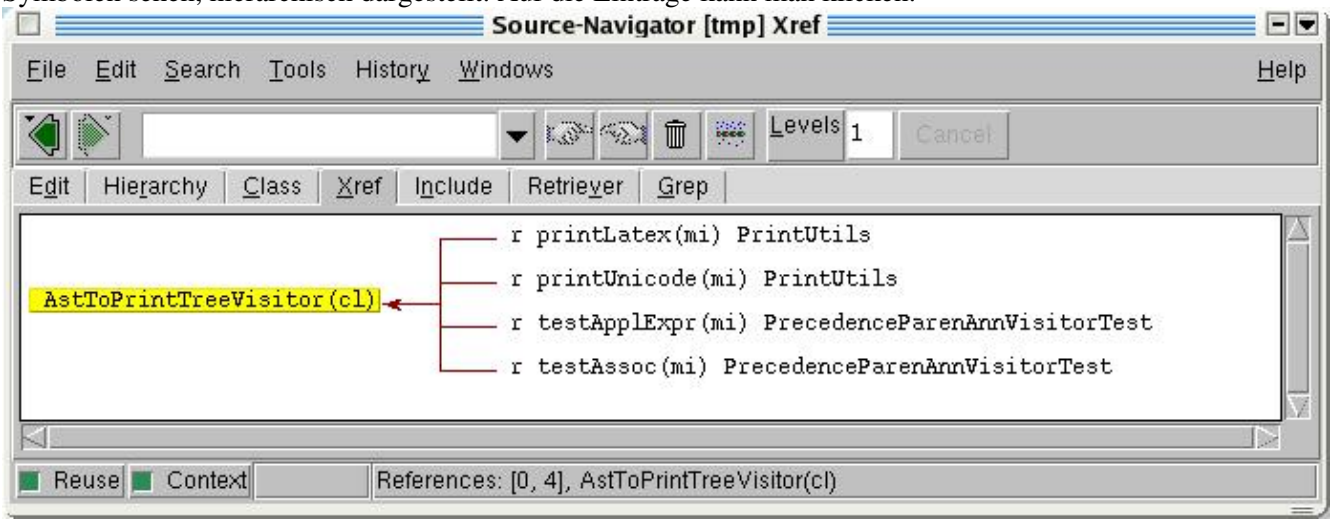


Das tut, was der Name sagt: Es bietet eine komfortable GUI, um durch alle beteiligten Quelldateien zu greppen. Passende Einträge sind markiert und verlinkt, der Quellcode kann so wie ein Haufen Webseiten betrachtet werden. Wie der Screenshot zeigt, kann die entsprechende Datei ausgewählt werden, und durch simples Klicken wird der Editor an der richtigen Stelle geöffnet. (Die gezeigte Suche erzeugt Treffer in vielen Java–Dateien :)

Das Xref–Fenster

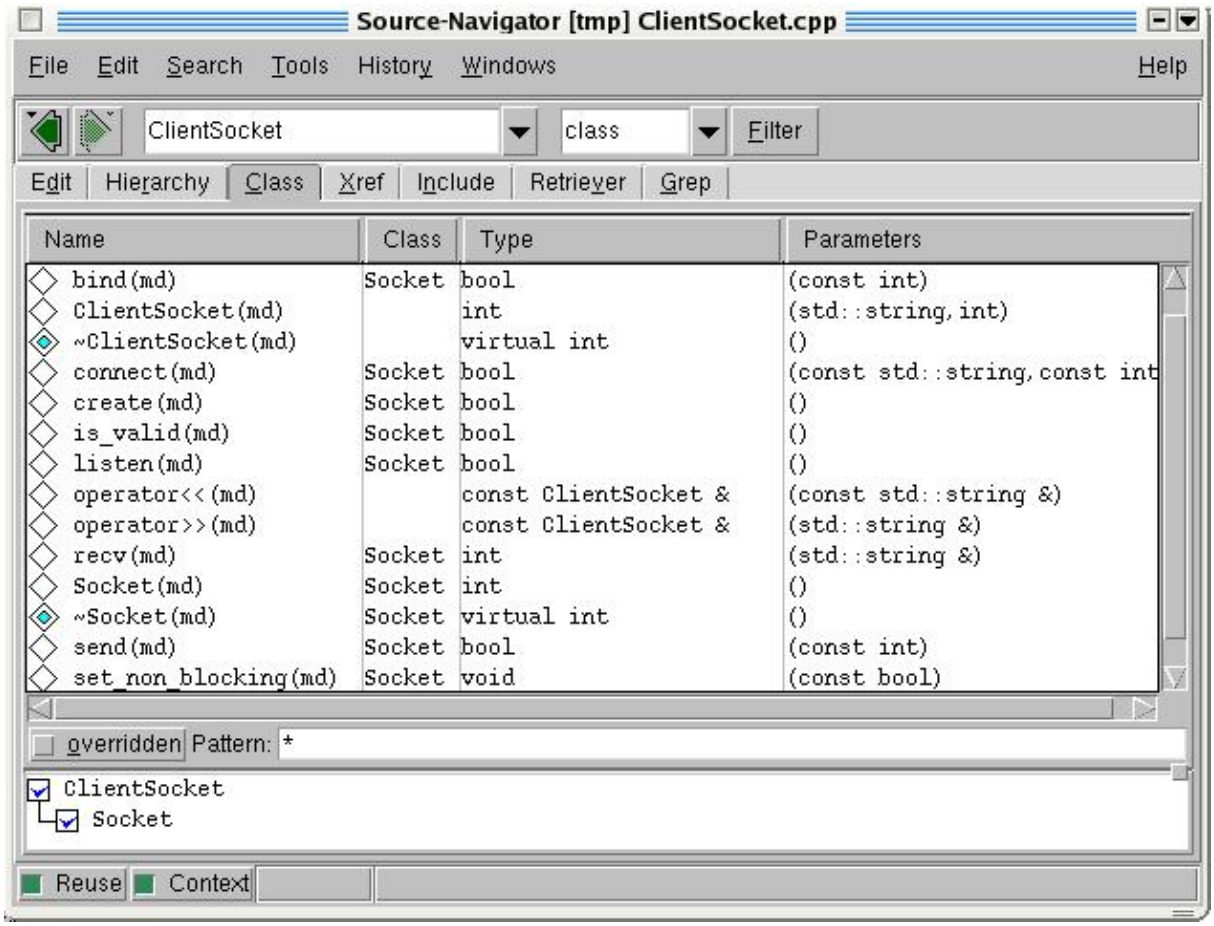
Hier haben wir eine Liste mit Querverweisen zu allen Symbolen, insbesondere kann man sehen, welche Methoden lesen (r), schreiben (w), usw. auf welchen Daten, und man kann die Beziehungen zwischen

Symbolen sehen, hierarchisch dargestellt. Auf die Einträge kann man klicken.



Das Klassenfenster

Dieses Interface sammelt alle nützliche Informationen, die man über Klassen in einer objektorientierten Sprache wissen will. Vor allem werden die Super– und Subklassen gezeigt sowie Attribut– und Methodennamen mit ihren Parametern. Zur Abwechslung zeigt das Fenster unten eine C++–Klasse `ClientSocket`, die von `Socket` erbt und ziemlich viele Methoden aufweist. Wieder gilt, daß du per Klick auf einen Eintrag ein Editorfenster an der entsprechenden Position öffnen kannst.



Alternativen

`cscope` ist ein interaktiver C−Quellcode−Browser für die Kommandozeile (er kann auch mit `C++` umgehen). Er hat einiges von der Funktionalität des snavigator, ein Screenshot ist hier. Tatsächlich ist er viel älter und ist in vielen sehr großen Projekten benutzt worden. Seine Homepage ist http://cscope.sourceforge.net. Aber du brauchst nicht einmal dorthin zu gehen − er ist in **vim** eingebaut und kann sehr ähnlich benutzt werden wie (g)vim in Kombination mit `tags`. Tippe einfach

```
:help cscope
```

in deiner vim−Sitzung, um die verfügbaren Optionen zu sehen. Es gibt noch ein paar von `cscope` abgeleitete Programme. Freescope ist ein `cscope`−Clone, dem etwas Funktionalität wie Symbol−Vervollständigung hinzugefügt wurde. Es gibt jetzt auch ein KDE−Frontend zu cscope namens `kscope`, das unter http://kscope.sourceforge.net liegt.

Zusammenfassung

Für jeden, der wenigstens zum Teil mit dem Re−Engineering oder der Integration von Quellcode beschäftigt ist, ist der snavigator ein sehr nützliches und mächtiges Werkzeug. Ich hatte einmal eine ältere Qt−Applikation, die leider nicht mit der aktuellen Version der Qt−Bibliothek lief. Indem ich die Fehlermeldungen las und ein bißchen mit dem snavigator arbeitete, fand ich bald heraus, daß nur die Parameterliste einer Funktion geändert werden mußte. Mit der klick−und−finde−Funktionalität *(click−and−locate)* war es möglich, das gesamte Softwarepaket in nur ein paar Minuten auf den aktuellen Stand zu bringen.

Der LinuxFocus Redaktion schreiben © Gerrit Renker "some rights reserved" see linuxfocus.org/license/ http://www.LinuxFocus.org	Autoren und Übersetzer: en −−> −− : Gerrit Renker <gerrit.renker(at)gmx.de> en −−> de: Viktor Horvath <ViktorHorvath/at/gmx.net>

2005−07−18, generated by lfparser_pdf version 2.51