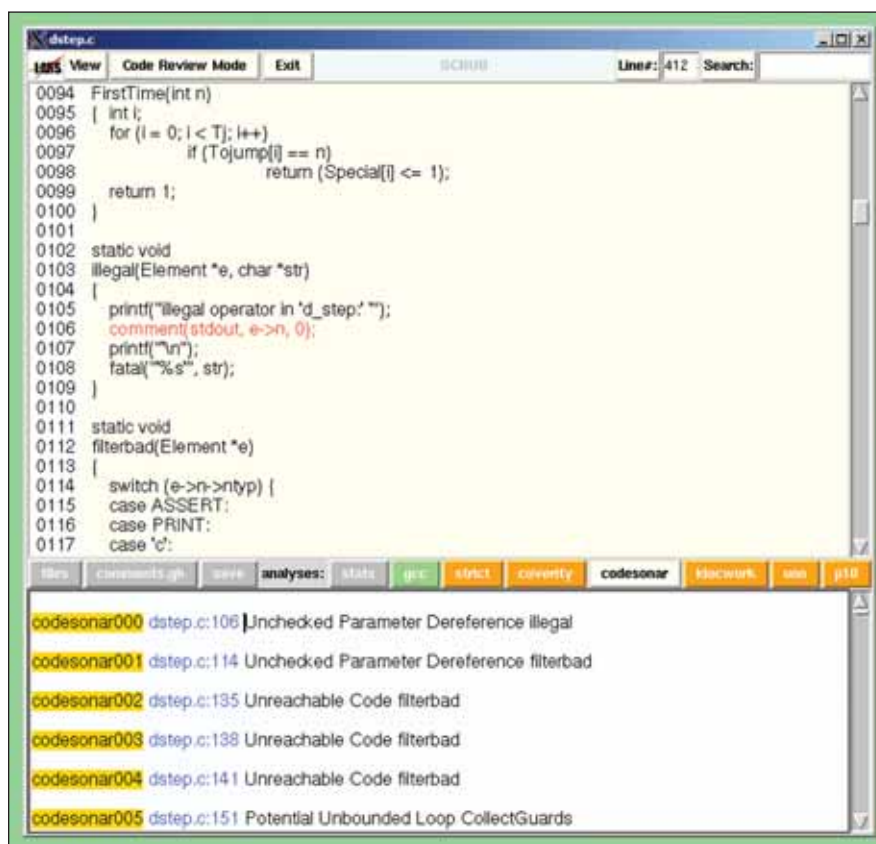


code with these tools can vary greatly. In each case, however, the tools produce results that would be difficult to realize with human code inspections alone. There is little overlap in the results produced by the different analyzers, and each analyzer used generally increases the effectiveness of the overall effort. The SCRUB tool allows all reports to be accessed through a single, uniform interface (see figure) that facilitates browsing code and reports. Improvements over existing software include significant simplification, and leveraging of a range of commercial, static source code analyzers in a single, uniform framework.

The tool runs as a small stand-alone application, avoiding the security problems related to tools based on Web-browsers. A developer or reviewer, for instance, must have already obtained access rights to a code base before that code can be browsed and reviewed with the SCRUB tool. The tool cannot open any files or folders to which the user does not already have access. This means that the tool does not need to enforce or administer any additional security policies. The analysis results presented through the SCRUB tool's user interface are always computed off-line, given that, especially for larger projects, this computation can take longer than appropriate for interactive tool use.

The recommended code review process that is supported by the SCRUB tool consists of three phases: Code Review, Developer Response, and Closeout Resolution. In the Code Review phase, all tool-based analysis reports are generated, and specific comments from expert code reviewers are entered into the SCRUB



The screenshot shows the SCRUB User Interface. The top window displays C code from a file named 'dstep.c'. The code includes a function 'FirstTime' and several static void functions: 'illegal', 'filterbad', and a switch statement. Below the code editor, a list of analysis results is shown, including 'codesonar000' through 'codesonar005' with their respective error messages and line numbers.

```

0094 FirstTime(int n)
0095 | int i;
0096 | for (i = 0; i < T; i++)
0097 |     if (Tojump[i] == n)
0098 |         return (Special[i] <= 1);
0099 |     return 1;
0100 |
0101 |
0102 static void
0103 illegal(Element *e, char *str)
0104 |
0105 |     printf("illegal operator in 'd_step:'.");
0106 |     comment(stdout, e->n, 0);
0107 |     printf("\n");
0108 |     fatal("%s", str);
0109 |
0110 |
0111 static void
0112 filterbad(Element *e)
0113 |
0114 |     switch (e->n->ntyp) {
0115 |     case ASSERT:
0116 |     case PRINT:
0117 |     case 'c':

```

codesonar000 dstep.c:106 Unchecked Parameter Dereference illegal  
codesonar001 dstep.c:114 Unchecked Parameter Dereference filterbad  
codesonar002 dstep.c:135 Unreachable Code filterbad  
codesonar003 dstep.c:138 Unreachable Code filterbad  
codesonar004 dstep.c:141 Unreachable Code filterbad  
codesonar005 dstep.c:151 Potential Unbounded Loop CollectGuards

SCRUB User Interface is shown when it is opened in local mode.

tool. In the second phase, Developer Response, the developer is asked to respond to each comment and tool-report that was produced, either agreeing or disagreeing to provide a fix that addresses the issue that was raised. In the third phase, Closeout Resolution, all disagreements are discussed in a meeting of all parties involved, and a resolution is made for all disagreements. The first two phases generally take

one week each, and the third phase is concluded in a single closeout meeting.

*This work was done by Gerard J. Holzmann of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).*

*This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at [danielb@caltech.edu](mailto:danielb@caltech.edu). Refer to NPO-46817.*

## Numerical Mean Element Orbital Analysis With Morbiter

*NASA's Jet Propulsion Laboratory, Pasadena, California*

The Morbiter software numerically averages an osculating orbit's equations of motion (EOM) to arrive at the mean orbit's EOMs, which are then numerically propagated to obtain the long-term orbital ephemerides. The long-term evolution characteristics, and stability, of an orbit are best characterized using a mean element propagation of the perturbed, two-body variational equations of motion. The average process eliminates short period terms, leaving only secular and long period effects. Doing this avoids the Fourier series expansions and truncations required by the traditional analytic methods.

The numerical methods require no analytic approximation, and the averaging theory and software implementation work at any solar system body. JPL's Monte mission analysis and navigation software was used as the underlying trajectory system (to the extent possible) for this innovation.

Morbiter is a package of Python scripts that implement the algorithms, and uses Monte for basic astrodynamics constructs and functions such as trajectories, ephemerides, coordinate systems, astrodynamics constants, and, in most cases, the perturbation acceleration methods. Python is an interpreted language that

provides an ideal platform for rapid development of algorithms; however, there is a performance penalty for using Python script-based applications. An end-user, future version of Morbiter that is fully compiled will not suffer from this speed penalty; development of this version is planned to begin in late FY '10.

*This work was done by Todd A. Ely of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact [iaoffice@jpl.nasa.gov](mailto:iaoffice@jpl.nasa.gov).*

*This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at [danielb@caltech.edu](mailto:danielb@caltech.edu). Refer to NPO-47212.*