Zeus Applicability to CZT Initiative

Sean R. Travis
Department of Computer Science
University of Virginia, Charlottesville, VA 22904
srt3k@virginia.edu

Zeus features compared to features described in the document, *CZT Requirements* (Dec-2001)

Each section below is labeled by a section number from the document. The corresponding Zeus features are compared to the features described in the sections.

[1.2]

Zeus can be easily modified to use any representation of the Z notation, because it already includes most (if not all) of the Z notation as referenced in Spivey's reference manual. It already can convert the Z notation between LaTeX representation and WYSIWYG representation (i.e. FrameMaker's representation). It already converts between the WYSIWYG representation and the Z/EVES 2.1 representation (an XML representation). Zeus, using FrameMaker, can convert the WYSIWYG into RTF, PDF, HTML, and other formats.

[1.3]

Zeus is currently part of a major research project for advancing tool support in the formal methods area. It is developed mainly by a full-time staff member.

Zeus source code will likely be made available.

[1.4]

Zeus was developed on the Windows platform, keeping its robust features available to even minimal Windows 95 environments. It is written in C++, so could reasonably be ported to UNIX platforms or converted to Java.

[3.1]

Zeus is implemented with a simple abstract data structure for the structures in Z. Basically, this data structure's interface is a ZSpec object. A ZSpec object hides the design decisions of manipulating ZDocument objects. ZDocuments are major sections of a specification which contain Z paragraphs such as schemas. FrameMaker files hold the actual text of ZDocuments in Zeus, and a ZDocument object (used by a ZSpec object) holds the needed information (such as location within the specification). Each ZDocument hides the design decisions of manipulating ZParagraph objects. A ZParagraph is a schema, a generic definition, an axiomatic description, or a single Z expression not used in the other paragraphs.

Zeus uses another class to abstract away the manipulating of a ZSpec in general, hiding such things as how to insert and delete new Z paragraphs (in the internal data structures) and how to set which Z paragraphs have been syntax and type checked.

Together these four classes keep track of where Z paragraphs are in the original (source) documents and provide an API for other tools to use.

[3.3]

Zeus already has a good GUI for the Z notation--one of the primary goals for Zeus! Modifying FrameMaker gives Zeus a powerful document-preparation interface. By "reusing" FrameMaker, Zeus provides familiar saving, printing, editing of natural language and, editing of graphics (such as tables), and even spell-checking.

WYSIWYG editing is one of the other primary goals of Zeus. One manipulates the Z notation directly--no LaTeX knowledge is needed. The schema graphics and special Z symbols are inserted directly by clicking on them from a menu or toolbar. Graphics sizes are adjusted automatically as material is entered or deleted.

In order to provide the Z symbols, we created our own Z font using Fontographer. Conforming this font to the standard Unicode should be straightforward.

Another major goal of Zeus has been to connect the editing of Z specifications with a syntax checker and a theorem prover; Zeus connects to Z/EVES for both of these purposes. Currently, only syntax and type checking are connected, but we have nearly completed work on connecting the theorem proving to Zeus.

There are also navigational functions, such as double-clicking on an error message to go to the line with the error and double-clicking on a tree view to go to a specific Z paragraph in the specification.

More features could easily be added to the existing object-oriented source code.

[4.1]

As described in the point above, Zeus already provides the first 3 GUI requirements and is near to fulfilling the third requirement in the second list.

Zeus runs in Windows. Zeus could reasonably be ported to XWindows for reasons stated earlier.

Some Related Papers

- Knight, John C., Kimberly S. Hanks, and Sean R. Travis
  Tool Support for Production Use of Formal Techniques
  International Symposium on Software Reliability Engineering, Hong Kong (November 2001)
  (http://www.cs.virginia.edu/~jck/publications/issre.2001.fm.pdf)
- Knight, J.C., C.L. DeJong, M.S. Gibble, and L.G. Nakano
  Why Are Formal Methods Not Used More Widely?
  Fourth NASA Formal Methods Workshop, Hampton, VA (September 1997)
  (http://www.cs.virginia.edu/~jck/publications/lfm.97.pdf)
- Hanks, Kimberly S., John C. Knight, and Elisabeth A. Strunk
  Erroneous Requirements: A Linguistic Basis for Their Occurrence and an Approach to Their Reduction, Software Engineering Workshop, NASA Goddard Space Flight Center (December 2001)
  (http://www.cs.virginia.edu/~jck/publications/gsfc.sel.2001.pdf)
- Hanks, Kimberly S., John C. Knight, and Elisabeth A. Strunk
  A Linguistic Analysis of Requirements Errors and Its Application.
  Technical Report CS-2001-30 (November 2001)
  (http://www.cs.virginia.edu/~jck/publications/icse.2002.pdf)