An Improved Debugging System

Mary Lewis

Why Improve?

Nearly half of programming time is spent debugging. The advancements we have seen in debugging tools in the last 20 years don't reflect how important it is.

Current Debugging Technology:

- Breakpoint debugging
- Algorithmic debugging
- "printf debugging"

Problems:

- These methods can be time-consuming.
- Errors can be difficult to find using these methods.

Proposed Alternative:

I propose a system you can run your program through that will produce a trace of your program.

The way you navigate though this trace will resemble the way you navigate through the help that comes with some software.

In short, this will be hyperlinked, searchable documentation that tells the programmer what the code does in English sentences.

For example…

Oftentimes debugging involves searching for code that syntactically but not semantically correct.

References:

Gerard Vink. **Trends in Debugging Technology**. In Embedded Systems Conference East, 1998.

M. Snyder and J. Blandy. **The Heisenburg Debugging Technology**. CYGNUS SOLUTIONS, 1999.

Josep Silva. **Debugging Techniques for Declarative Languages: Profiling, program slicing and algorithmic debugging**. In AI Communications, Volume 21, Issue 1, Pages 91-92, 2008.

The GDB Developers. **GDB: The GNU Project Debugger**. http://www.gnu.org/software/gdb/

How:

- Use current debugging technology to gather information during runtime
- Anticipate or detect when the variable changes
- Use regular expressions to create natural language
- Use free help authoring tools to produce HTML help files

Problems:

- Time
- Memory

These problems can possibly be alleviated through the use of checkpoints.