DEFINITION

debugging

By **Matt Heusser,** Excelon Development

What is debugging?

Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works.

In software development, the debugging process begins when a developer locates a code error in a computer program and is able to reproduce it. Debugging is part of the software testing process and is an integral part of the entire software development lifecycle.

In hardware development, the debugging process looks for hardware components that are not installed or configured correctly. For example, an engineer might run a JTAG connection test to debug connections on an integrated circuit.

How debugging works in software

The debugging process starts as soon as code is written and continues in successive stages as code is combined with other units of programming to form a software product. In a large program that has thousands and thousands of lines of code, the debugging process can be made easier by using strategies such as unit tests, code reviews and pair programming.

To identify bugs, it can be useful to look at the code's logging and use a stand-alone debugger tool or the debug mode of an integrated development environment (IDE). It can be helpful at this point if the developer is familiar with standard error messages. If developers aren't commenting adequately when writing code, even the cleanest code can be a challenge for someone to debug.

In some cases, the module that presents the problem is obvious, while the line of code itself is not. In that case, unit tests -- such as JUnit and xUnit, which allow the programmer to run a specific function with specific inputs -- can be helpful in debugging.

The standard practice is to set up a "breakpoint" and run the program until that breakpoint, at which time program execution stops. The debugger component of an IDE provides the programmer with the capability to view memory and see variables, run the program to the next breakpoint, execute the next line of code, and, in some cases, change the value of variables or even change the contents of the line of code about to be executed.

Why is debugging important?

Debugging is an important part of determining why an operating system, application or program is misbehaving. Even if developers use the same coding standard, it's still likely that a new software program will still have bugs. In many cases, the process of debugging a new software program can take more time than it took to write the program.

Invariably, the bugs in software components that get the most use are found and fixed first.

Debugging vs. testing

Debugging and testing are complementary processes. The purpose of testing is to identify what happens when there is a mistake in a program's source code. The purpose of debugging is to locate and fix the mistake.

The testing process does not help the developer figure out what the coding mistake is -- it simply reveals what effects the coding error has on the program. Once the mistake has been error identified, debugging helps the developer determine the cause of the error so it can be fixed.

Common coding error examples

Some examples of common coding errors include the following:

- Syntax error
- Runtime error
- Semantic error
- Logic error
- Disregarding adopted conventions in the coding standard
- Calling the wrong function
- Using the wrong variable name in the wrong place
- Failing to initialize a variable when absolutely required
- Skipping a check for an error return

Debugging strategies

Source code analyzers, which include security, common code errors and complexity analyzers, can be helpful in debugging. A complexity analyzer can find intricate modules that are hard to understand and test. Other debugging strategies include the following:

- **Static analysis.** The developer examines the code without executing the program.

statements and monitors flow.

- **Remote debugging.** The developer's debugger runs on a different system than the program that is being debugged.
- **Post-mortem debugging.** The developer only stops to debug the program if it experiences fatal exceptions.

Debugging tools

A debugger is a software tool that can help the software development process by identifying coding errors at various stages of the operating system or application development.

Some debuggers will analyze a test run to see what lines of code were not executed. Other debugging tools provide simulators that allow the programmer to model how an app will display and behave on a given operating system or computing device.

Many open-source debugging tools and scripting languages do not run in an IDE, so they require a more manual approach to debugging. **For example, USB Debugging allows an Android device to communicate with a computer running the Android SDK.**

In this situation, the developer might debug a program by dropping values to a log, creating extensive print statements to monitor code execution or implement hard-coded wait commands that will simulate a breakpoint by waiting for keyboard input at specific intervals.

Challenges of debugging

The debugging process can be quite difficult and require as much work -- if not more -- than writing the code to begin with. The process can be especially challenging when:

- The negative effect of the coding error is clear, but the cause is not.
- The negative effect of the coding error is difficult to reproduce -- for example, when web content contains drop-down menus.

- Dependencies are not clear, so fixing a coding error in one part of the program accidentally introduces new errors in other parts of the program.

History of debugging

The use of the word bug as a synonym for error originated in engineering. The term's application to computing and the inspiration for using the word debugging as a synonym for troubleshooting has been attributed to Admiral Grace Hopper, a pioneer in computer programming, who was also known for her dry sense of humor. When an actual bug (a moth) got caught between electrical relays and caused a problem in the U.S. Navy's first computer, Admiral Hopper and her team "debugged" the computer and saved the moth. It now resides in the Smithsonian Museum.

COURTESY OF THE NAVAL SURFACE WARFARE CENTER

Photo of moth taped to log book.

The first computer bug is taped to a log book that now resides at the Smithsonian.

This was last updated in November 2022

Continue Reading About debugging

- How do you debug a Kubernetes service deployment?