While Admiral Grace Hopper was working on a Mark II computer at Harvard University, her associates discovered a moth stuck in a relay and thereby impeding operation, whereupon she remarked that they were "debugging" the system. However, the term "bug", in the sense of "technical error", dates back at least to 1878 and Thomas Edison (see software bug for a full discussion). Similarly, the term "debugging" seems to have been used as a term in aeronautics before entering the world of computers. Indeed, in an interview Grace Hopper remarked that she was not coining the term[*citation needed*]. The moth fit the already existing terminology, so it was saved. A letter from J. Robert Oppenheimer (director of the WWII atomic bomb "Manhattan" project at Los Alamos, NM) used the term in a letter to Dr. Ernest Lawrence at UC Berkeley, dated October 27, 1944,[2] regarding the recruitment of additional technical staff.

The Oxford English Dictionary entry for "debug" quotes the term "debugging" used in reference to airplane engine testing in a 1945 article in the Journal of the Royal Aeronautical Society. An article in "Airforce" (June 1945 p. 50) also refers to debugging, this time of aircraft cameras. Hopper's bug was found on September 9, 1947. The term was not adopted by computer programmers until the early 1950s. The seminal article by Gill[3] in 1951 is the earliest in-depth discussion of programming errors, but it does not use the term "bug" or "debugging". In the ACM's digital library, the term "debugging" is first used in three papers from 1952 ACM National Meetings.[4][5][6] Two of the three use the term in quotation marks. By 1963 "debugging" was a common enough term to be mentioned in passing without explanation on page 1 of the CTSS manual.[7]

Debugging ranges in complexity from fixing simple errors to performing lengthy and tiresome tasks of data collection, analysis, and scheduling updates. The debugging skill of the programmer can be a major factor in the ability to debug a problem, but the difficulty of software debugging varies greatly with the complexity of the system, and also depends, to some extent, on the programming language(s) used and the available tools, such as *debuggers*. Debuggers are software tools which enable the programmer to monitor the execution of a program, stop it, restart it, set breakpoints, and change values in memory. The term *debugger* can also refer to the person who is doing the debugging.

Generally, high-level programming languages, such as Java, make debugging easier, because they have features such as exception handling and type checking that make real sources of erratic behaviour easier to spot. In programming languages such as C or assembly, bugs may cause silent problems such as memory corruption, and it is often difficult to see where the initial problem happened. In those cases, memory debugger tools may be needed.

In certain situations, general purpose software tools that are language specific in nature can be very useful. These take the form of *static code analysis tools*. These tools look for a very specific set of known problems, some common and some rare, within the source code. All such issues detected by these tools would rarely be picked up by a compiler or interpreter, thus they are not syntax checkers, but more semantic checkers. Some tools claim to be able to detect 300+ unique problems. Both commercial and free tools exist in various languages. These tools can be extremely useful when checking very large source trees, where it is impractical to do code walkthroughs. A typical example of a problem detected would be a variable dereference that occurs *before* the variable is assigned a value. As another example, some such tools perform strong type checking when the language does

not require it. Thus, they are better at locating likely errors, versus actual errors. As a result, these tools have a reputation of false positives. The old Unix *lint* program is an early example.

For debugging electronic hardware (e.g., computer hardware) as well as low-level software (e.g., BIOSes, device drivers) and firmware, instruments such as oscilloscopes, logic analyzers or in-circuit emulators (ICEs) are often used, alone or in combination. An ICE may perform many of the typical software debugger's tasks on low-level software and firmware.

Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.

Debugging tactics can involve interactive debugging, control flow analysis, unit testing, integration testing, log file analysis, monitoring at the application or system level, memory dumps, and profiling.