

Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Also, specific user environment and usage history can make it difficult to reproduce the problem. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. Use of a static code analysis tool can help detect some possible problems. It is usually easier to code in "high-level" languages than in "low-level" ones. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Scripting and breakpointing is also part of this process. Also, specific user environment and usage history can make it difficult to reproduce the problem. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. One approach popular for requirements analysis is Use Case analysis. One approach popular for requirements analysis is Use Case analysis. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Following a consistent programming style often helps readability.