

A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Techniques like Code refactoring can enhance readability. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. In 1206, the Arab engineer Al-Jazari invented a programmable drum machine where a musical mechanical automaton could be made to play different rhythms and drum patterns, via pegs and cams. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. 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. Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. Following a consistent programming style often helps readability. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. It is usually easier to code in "high-level" languages than in "low-level" ones.