It is usually easier to code in "high-level" languages than in "low-level" ones. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. 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. One approach popular for requirements analysis is Use Case analysis. Programs were mostly entered using punched cards or paper tape. Many applications use a mix of several languages in their construction and use. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Techniques like Code refactoring can enhance readability. Programs were mostly entered using punched cards or paper tape. Normally the first step in debugging is to attempt to reproduce the problem. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. 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. 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. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. 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. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language.