

One approach popular for requirements analysis is Use Case analysis. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Many applications use a mix of several languages in their construction and use. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. 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. Following a consistent programming style often helps readability. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Code-breaking algorithms have also existed for centuries. Integrated development environments (IDEs) aim to integrate all such help. 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. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. Following a consistent programming style often helps readability. Use of a static code analysis tool can help detect some possible problems. However, readability is more than just programming style. 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. 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.