Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Different programming languages support different styles of programming (called programming paradigms). 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. 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Techniques like Code refactoring can enhance readability. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Code-breaking algorithms have also existed for centuries. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. One approach popular for requirements analysis is Use Case analysis. Following a consistent programming style often helps readability. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. It affects the aspects of quality above, including portability, usability and most importantly maintainability. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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.