It is usually easier to code in "high-level" languages than in "low-level" ones. Whatever the approach to development may be, the final program must satisfy some fundamental properties. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. There are many approaches to the Software development process. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Also, specific user environment and usage history can make it difficult to reproduce the problem. It is very difficult to determine what are the most popular modern programming languages. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Techniques like Code refactoring can enhance readability. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Programming languages are essential for software development. Also, specific user environment and usage history can make it difficult to reproduce the problem. Code-breaking algorithms have also existed for centuries. One approach popular for requirements analysis is Use Case analysis. It affects the aspects of quality above, including portability, usability and most importantly maintainability. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Computer programmers are those who write computer software. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Use of a static code analysis tool can help detect some possible problems. There exist a lot of different approaches for each of those tasks.