Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. 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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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). Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. It is very difficult to determine what are the most popular modern programming languages. Integrated development environments (IDEs) aim to integrate all such help. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. One approach popular for requirements analysis is Use Case analysis.