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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. However, readability is more than just programming style. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. 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. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. One approach popular for requirements analysis is Use Case analysis. 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). 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. 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. It is very difficult to determine what are the most popular modern programming languages. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. Integrated development environments (IDEs) aim to integrate all such help. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Following a consistent programming style often helps readability. 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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.