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. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. There exist a lot of different approaches for each of those tasks. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Programming languages are essential for software development. Whatever the approach to development may be, the final program must satisfy some fundamental properties. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. Following a consistent programming style often helps readability. The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. 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. Programming languages are essential for software development. Scripting and breakpointing is also part of this process. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.