Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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 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. Programs were mostly entered using punched cards or paper tape. 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. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. Following a consistent programming style often helps readability. However, readability is more than just programming style. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Following a consistent programming style often helps readability. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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). The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. It affects the aspects of quality above, including portability, usability and most importantly maintainability. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. Scripting and breakpointing is also part of this process. Different programming languages support different styles of programming (called programming paradigms). Also, specific user environment and usage history can make it difficult to reproduce the problem. Programming languages are essential for software development. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and 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.