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. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. 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. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Different programming languages support different styles of programming (called programming paradigms). Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. 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. Use of a static code analysis tool can help detect some possible problems. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Ideally, the programming language best suited for the task at hand will be selected. Scripting and breakpointing is also part of this process. Programs were mostly entered using punched cards or paper tape. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses.