Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. 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 study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. It affects the aspects of quality above, including portability, usability and most importantly maintainability. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. 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. There exist a lot of different approaches for each of those tasks. Following a consistent programming style often helps readability. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. Scripting and breakpointing is also part of this process. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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.