

Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. 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. Programs were mostly entered using punched cards or paper tape. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. There are many approaches to the Software development process. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Programs were mostly entered using punched cards or paper tape. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Following a consistent programming style often helps readability. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. 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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.