

It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Different programming languages support different styles of programming (called programming paradigms). 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. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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). Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Many applications use a mix of several languages in their construction and use. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. 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. Programming languages are essential for software development. Code-breaking algorithms have also existed for centuries. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Techniques like Code refactoring can enhance readability. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Following a consistent programming style often helps readability. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). There are many approaches to the Software development process. 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.