

Programmable devices have existed for centuries. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Use of a static code analysis tool can help detect some possible problems. 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 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. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. 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. 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. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Code-breaking algorithms have also existed for centuries. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. It is usually easier to code in "high-level" languages than in "low-level" ones. 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.