

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. Different programming languages support different styles of programming (called programming paradigms). 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. 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. Whatever the approach to development may be, the final program must satisfy some fundamental properties. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. However, readability is more than just programming style. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. There are many approaches to the Software development process. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). 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. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Many applications use a mix of several languages in their construction and use. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. Use of a static code analysis tool can help detect some possible problems. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.