

Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. Many applications use a mix of several languages in their construction and use. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. 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 Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. One approach popular for requirements analysis is Use Case analysis. Scripting and breakpointing is also part of this process. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. In 1206, the Arab engineer Al-Jazari invented a programmable drum machine where a musical mechanical automaton could be made to play different rhythms and drum patterns, via pegs and cams. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. 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. Use of a static code analysis tool can help detect some possible problems.