

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. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Programmable devices have existed for centuries. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. One approach popular for requirements analysis is Use Case analysis. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. 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. There are many approaches to the Software development process. 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Following a consistent programming style often helps readability. 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 applications use a mix of several languages in their construction and use. 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).