However, readability is more than just programming style. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. 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). 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. Integrated development environments (IDEs) aim to integrate all such help. 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. Following a consistent programming style often helps readability. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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). 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. One approach popular for requirements analysis is Use Case analysis. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Programmable devices have existed for centuries. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Also, specific user environment and usage history can make it difficult to reproduce the problem. Many applications use a mix of several languages in their construction and use. 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.