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. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. It is very difficult to determine what are the most popular modern programming languages. However, readability is more than just programming style. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. There are many approaches to the Software development process. 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. 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. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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). Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. It is usually easier to code in "high-level" languages than in "low-level" ones. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. However, readability is more than just programming style. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Programming languages are essential for software development. 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). One approach popular for requirements analysis is Use Case analysis.