Unreadable code often leads to bugs, inefficiencies, and duplicated code. Many applications use a mix of several languages in their construction and use. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Use of a static code analysis tool can help detect some possible problems. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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 text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. 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. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Code-breaking algorithms have also existed for centuries. Ideally, the programming language best suited for the task at hand will be selected. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. Techniques like Code refactoring can enhance readability. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. Computer programmers are those who write computer software.