The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. Scripting and breakpointing is also part of this process. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. It is usually easier to code in "high-level" languages than in "low-level" ones. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. 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. However, readability is more than just programming style. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Normally the first step in debugging is to attempt to reproduce the problem. 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). Computer programmers are those who write computer software. Different programming languages support different styles of programming (called programming paradigms). One approach popular for requirements analysis is Use Case analysis. 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. There exist a lot of different approaches for each of those tasks. Different programming languages support different styles of programming (called programming paradigms). Computer programmers are those who write computer software. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. It is very difficult to determine what are the most popular modern programming languages. 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. Programs were mostly entered using punched cards or paper tape. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.