One approach popular for requirements analysis is Use Case analysis. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Unreadable code often leads to bugs, inefficiencies, and duplicated code. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. 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. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. 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. 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. 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. 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). Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. Ideally, the programming language best suited for the task at hand will be selected. However, readability is more than just programming style. Unreadable code often leads to bugs, inefficiencies, and duplicated code. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). It is usually easier to code in "high-level" languages than in "low-level" ones. 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. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.