

The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. 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. There exist a lot of different approaches for each of those tasks. 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). Also, specific user environment and usage history can make it difficult to reproduce the problem. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Use of a static code analysis tool can help detect some possible problems. However, readability is more than just programming style. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. It affects the aspects of quality above, including portability, usability and most importantly maintainability. However, readability is more than just programming style. 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. Scripting and breakpointing is also part of this process. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. There exist a lot of different approaches for each of those tasks. 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. Programs were mostly entered using punched cards or paper tape. Techniques like Code refactoring can enhance readability. 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Whatever the approach to development may be, the final program must satisfy some fundamental properties.