

The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Code-breaking algorithms have also existed for centuries. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Scripting and breakpointing is also part of this process. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Ideally, the programming language best suited for the task at hand will be selected. 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. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. There exist a lot of different approaches for each of those tasks. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. Techniques like Code refactoring can enhance readability. 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. 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. 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. Scripting and breakpointing is also part of this process. Techniques like Code refactoring can enhance readability. 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.