For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). It is usually easier to code in "high-level" languages than in "low-level" ones. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Normally the first step in debugging is to attempt to reproduce the problem. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. There are many approaches to the Software development process. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. 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). The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. Also, specific user environment and usage history can make it difficult to reproduce the problem. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Scripting and breakpointing is also part of this process. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. 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. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.