Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. 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. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. 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. One approach popular for requirements analysis is Use Case analysis. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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). Normally the first step in debugging is to attempt to reproduce the problem. It is usually easier to code in "high-level" languages than in "low-level" ones. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Code-breaking algorithms have also existed for centuries. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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 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. 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. Programmable devices have existed for centuries. 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.