In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Allen Downey, in his book How To Think Like A Computer Scientist, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Following a consistent programming style often helps readability. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Ideally, the programming language best suited for the task at hand will be selected. Code-breaking algorithms have also existed for centuries. 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. Computer programmers are those who write computer software. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. 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. 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. Many applications use a mix of several languages in their construction and use. 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. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. 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. However, readability is more than just programming style. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. One approach popular for requirements analysis is Use Case analysis. It is usually easier to code in "high-level" languages than in "low-level" ones.