Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Programmable devices have existed for centuries. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. 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. 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. However, readability is more than just programming style. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Different programming languages support different styles of programming (called programming paradigms). The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Code-breaking algorithms have also existed for centuries. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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 compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. Many applications use a mix of several languages in their construction and use. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.