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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Use of a static code analysis tool can help detect some possible problems. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. 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. 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. 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. Many applications use a mix of several languages in their construction and use. 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. 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. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. 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. Ideally, the programming language best suited for the task at hand will be selected. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). 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. Different programming languages support different styles of programming (called programming paradigms). Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.