Techniques like Code refactoring can enhance readability. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). 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. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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 the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Normally the first step in debugging is to attempt to reproduce the problem. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). They are the building blocks for all software, from the simplest applications to the most sophisticated ones. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. Normally the first step in debugging is to attempt to reproduce the problem. 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. Following a consistent programming style often helps readability. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.