It is usually easier to code in "high-level" languages than in "low-level" ones.  
Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
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.  
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.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
To produce machine code, the source code must either be compiled or transpiled.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.  
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.  
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.  
 Programmable devices have 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.  
Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly.