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.  
There are many approaches to the Software development process.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
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.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
 Following a consistent programming style often helps readability.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic.  
The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.  
 Programs were mostly entered using punched cards or paper tape.  
 Different programming languages support different styles of programming (called programming paradigms).  
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.  
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.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.