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..  
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.  
 Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.  
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.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
 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.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research.  
Ideally, the programming language best suited for the task at hand will be selected.  
There are many approaches to the Software development process.  
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.  
It is usually easier to code in "high-level" languages than in "low-level" ones.