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.  
For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software.  
There exist a lot of different approaches for each of those tasks.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
This is interpreted into machine code.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software.  
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.  
Scripting and breakpointing is also part of this process.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.