It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.  
This is interpreted into machine code.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
One approach popular for requirements analysis is Use Case analysis.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
Also, those involved with software development may at times engage in reverse engineering, which is the practice of seeking to understand an existing program so as to re-implement its function in some way.  
The source code of a program is written in one or more languages that are intelligible to programmers, rather than machine code, which is directly executed by the central processing unit.  
Normally the first step in debugging is to attempt to reproduce the problem.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.