Ideally, the programming language best suited for the task at hand will be selected.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
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.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
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).  
There exist a lot of different approaches for each of those tasks.  
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.  
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.  
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.  
 Following a consistent programming style often helps readability.  
 These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics.  
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.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Normally the first step in debugging is to attempt to reproduce the problem.