This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
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.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
Scripting and breakpointing is also part of this process.  
Ideally, the programming language best suited for the task at hand will be selected.  
Normally the first step in debugging is to attempt to reproduce the problem.  
To produce machine code, the source code must either be compiled or transpiled.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
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.  
 Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display.  
To produce machine code, the source code must either be compiled or transpiled.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
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.