Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
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.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
Scripting and breakpointing is also part of this process.  
The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.  
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.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
One approach popular for requirements analysis is Use Case analysis.  
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.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).