There are many approaches to the Software development process. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. 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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. However, readability is more than just programming style. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. One approach popular for requirements analysis is Use Case analysis. Scripting and breakpointing is also part of this process. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. Following a consistent programming style often helps readability. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Whatever the approach to development may be, the final program must satisfy some fundamental properties. There exist a lot of different approaches for each of those tasks. 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. Integrated development environments (IDEs) aim to integrate all such help. Programs were mostly entered using punched cards or paper tape. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.