Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). Whatever the approach to development may be, the final program must satisfy some fundamental properties. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Use of a static code analysis tool can help detect some possible problems. Techniques like Code refactoring can enhance readability. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Many applications use a mix of several languages in their construction and use. Scripting and breakpointing is also part of this process. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. One approach popular for requirements analysis is Use Case analysis. 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. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. 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. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Programmable devices have existed for centuries. Normally the first step in debugging is to attempt to reproduce the problem. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Different programming languages support different styles of programming (called programming paradigms). There are many approaches to the Software development process.