One approach popular for requirements analysis is Use Case analysis. It is usually easier to code in "high-level" languages than in "low-level" ones. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. Normally the first step in debugging is to attempt to reproduce the problem. Integrated development environments (IDEs) aim to integrate all such help. Allen Downey, in his book How To Think Like A Computer Scientist, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Use of a static code analysis tool can help detect some possible problems. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. There are many approaches to the Software development process. Computer programmers are those who write computer software. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.