One approach popular for requirements analysis is Use Case analysis. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Programmable devices have existed for centuries. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Unreadable code often leads to bugs, inefficiencies, and duplicated code. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Integrated development environments (IDEs) aim to integrate all such help. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. 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. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Programs were mostly entered using punched cards or paper tape. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language.