Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Use of a static code analysis tool can help detect some possible problems. 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. However, readability is more than just programming style. Use of a static code analysis tool can help detect some possible problems. 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. 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. Programmable devices have existed for centuries. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. Computer programmers are those who write computer software. 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. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Integrated development environments (IDEs) aim to integrate all such help. 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. 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. 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).