The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Also, specific user environment and usage history can make it difficult to reproduce the problem. Normally the first step in debugging is to attempt 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. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). It is usually easier to code in "high-level" languages than in "low-level" ones. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Different programming languages support different styles of programming (called programming paradigms). 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Normally the first step in debugging is to attempt to reproduce the problem. 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. However, readability is more than just programming style. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. There exist a lot of different approaches for each of those tasks. One approach popular for requirements analysis is Use Case analysis. Techniques like Code refactoring can enhance readability. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.