МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Кузбасский государственный технический университет имени Т. Ф. Горбачева»

Составители
П. А. Стрельников
М. М. Горбачева

ИНОСТРАННЫЙ ЯЗЫК ДЛЯ ПРИКЛАДНОЙ ИНФОРМАТИКИ

Методические материалы

Рекомендовано учебно-методической комиссией
направления подготовки 09.04.03 Прикладная информатика
в качестве электронного издания
для использования в образовательном процессе

Кемерово 2019

Рецензенты

Зникина Л. С. – доктор педагогических наук, заведующий кафедрой иностранных языков, профессор кафедры иностранных языков.

Пимонов А. Г. – доктор технических наук, председатель учебно-методической комиссии направления подготовки 09.04.03 Прикладная информатика.

Стрельников Павел Алексеевич
Горбачева Марина Михайловна
Иностранный язык для прикладной информатики: методические материалы [Электронный ресурс] для обучающихся направления подготовки 09.04.03 Прикладная информатика очной формы обучения / сост. П. А. Стрельников, М. М. Горбачева; КузГТУ. – Кемерово, 2019.

Целью методических материалов является помощь в практическом овладении навыками перевода специализированной иностранной литературы по направлению подготовки для активного использования в профессиональной деятельности.

© КузГТУ, 2019
© Стрельников П. А.,
Горбачева М. М.,
составление, 2019

Предисловие

Целью методических материалов является помощь в практическом овладении навыками перевода специализированной иностранной литературы по направлению подготовки для активного использования в профессиональной деятельности.

Данные методические материалы включают в себя практические задания.

Лексический состав материала методических указаний соответствует современному состоянию английского языка и включает в себя общепрофессиональную научно-техническую терминологию.

ОСОБЕННОСТИ АНГЛИЙСКОГО НАУЧНО-ТЕХНИЧЕСКОГО ТЕКСТА. ОСОБЕННОСТИ ПЕРЕВОДА НАУЧНО-ТЕХНИЧЕСКОГО ТЕКСТА

Задания и упражнения по темам: 1. Характер английского научно-технического текста. Научно-техническая терминология и лексические особенности научно-технического текста. 2. Грамматические особенности научно-технического текста. 3. Особенности русского научного текста. 4. Особенности английского текста, чуждые русскому языку. Стилистическая адаптация.

1. Проанализируйте терминологический состав текста «Computer Science»:

а) найдите простые и сложные термины;

б) определите, к какому типу (словосочетания, аббревиатура, слоговые сокращения, литерные термины) относятся найденные сложные термины;

в) назовите способы перевода (транслитерации, поиск эквивалента, калькирование, описательный перевод) найденных терминов.

2. Проанализируйте грамматическую структуру текста «Computer Science»:

а) выпишите из текста и переведите

- атрибутивные группы в роли определений,
- глаголы-операторы,
- переходные глаголы в непереходной форме с пассивным значением,
- эллиптические конструкции,
- причинно-следственные союзы и логические связки;

б) найдите в тексте и переведите:

- предложно-именные сочетания, которые можно заменить наречиями;
- отглагольные прилагательные с предлогами, которые можно заменить глаголами.

3. Переведите полностью текст «Computer Science».

COMPUTER SCIENCE

General definition

Computer science is the scientific and practical approach to computation and its applications. It is the systematic study of the feasibility, structure, expression, and mechanization of the methodical processes (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, and access to information, whether such information is encoded in bits and bytes in a computer memory or transcribed engines and protein structures in a human cell. A computer scientist specializes in the theory of computation and the design of computational systems.

Its subfields can be divided into a variety of theoretical and practical disciplines. Some fields, such as computational complexity theory (which explores the fundamental properties of computational problems), are highly abstract, while fields such as computer graphics emphasize real-world visual applications. Still other fields focus on the challenges in implementing computation. For example, programming language theory considers various approaches to the description of computation, whilst the study of computer programming itself investigates various aspects of the use of programming language and complex systems. Human-computer interaction considers the challenges in making computers and computations useful, usable, and universally accessible to humans.

History of computer science

The earliest foundations of what would become computer science predate the invention of the modern digital computer. Machines for calculating fixed numerical tasks such as the abacus have existed since antiquity but they only supported the human mind, aiding in computations as complex as multiplication and division.

Blaise Pascal designed and constructed the first working mechanical calculator, Pascal's calculator, in 1642. Two hundred years later, Thomas de Colmar launched the mechanical calculator industry when he released his simplified arithmometer, which was the first calculating machine strong enough and reliable enough to be used daily in an office environment.

Charles Babbage started the design of the first automatic mechanical calculator, his difference engine, in 1822, which eventually gave him the idea of the first programmable mechanical calculator, his «Analytical Engine». He started developing this machine in 1834 and «in less than two years he had sketched out many of the salient features of the modern computer. A crucial step was the adoption of a punched card system derived from the Jacquard loom» making it infinitely programmable.

In 1843, during the translation of a French article on the analytical engine, Ada Lovelace wrote, in one of the many notes she included, an algorithm to compute the Bernoulli numbers, which is considered to be the first computer program. Around 1885, Herman Hollerith invented the tabulator which used punched cards to process statistical information; eventually his company became part of IBM. In 1937, one hundred years after Babbage's impossible dream, Howard Aiken convinced IBM, which was making all kinds of punched card equipment and was also in the calculator business to develop his giant programmable calculator, the ASCC/Harvard Mark I, based on Babbage's analytical engine, which itself used cards and a central computing unit. When the machine was finished, some hailed it as «Babbage's dream come true».

During the 1940s, as new and more powerful computing machines were developed, the term «computer» came to refer to the machines rather than their human predecessors. As it became clear that computers could be used for more than just mathematical calculations, the field of computer science broadened to study computation in general. Computer science began to be established as a distinct academic discipline in the 1950s and early 1960s. The world's first computer science degree program, the Cambridge Diploma in Computer Science, began at the University of Cambridge Computer Laboratory in 1953. The first computer science degree program in the United States was formed at Purdue University in 1962. Since practical computers became available, many applications of computing have become distinct areas of study in their own right.

Although many initially believed it was impossible that computers themselves could actually be a scientific field of study, in the late fifties it gradually became accepted among the greater academic population. It is the now well-known IBM brand that formed part of the computer science revolution during this time. IBM (short for International Business Machines) released the IBM 704 and later the IBM 709 computers, which were widely used during the exploration period of such devices. «Still, working with the IBM was frustrating ... if you had misplaced as much as one letter in one instruction, the program would crash, and you would have to start the whole process over again». During the late 1950s, the computer science discipline was very much in its developmental stages, and such issues were commonplace.

Time has seen significant improvements in the usability and effectiveness of computing technology. Modern society has seen a significant shift in the users of computer technology, from usage only by experts and professionals, to a near-ubiquitous user base. Initially, computers were quite costly, and some degree of human aid was needed for efficient use – in part from professional computer operators. As computer adoption became more widespread and affordable, less human assistance was needed for common usage.

4. Найдите в тексте «Computer programming»:

а) личные формы глагола, которые переводятся на русский язык безличными или неопределенно-личными оборотами;

б) глаголы в будущем времени, употребляемые для выражения обычного действия;

в) пассивные обороты, которые при переводе на русский язык будут заменяться конструкциями с глаголами в активном залоге.

5. Найдите в тексте «Computer programming»:

а) сокращения, неупотребительные в русском языке, и расшифруйте их;

б) слова и выражения, чуждые русскому языку; подберите аналоги на русском языке для их замены;

в) глаголы, которые при переводе будут заменяться существительными.

6. Переведите текст «Computer programming».

COMPUTER PROGRAMMING

Computer programming is the comprehensive process that leads from an original formulation of a computing problem to executable programs. It involves activities such as analysis, understanding, and generically solving such problems resulting in an algorithm, verification of requirements of the algorithm including its correctness and its resource consumption, implementation (or coding) of the algorithm in a target programming language, testing, debugging, and maintaining the source code, implementation of the build system and management of derived artefacts such as machine code of computer programs. The algorithm is often only represented in human-parseable form and reasoned about using logic. Source code is written in one or more programming languages (such as C++, C#, Java, Python, Smalltalk, JavaScript,etc.). The purpose of programming is to find a sequence of instructions that will automate performing a specific task or solve a given problem. The process of programming thus often requires expertise in many different subjects, including knowledge of the application domain, specialized algorithms and formal logic.

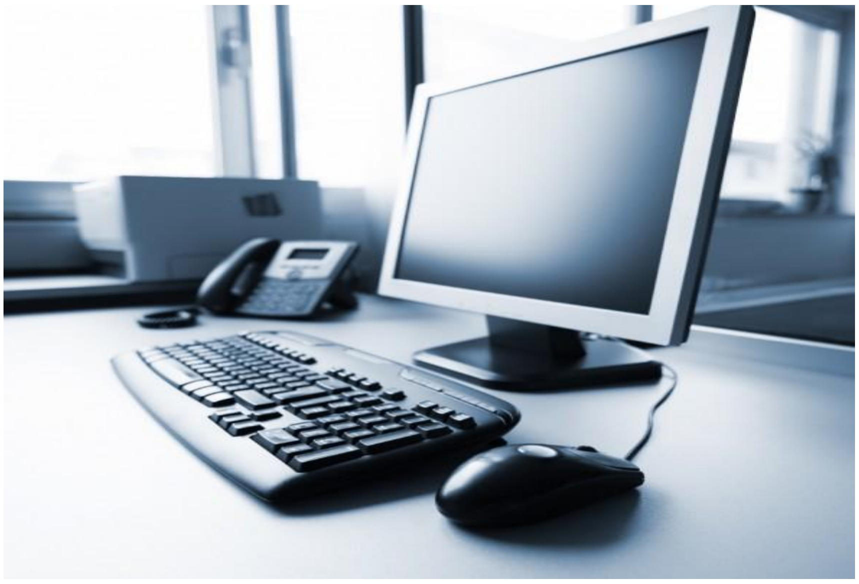
Within software engineering, programming is regarded as one phase in a software development process.

FORTRAN – the first high level programming language – was invented in 1954 and it was also first to have a functional implementation, as opposed to just a design on paper. (A high-level language is, in very general terms, any programming language that allows the programmer to write programs in terms that are more abstract than assembly language instructions, i.e. at a level of abstraction «higher» than that of an assembly language.) It allowed programmers to specify calculations by entering a formula directly (e.g. Y = X*2 + 5*X + 9). The program text, or «source», is converted into machine instructions using a special program called a compiler, which translates the FORTRAN program into machine language. In fact, the name FORTRAN stands for «Formula Translation». Many other languages were developed, including some for commercial programming, such as COBOL. Programs were mostly still entered using punched cards or paper tape. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Text editors were developed that allowed changes and corrections to be made much more easily than with punched cards.

As time has progressed, computers have made giant leaps in the area of processing power. This has brought about newer programming languages that are more abstracted from the underlying hardware. Popular programming languages of the modern era include ActionScript, C++, C#, Haskell, HTML with PHP, Java, JavaScript, Objective-C, Perl, Python, Ruby, Smalltalk, SQL, Visual Basic, and dozens more. Although these high-level languages usually incur greater overhead, the increase in speed of modern computers has made the use of these languages much more practical than in the past. These increasingly abstracted languages typically are easier to learn and allow the programmer to develop applications much more efficiently and with less source code. However, high-level languages are still impractical for a few programs, such as those where low-level hardware control is necessary or where maximum processing speed is vital. Computer programming has become a popular career in the developed world, particularly in the United States, Europe, and Japan. Due to the high labor cost of programmers in these countries, some forms of programming have been increasingly subject to offshore outsourcing (importing software and services from other countries, usually at a lower wage), making programming career decisions in developed countries more complicated, while increasing economic opportunities for programmers in less developed areas, particularly China and India.

There is an on-going debate on the extent to which the writing of programs is an art form, a craft, or an engineering discipline. In general, good programming is considered to be the measured application of all three, with the goal of producing an efficient and evolvable software solution (the criteria for «efficient» and «evolvable» vary considerably). The discipline differs from many other technical professions in that programmers, in general, do not need to be licensed or pass any standardized (or governmentally regulated) certification tests in order to call themselves «programmers» or even «software engineers». Because the discipline covers many areas, which may or may not include critical applications, it is debatable whether licensing is required for the profession as a whole. In most cases, the discipline is self-governed by the entities which require the programming, and sometimes very strict environments are defined (e.g. United States Air Force use of AdaCore and security clearance). However, representing oneself as a «Professional Software Engineer» without a license from an accredited institution is illegal in many parts of the world.

Another on-going debate is the extent to which the programming language used in writing computer programs affects the form that the final program takes. This debate is analogous to that surrounding the Sapir-Whorf hypothesis in linguistics and cognitive science, which postulates that a particular spoken language’s nature influences the habitual thought of its speakers. Different language patterns yield different patterns of thought. This idea challenges the possibility of representing the world perfectly with language, because it acknowledges that the mechanisms of any language condition the thoughts of its speaker community.



РАЗДЕЛ 2

ВИДЫ ТЕХНИЧЕСКОГО ПЕРЕВОДА. ПЕРЕВОД ПАТЕНТОВ

Задания и упражнения по темам: 1. Полный письменный перевод. 2. Реферативный перевод. 3. Аннотационный перевод. 4. Перевод патентов.

1. Выполните полный письменный перевод текста «Databases», пользуясь поэтапной технологией перевода (см. стр. учебного пособия «Иностранный язык для прикладной информатики»).

2. Выполните реферативный перевод текста «Databases», пользуясь поэтапной технологией перевода (см. стр. учебного пособия «Иностранный язык для прикладной информатики»).

3. Выполните аннотационный перевод текста «Databases».

DATABASES

Overview

A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processes requiring this information. Example: modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database management systems (DBMSs) are specially designed applications that interact with the user, other applications, and the database itself to capture and analyze data. A general-purpose database management system (DBMS) is a software system designed to allow the definition, creation, querying, update, and administration of databases. Well-known DBMSs include MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Microsoft Access, Oracle, SAP, dBASE, FoxPro, IBM DB2, LibreOffice Base and FileMaker Pro. A database is not generally portable across different DBMS, but different DBMSs can inter-operate by using standards such as SQL and ODBC or JDBC to allow a single application to work with more than one database.

Formally, the term «database» refers to the data itself and supporting data structures. Databases are created to operate large quantities of information by inputting, storing, retrieving, and managing that information. Databases are set up so that one set of software programs provides all users with access to all the data.

A «database management system» (DBMS) is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term «database» when used casually often refers to both a DBMS and the data it manipulates.

Outside the world of professional information technology, the term «database» is sometimes used casually to refer to any collection of data (perhaps a spreadsheet, maybe even a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.

The interactions catered for by most existing DBMS fall into four main groups:

- Data definition. Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- Update. Inserting, modifying, and deleting data.
- Retrieval. Obtaining information either for end-user queries and reports or for processing by applications.
- Administration. Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

Both a database and its DBMS conform to the principles of a particular database model. «Database system» refers collectively to the database model, database management system, and database.

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used for recovery of data if any of the disks fails. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications. DBMSs may be built around a custom multitasking kernel with built-in networking support, but modern DBMSs typically rely on a standard operating system to provide these functions. Since DBMSs comprise a significant economical market, computer and storage vendors often take into account DBMS requirements in their own development plans.

Databases and DBMSs can be categorized according to the database model(s) that they support (such as relational or XML), the type(s) of computer they run on (from a server cluster to a mobile phone), the query language(s) used to access the database (such as SQL or XQuery), and their internal engineering, which affects performance, scalability, resilience, and security.

Database design

The first task of a database designer is to produce a conceptual data model that reflects the structure of the information to be held in the database. A common approach to this is to develop an entity-relationship model, often with the aid of drawing tools. Another popular approach is the Unified Modeling Language. A successful data model will accurately reflect the possible state of the external world being modeled: for example, if people can have more than one phone number, it will allow this information to be captured. Designing a good conceptual data model requires a good understanding of the application domain; it typically involves asking deep questions about the things of interest to an organization, like «can a customer also be a supplier?», or «if a product is sold with two different forms of packaging, are those the same product or different products?», or «if a plane flies from New York to Dubai via Frankfurt, is that one flight or two (or maybe even three)?». The answers to these questions establish definitions of the terminology used for entities (customers, products, flights, flight segments) and their relationships and attributes.

Producing the conceptual data model sometimes involves input from business processes, or the analysis of workflow in the organization. This can help to establish what information is needed in the database, and what can be left out. For example, it can help when deciding whether the database needs to hold historic data as well as current data.

Having produced a conceptual data model that users are happy with, the next stage is to translate this into a schema that implements the relevant data structures within the database. This process is often called logical database design, and the output is a logical data model expressed in the form of a schema. Whereas the conceptual data model is (in theory at least) independent of the choice of database technology, the logical data model will be expressed in terms of a particular database model supported by the chosen DBMS.

The most popular database model for general-purpose databases is the relational model, or more precisely, the relational model as represented by the SQL language. The process of creating a logical database design using this model uses a methodical approach known as normalization. The goal of normalization is to ensure that each elementary «fact» is only recorded in one place, so that insertions, updates, and deletions automatically maintain consistency.

The final stage of database design is to make the decisions that affect performance, scalability, recovery, security, and the like. This is often called «physical database design». A key goal during this stage is data independence, meaning that the decisions made for performance optimization purposes should be invisible to end-users and applications. Physical design is driven mainly by performance requirements, and requires a good knowledge of the expected workload and access patterns, and a deep understanding of the features offered by the chosen DBMS.

Another aspect of physical database design is security. It involves both defining access control to database objects as well as defining security levels and methods for the data itself.

Database security

Database security deals with all various aspects of protecting the database content, its owners, and its users. It ranges from protection from intentional unauthorized database uses to unintentional database accesses by unauthorized entities (e.g., a person or a computer program).

Database access control deals with controlling who (a person or a certain computer program) is allowed to access what information in the database. The information may comprise specific database objects (e.g., record types, specific records, data structures), certain computations over certain objects (e.g., query types, or specific queries), or utilizing specific access paths to the former (e.g., using specific indexes or other data structures to access information). Database access controls are set by special authorized (by the database owner) personnel that uses dedicated protected security DBMS interfaces.

This may be managed directly on an individual basis, or by the assignment of individuals and privileges to groups, or (in the most elaborate models) through the assignment of individuals and groups to roles which are then granted entitlements. Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called «subschema». For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data. If the DBMS provides a way to interactively enter and update the database, as well as interrogate it, this capability allows for managing personal databases.

Data security in general deals with protecting specific chunks of data, both physically, or the interpretation of them, or parts of them to meaningful information.

Change and access logging records that accessed which attributes, what was changed, and when it was changed. Logging services allow for a forensic database audit later by keeping a record of access occurrences and changes. Sometimes application-level code is used to record changes rather than leaving this to the database. Monitoring can be set up to attempt to detect security breaches.

4. Переведите фрагмент текста патента, пользуясь таблицей приведённой на стр. учебного пособия «Иностранный язык для прикладной информатики».

System, method, and computer program product
for comparing text portions by reference to index information

US 6014663 A
https://www.google.com/patents/US6014663

Abstract

A system and method for assisting in the preparation of a document, and for analyzing a document, such as a patent or patent application, are described herein. The system aids a user to verify that terms in a patent application are being used consistently. The system also facilitates editing of the patent application so as to achieve terminology consistency. The system operates by allowing a user to select a document containing a patent application. The user then selects the specification portion of the patent application, and also selects the claims portion of the patent application. The system indexes the specification portion and the claims portion to thereby generate a merged index table. The system analyzes the merged index table to identify terms in the claims portion that are not present in the specification portion, and then displays these terms (called claim terms). A user can then edit the patent application so as to properly describe these terms in the specification.

This application is a continuation of Ser. No. 08/590,082 filed Jan. 23, 1996 now U.S. Pat. No. 5,754,840.

Background of the invention

1. Field of the Invention. This present invention relates generally to developing, maintaining, and analyzing documents.

2. Related Art. When drafting certain types of documents, the choice of terminology can have significant ramifications. For example, in the patent field, the use of consistent terminology between the patent specification and the claims is extremely important. Inconsistent terminology could result in ambiguity, vagueness, and indefiniteness as to the subject matter being described and claimed. Such ambiguity, vagueness, and indefiniteness could negatively impact the prosecution of

the patent application, and the validity and enforcement of any patent that may issue from the patent application.

The need for consistent terminology is not limited to patent documents. Other types of documents having stringent requirements of consistent terminology include legal documents (such as contracts and wills), business documents, technical/scientific manuscripts, medical documents, computer documents, etc.

Accordingly, a need exists for a system and method for enabling a user to easily determine whether consistent terminology exists in a document, and for enabling the user to easily modify the document so as to achieve consistent terminology.

Summary of the invention

The present invention is directed to a system and method for assisting in the preparation of a document, such as a patent application. The present invention aids a user to verify that terms in a patent application are being used consistently. The present invention also facilitates editing of the patent application so as to achieve terminology consistency. It can also be used to verify terminology consistency in an already existing document such as an issued patent.

The present invention operates by allowing a user to select a document containing a patent application or an issued patent. The user then selects the specification portion of the patent application, and also selects the claims portion of the patent application. The invention indexes the specification portion and the claims portion to thereby generate a merged index table. The invention analyzes the merged index table to identify terms in the claims portion that are not present in the specification portion, and then displays these terms (called claim terms).

Further features and advantages of the invention, as well as the structure and operation of various embodiments of the invention, are described in detail below with reference to the accompanying drawings. In the drawings, like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements. The drawing in which an element first appears is indicated by the leftmost digit(s) in the corresponding reference number.

Claims

What is claimed is:

1. A method for comparing text portions, comprising the steps of: (1) indexing a first selected non-predefined text portion to generate first index information; (2) indexing a second selected non-predefined text portion to generate second index information; and (3) comparing said first and second index information generated in steps (1) and (2).

2. The method of claim 1, further comprising the step of: (4) identifying differences between said first and second text portions by reference to results of said comparison performed in step (3).

3. The method of claim 1, further comprising the step of: (4) identifying similarities between said first and second text portions by reference to results of said comparison performed in step (3).

4. The method of claim 1, wherein step (1) results in generating a first index table, and step (2) results in generating a second index table.

5. The method of claim 4, further comprising the step of: merging said first and second index tables to thereby generate a merged index table.

6. The method of claim 5, wherein step (3) comprises the step of: determining by reference to said merged index table terms that are present in said second text portion, but not present in said first text portion.

7. The method of claim 4, wherein said first text portion and said second text portion are from the same document.

8. A system for comparing text portions, comprising: first text portion indexing means for indexing a first selected non-predefined text portion to generate first index information; second text portion indexing means for indexing a second selected non-predefined text portion to generate second index information; and index comparing means for comparing said first and second index information generated by said first and second text portion indexing means.

9. The system of claim 8, further comprising: means for identifying differences between said first and second text portions by reference to results of said comparison performed by said index comparing means.

10. The system of claim 8, further comprising: means for identifying similarities between said first and second text portions by reference to results of said comparison performed by said index comparing means.

11. The system of claim 8, wherein said first text portion indexing means generates a first index table, and said second text portion indexing means generates a second index table.

12. The system of claim 11, further comprising: means for merging said first and second index tables to thereby generate a merged index table.

13. The system of claim 12, wherein said index comparing means comprises: means for determining by reference to said merged index table terms that are present in said second text portion, but not present in said first text portion.

14. The system of claim 8, wherein said first text portion and said second text portion are from the same document.

15. A computer program product having control logic stored therein, said control logic, when executed, enabling a computer to compare text portions, said control logic comprising: first text portion indexing means for enabling the computer to index a first selected non-predefined text portion to generate first index information; second text portion indexing means for enabling the computer to index a second selected non-predefined text portion to generate second index information; and index comparing means for enabling the computer to compare said first and second index information generated by said first and second text portion indexing means.

16. The computer program product of claim 15, said control logic further comprising: means for enabling the computer to identify differences between said first and second text portions by reference to results of said comparison performed by said index comparing means.

17. The computer program product of claim 15, said control logic further comprising: means for enabling the computer to identify similarities between said first and second text portions by reference to results of said comparison performed by said index comparing means.

18. The computer program product of claim 15, wherein said first text portion indexing means enables the computer to generate a first index table, and said second text portion indexing means enables the computer to generate a second index table.

19. The computer program product of claim 18, said control logic further comprising: means for enabling the computer to merge said first and second index tables to thereby generate a merged index table.

20. The computer program product of claim 19, wherein said index comparing means comprises: means for enabling the computer to determine by reference to said merged index table terms that are present in said second text portion, but not present in said first text portion.

21. The computer program product of claim 15, wherein said first text portion and said second text portion are from the same document.

Brief description of the figures

The present invention will be described with reference to the accompanying drawings, wherein:

FIG. 1 is a block diagram of a preferred computer system of the present invention;

FIG. 2 is a block diagram of a document development and maintenance system according to a preferred embodiment of the present invention;

FIGS. 3, 4, 5, 6, 11, and 29 are screen shots generated by a graphical user interface of the present invention;

FIGS. 7, 12, 13, 15, 17, 18, 19, 20, 21, 22, 23, 25, 26, 27, and 28 are flowcharts depicting the preferred operation of the present invention;

FIGS. 8, 9, 10, 14, and 24 are preferred index tables according to the present invention; and

FIG. 16 depicts a document used by the present invention.