Glossary

abstract data type
: A type whose representation is concealed and which is defined by its operations.

activity (PERT) chart
: A chart used by project managers to show the dependencies between tasks that have to be completed. The chart shows the tasks, the time expected to complete these tasks and the task dependencies. The **critical path** is the longest path (in terms of the time required to complete the tasks) through the activity chart. The critical path defines the minimum time required to compete the project.

Ada
: A programming language that was developed for the US Department of Defense as a standard language for developing military software. It is based on programming language research from the 1970s and includes constructs such as abstract data types and support for concurrency. It is still used for large, complex military and aerospace systems.

agile methods
: Methods of software development that are geared to rapid software delivery. The software is developed and delivered in increments and process documentation and bureaucracy are minimised.

algorithmic cost modelling
: An approach to software cost estimation where a formula is used to estimate the project cost. The parameters in the formula are attributes of the project and the software itself.

application family
: A set of software application programs that have a common architecture and generic functionality. These can be tailored to the needs of specific customers by modifying components and program parameters.

application framework
: A generic structure in some specific domain that can form the basis of a family of applications. Application frameworks are generally implemented as a set of concrete and abstract classes that are specialised and instantiated to create an application.

application program interface (API)
: An interface, generally specified as a set of operations, which is defined by an application program that allows access to the program's functionality. This means that this functionality can be called on directly by other programs and not just accessed through the user interface.

aspect-oriented programming
: An approach to programming that combined generative and component-based development. Cross-cutting concerns are identified in a program and

the implementation of these concerns is defined as aspects. A program weaver then weaves the aspects into the appropriate places in the program.

availability
The readiness of a system to deliver services when requested. Availability is usually expressed as a decimal number so an availability of 0.999 means that the system can deliver services for 999 out of 1000 time units. <change this?>

bar (Gantt) chart
A chart used by project managers to show the project tasks, the schedule associated with these tasks and the people who will work on them. It shows the tasks start and end dates and the people allocations against a timeline.

C
A programming language that was originally developed to help implement the Unix system. C is a relatively low-level system implementation language that allows access to the system hardware and which can be compiled to efficient code. It is still widely used for low-level systems programming.

C++
An object-oriented programming language that is a superset of C.

CASE
Computer-aided software engineering. The process of developing software using automated support.

CASE tool
A software tool, such as a design editor or a program debugger, used to support an activity in the software development process.

CASE workbench
An integrated set of CASE tools that work together to support a major process activity such as software design or configuration management.

client-server architecture
An architectural model for distributed systems where the system functionality is offered as a set of services provided by a server. These are accessed by client computers that make use of the services. Variants of this approach such as three-tier client-server architectures use multiple servers.

cleanroom software engineering
An approach to software development where the aim is to avoid introducing faults into the software (by analogy with a cleanroom used in semiconductor fabrication). The process involves formal software specification, structured transformation of a specification to a program, the development of correctness arguments and statistical program testing.

CMMI
An integrated approach to process capability maturity modelling. It supports discrete and continuous maturity modelling and integrates systems and software engineering process maturity models.

COCOMO
Constructive Cost Modeling. Perhaps the best known algorithmic cost estimation model.

code of ethics and professional practice
A set of guidelines that set out expected ethical and professional behaviour for software engineers. This was defined by the major US professional societies (the ACM and the IEEE) and defines ethical behaviour under 8 headings – public, client and employer, product, judgement, management, colleagues, profession, self.

COM+
A component model designed for use on Microsoft platforms

component
A deployable, independent unit of software that is completely defined and accessed through a set of interfaces.

component model
A set of standards for component implementation, documentation and deployment. These cover the specific interfaces that may be provided by a component, component naming, component inter-operation and component composition. Component models provide the basis for middleware to support executing components.

component-based software engineering (CBSE)
The development of software by composing independent, deployable components.

configuration item
A machine readable unit, such as a document or a source code file, that is subject to change and where the change has to be controlled by a configuration management system.

configuration management
The process of managing the changes to an evolving software product. Configuration management involves configuration planning, version management, system building and change management.

CORBA
Common Request Broker Architecture. A set of standards proposed by the OMG that define a distributed object model and object communications.

CORBA component model
A component model designed for use for the CORBA platform.

critical system
A computer system whose failure can result in significant economic, human or environmental losses.

data processing system
A system whose aim is to process large amounts of structured data. These systems usually process the data in batches and follow an input-process-output model. Examples of data processing systems are billing and invoicing systems, payment systems and so on.

dependability
The dependability of a system is an aggregate property that takes into account the system's safety, reliability, availability, security and other attributes. The dependability of a system reflects the extent to which it can be trusted by its users.

dependability requirement
A system requirement that is included to help achieve the required dependability for a system. Non-functional dependability requirements specify dependability attribute values; functional dependability requirements are functional requirements to avoid, detect, tolerate or recover from system faults and failures.

dependability case
A structured document that is used to back up claims made by a system developer about the dependability of a system.

design pattern
A well-tried solution to a common problem that captures experience and good practice in a form that can be reused. It is an abstract representation than can be instantiated in a number of different ways.

distributed system
A software system where the software sub-systems or components execute on different processors.

distributed object system
A distributed system where the executing components are objects.

domain
A specific problem or business area where software systems are used. Examples of domains are real-time control, business data processing, telecommunications switching, etc.

domain model
A definition of domain abstractions such as policies, procedures, objects, relationships, events, etc. It serves as a base of knowledge about some problem area.

emergent property
A property that only becomes apparent once all of the components of the system have been integrated to create the system.

enterprise Java beans (EJB)
A Java-based component model.

ethnography
An observational technique that may be used in requirements elicitation

and analysis. The ethnographer immerses himself or herself in the user's environment and observes their day to day working. Requirements for software support can be inferred from these observations.

event-based systems
Systems where the control of operation is determined by events that are generated in the system's environment. Most real-time systems are event-based systems.

extreme programming
An agile method of software development that includes practices such as scenario-based requirements, test-first development and pair programming.

fault avoidance
Developing software in such a way that faults are not introduced into that software.

fault detection
The use of processes and run-time checking to detect and remove faults in a program before these result in a system failure.

fault tolerance
The ability of a system to continue in execution even after faults have occurred.

formal methods
Methods of software development that are based on mathematically rigorous approaches and which model the software using formal mathematical constructs such as predicates and sets.

formal specification, algebraic
A method of mathematical system specification where a system or component is specified by defining relationships between the operations defined in its external interfaces.

formal specification, model-based
A method of mathematical system specification where a system or component is specified by defining pre-conditions, post-conditions and invariants that apply to the system state.

information hiding
Using programming language constructs to conceal the representation of data structures and to control external access to these structures.

incremental development
An approach to software development where the software is delivered and deployed in increments.

interface
A specification of attributes and operations associated with a software component. The interface is used as the means of accessing the component's functionality.

ISO 9000
: A standard for quality management processes that is defined by the International Standards Organisation (ISO).

iterative development
: An approach to software development where the processes of specification, design, programming and testing are inter-leaved.

Java
: An object-oriented programming language that was designed by Sun with the aim of platform independence.

language processing system
: A system that translates one language to another. For example, a compiler is a language processing system that translates program source code to object code.

legacy system
: A socio-technical system that is useful or essential to an organisation but which has been developed using obsolete technology or methods. As legacy systems often perform critical business functions, they have to be maintained.

maintenance
: The process of making changes to a system after it has been put into operation.

middleware
: The infrastructure software in a distributed system. It helps manage interactions between the distributed entities in the system and the system databases. Examples of middleware are an object request broker and a transaction management system.

object class
: An object class defines the attributes and operations of objects. Objects are created at run-time by instantiating the class definition. The object class name can be used as a type name in some object-oriented languages.

object model
: A model of a software system that is structured and organised as a set of object classes and the relationships between these classes. Various different perspectives on the model may exist such as a state perspective, a sequence perspective, etc.

object-oriented development
: An approach to software development where the fundamental abstractions in the system are independent objects. The same type of abstraction is used during specification, design and development.

OCL
: Object Constraint Language. A language that is part of the UML that is

used to define predicates that apply to object classes and interactions in a UML model.

OMG
The Object Management Group. A group of companies formed to develop standards for object-oriented development. Examples of standards promoted by the OMG are CORBA, UML and MDA.

peer-to-peer system
A distributed system where there is no distinction between clients and servers. Computers in the system can act as both clients and servers. Peer-to-peer applications include file sharing, instant messaging and cooperation support systems.

people capability maturity model
A process maturity model that reflects how effective an organisation is at managing the skills, training and experience of the people in that organisation.

process improvement
The process of making changes to a process with the aim of making that process more predictable or to improve the quality of its outputs. For example, if your aim is to reduce the number of defects in the delivered software, you might improve the process by adding new validation activities.

process model
An abstract representation of a process. Process models may be developed from different perspectives and may show the activities involved in a process, the artefacts used in the process, constraints that apply to the process and the roles of the people enacting the process.

process maturity model
A model of the extent to which a process includes good practice and reflective and measurement capabilities that are geared to process improvement.

program evolution dynamics
The study of the ways in which an evolving software system changes.

program generator
A program that generates another program from a high-level, abstract specification. The generator embeds knowledge that is reused in each generation activity.

program inspection
A verification process where a group of inspectors examine a program, line by line, with the aim of detecting program errors.

quality assurance
The overall process of defining how software quality can be achieved and

how the development organisation knows that the software has the required level of quality.

quality control
The process of ensuring that a software development team is following quality standards.

quality plan
A plan that defines the quality processes and procedures that should be used. This involves selecting and instantiating standards for products and processes and defining the required quality attributes of the system.

rapid application development (RAD)
An approach to software development aimed at rapid delivery of the software. It often involves the use of database programming and development support tools such as screen and report generators.

Rational Unified Process (RUP)
A generic software process model that presents software development as a four-phase iterative activity where the phases are inception, elaboration, construction and transition. Inception establishes a business case for the system, elaboration defines the architecture, construction implements the system and transition deploys the system in the customer's environment.

real-time system
A system that has to respond to and process external events in 'real-time'. The correctness of the system does not just depend on what it does but also how quickly it does it. Real-time systems are usually organised as a set of cooperating sequential processes.

reengineering
Modifying a software system to make it easier to understand and change. Reengineering often involves software and data restructuring and organisation, program simplification and re-documentation.

reengineering, business process
Changing a business process to meet some new organisational objectives such as reduced cost, faster execution, etc.

reference architecture
A generic system architecture that is an idealised architecture that includes all the features that systems might incorporate. They are a way of informing designers about the general structure of that class of system

release
A version of a software system that is made available to system customers.

reliability
The ability of a system to deliver services as specified. Reliability can be specified quantitatively as a probability of failure on demand or as the rate of occurrence of failure.

reliability growth modelling
The development of a model of how the reliability of a system changes (hopefully improves) as it is tested and program defects are removed.

requirement, functional
A statement of some function or feature that should be implemented in a system.

requirement, non-functional
A statement of a constraint or expected behaviour that applies to a system. This constraint may refer to the emergent properties of the software that is being developed or to the development process.

requirements management
The process of managing changes to requirements to ensure that the changes made are properly analysed and tracked through the system.

risk
An undesirable outcome that poses a threat to the achievement of some objective. A process risk threatens the schedule or cost of a process; a product risk is a risk that may mean that some of the system requirements may not be achieved.

risk management
The process of identifying risks, assessing their severity, planning measures to put in place if the risks arise and monitoring the software and the software process for risks.

safety
The ability of a system to operate without catastrophic failure.

safety case
A structured argument that a system is safe. Usually required by regulators such as nuclear safety regulators.

scenario
A description of one typical way in which a system is used or a user carried out some activity.

security
The ability of a system to protect itself against accidental or deliberate intrusion.

sequence diagram
A diagram that shows the sequence of interactions required to complete some operation. In the UML, sequence diagrams may be associated with use-cases.

server
A program that provides some service to other (client) programs.

software architecture
A model of the fundamental structure and organisation of a software system.

software metric
An attribute of a software system or process that can be expressed numerically and measured. Process metrics are attributes of the process such as the time taken to complete a task; product metrics are attributes of the software itself such as size or complexity.

software product line
See application family.

socio-technical system
A system that includes hardware and software components, that has defined operational processes that are followed by human operators and that operates within an organisation. It is therefore influenced by organisational policies, procedures and structures.

software process
The related set of activities and processes that are involved in developing and evolving a software system.

software life cycle
Often used as another name for the software process. Originally coined to refer to the waterfall model of the software process.

spiral model
A model of a development process where the process is represented as a spiral with each round of the spiral incorporating the different stages in the process. As you move from one round of the spiral to another, you repeat all of the stages of the process.

SQL
Structured Query Language. A standard language used for relational database programming.

static analysis
Tool-based analysis of a program's source code to discover errors and anomalies. Anomalies such as successive assignments to a variable with no intermediate use may be programming errors.

structured method
A method of software design that defines the system models that should be developed, the rules and guidelines that should apply to these models and a process to be followed in developing the design.

system building
The process of compiling the components or units that make up a system and linking these with other components to create an executable program. System building is normally automated so that re-compilation is minimised. This automation may be built-in to the language processing system (as in Java) or may involve CASE tools to support system building.

systems engineering
A process that is concerned with specifying a system, integrating its components and testing that the system meets its requirements. System engineering is concerned with the whole socio-technical system –

software, hardware and operational processes, not just the system software.

transaction
An unit of interaction with a computer system. Transactions are independent and atomic (they are not broken down into smaller units) and are a fundamental unit of recovery, consistency and concurrency.

transaction processing system
A system that ensures that transactions are processed in such a way so that they do not interfere with each other and so that individual transaction failure does not affect other transactions or the system's data.

UML
Unified Modeling Language. A graphical language that is used in object-oriented development that includes a several types of system model that provide different views of a system. The UML has become a de facto standard for object-oriented modelling.

use-case
A specification of one type of interaction with a system.

user interface design
The process of designing the way in which system users access the system functionality and information produced by the system is displayed.

user interface design principles
A set of principles that embody good practice for user interface design.

validation
The process of checking that a system meets the needs and expectations of the customer.

verification
The process of checking that a system meets its specification.

waterfall model
A software process model where there are discrete development stages – specification, design, implementation, testing and maintenance. In principle, one stage must be complete before progress to the next stage is possible. In practice, this is not the case and there is iteration between stages.

web service
An independent software component that can be accessed through the Internet using standard protocols. SOAP (Standard Object Access Protocol) is used for web service information exchange. WSDL (Web Service Definition Language) is used to define the web service interfaces.

Wizard-of-Oz prototyping
An approach to user interface prototyping where commands input by a

user are interpreted by a person who responds as if they were the computer.

XML

Extended markup language. XML is a text markup language that supports the interchange of structured data. Each data field is delimited by tags that give information about that field. XML is now very widely used and has become the basis of protocols for web services.

Z

A model-based, formal specification language developed at the University of Oxford in England.

Definitions of many other terms are available in the on-line glossary accessible from the book's web site.