IEEE
Std 610.12-1990
(Revision and redesignation of IEEE Std 792-1983)

IEEE Standard Glossary of Software Engineering Terminology

Sponsor

Standards Coordinating Committee of the Computer Society of the IEEE

Approved September 28, 1990

IEEE Standards Board

Abstract: IEEE Std 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, identifies terms currently in use in the field of Software Engineering. Standard definitions for those terms are established.
Keywords: Software engineering; glossary; terminology; definitions; dictionary

ISBN 1-55937-067-X

The Institute of Electrical and Electronics Engineers
345 East 47th Street, New York, NY 10017, USA

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
345 East 47th Street
New York, NY 10017
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Foreword

(This Foreword is not a part of IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology.)

The computer field is continuing to expand. New terms are being generated and new meanings are being adopted for existing terms. The IEEE Computer Dictionary project was undertaken to document this vocabulary. Its purpose is to identify terms currently in use in the computer field and to establish standard definitions for these terms. The dictionary is intended to serve as a useful reference for those in the computer field and for those who come into contact with computers either through their work or in their everyday lives.

The dictionary is being developed as a set of subject-area glossaries covering Computer Architecture, Computer Processors, Computer Storage, Software Engineering, Mathematics of Computing, Theory of Computation, Computer Applications, Artificial Intelligence, Data Management, Image Processing and Pattern Recognition, Modeling and Simulation, Computer Graphics, Computer Networking, Computer Languages, and Computer Security and Privacy. This glossary contains the terms related to Software Engineering. It updates IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology (ANSI).

Every effort has been made to use definitions from established standards in this dictionary. When existing standards were found to be incomplete, unclear, or inconsistent with other entries in the dictionary, however, new, revised, or composite definitions have been developed.

At the time this glossary was approved, the following people formed the steering committee of the Computer Dictionary working group:

Jane Radatz, Chairperson, *Software Engineering Glossary*

Other subgroup leaders:

Anne Geraci
Freny Katki
Dr. John Lane
Louise McMonegal
Bennett Meyer
Dr. Hugh Porteous
Dr. Fredrick Springsteel
Paul Wilson
Mary Yee
John Young

Other working group members who contributed to this glossary were as follows:

Russell J. Abbott
A. Frank Ackerman
Roger R. Baldwin
H. Ronald Berlack
J. David Bezek
James H. Bradley
Kathleen L.Briggs
Homer C. Carney
Susann Chonoles
Taz Daughtrey
Frank J. Douglas
William P. Dupras
John D. Earls
Mary Forcht-Tucker
David Gelperin
Al Gillen
Shirley A. Gloss-Soler
John A. Goetz
David A. Gustafson
Virl Haas
James Ingram
Gary S. Lindsay
Robert McBeth
Alicia McCurdy
Dr. José Muñoz
Geraldine Neidhart
Mary Rasmussen
Max Schindler
Paul Schmid
Leonard W. Seagren
Sonja Peterson Shields
Kevin Smith
Wayne Smith
Paul U. Thompson
Andrew H. Weigel
W. Martin Wong

Special representatives to the Computer Dictionary working group were as follows:

Frank Jay, *Advisor, IEEE Standards Office*

Rollin Mayer, *Liaison, Accredited Standards Committee X3K5*

The following organizations supported employee participation in the development of this standard:

Applied Information Development
Atlantic Consultants
Babcock and Wilcox
Burroughs Wellcome
Carney Associates
Computer Sciences Corporation
Datapoint Corporation
Dutchess Engineering Company
Edinboro University of Pennsylvania
Electronics Design Magazine
Eyring Research Institute
General Electric Company
Harris Corporation
Information Spectrum, Inc.
Institute for Zero Defect Software
International Bureau of Software Test
Kansas State University
Lockheed
Logicon
Marine Midland Bank
The MITRE Corporation
Perkin-Elmer Corporation
Quality Assurance Institute
Rabbitt Software Corporation
RCA
Sanders Associates
SILOGIC
Softran
Teledyne Brown Engineering
University of Wisconsin, Madison
U.S. Naval Facilities
U.S. Dept. of HUD
Wyse Technology

The IEEE 610 working group wishes to acknowledge the contribution of those who developed IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology (ANSI), which formed the basis for this glossary. The steering committee of this group had the following members:

Shirley A. Gloss-Soler, *Chair*

Russell J. Abbott
Joan P. Bateman
Stephen R. Beason
Milton E. Boyd, Jr.
Kurt F. Fischer
John M. Ives
John J. McKissick, Jr.
Albrecht J. Neumann
John N. Postak
Jane W. Radatz
Marilyn J. Stewart
Alan N. Sukert
Donald A. Woodmancy
David Yablon

The sponsor for the Computer Dictionary project is the IEEE Computer Society Standards Coordinating Committee, which balloted this document for submission to the IEEE Standards Board. At the time this standard was approved, the committee had the following membership:

Harrison Beasley
H. Ronald Berlack
William Billowitch
Richard Boberg
John Boebinger
Paul L. Borrill
Terry Bowen
Elliott Brebner
J. Reese Brown, Jr.
Lin Brown
Fletcher Buckley
Randy Bush
Clyde Camp
Steve Carter
Alan Cobb
Paul Cook
Bill Corwin
Alan Davis
Steven Deller
Bulent Dervisoglu
Bob Donnan
Paul Eastman
D. Vera Edelstein
Tim Elsmore
Dick Evans
Richard Fairley
Wayne Fischer
Kester Fong
David Gelperin
Anne Geraci
Al Gilman
John Graham
Steve Grout
Dave Gustavson
Al Hankinson
Fred Harrison
Chris Haynes
Rick Henderson
Ken Hobday
Scott Hopkinson
John Horch
Russell Housley
Charles Hudson
Marlyn Huckeby
Mike Humphrey
John Hyde
James Isaak
David James
Hal Jesperson
Richard Kalish
Matt Kaltenbach
Hans Karlsson
Freni Katki
Guy Kelley
Kim Kirkpatrick
Bob Knighten
Stanley Krolikoski
John B. Lane
Ron Leckie
Kevin Lewis
William Lidinsky
Donald C. Loughry
Al Lowenstein
Bill Maciejewski
Roger Martin
Philip Marriott
Colin Maunder
John McGrory
Louise McMonegal
Sunil Mehta
Paul Menchini
Jerry Mersky
Bennett Meyer
Louis Miller
James Mollenauer
Jim Mooney
Gary A. Nelson
Tom Pittman
Robert M. Poston
Shlomo Pri-Tal
Jane Radatz
Michael Raynham
Gordon Robinson
Larry Saunders
Richard Schmidt
Norman Schneidewind
Rudolph Schubert
David Schultz
Karen Sheaffer
Basil Sherlund
Sava Sherr
Fred Springsteel
John Starkweather
Dennis Steinauer
Robert Sulgrove
Oryal Tanir
Michael D. Teener
Donn Terry
Pat Thaler
Joseph Toy
Leonard Tripp
Margaret Updike
Eike Waltz
John W. Walz
Camille White-Partain
Les Wibberley
Cynthia Wright
John Young
Jason Zions

When the IEEE Standards Board approved this standard on September 28, 1990, it had the following membership:

Marco W. Migliaro, *Chairman* **James M. Daly,** *Vice Chairman*
Andrew G. Salem, *Secretary*

Dennis Bodson
Paul L. Borrill
Fletcher J. Buckley
Allen L. Clapp
Stephen R. Dillon
Donald C. Fleckenstein
Jay Forster*
Thomas L. Hannan
Kenneth D. Hendrix
John W. Horch
Joseph L. Koepfinger*
Irving Kolodny
Michael A. Lawler
Donald J. Loughry
John E. May, Jr.
Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
Stig Nilsson
Roy T. Oishi
Gary S. Robinson
Terrance R. Whittemore
Donald W. Zipse

*Member Emeritus

Contents

IEEE Standard Glossary of Software Engineering Terminology

1. Scope

This glossary defines terms in the field of Software Engineering. Topics covered include addressing; assembling, compiling, linking, loading; computer performance evaluation; configuration management; data types; errors, faults, and failures; evaluation techniques; instruction types; language types; libraries; microprogramming; operating systems; quality attributes; software documentation; software and system testing; software architecture; software development process; software development techniques; and software tools.

Every effort has been made to include all terms that meet these criteria. Terms were excluded if they were considered to be parochial to one group or organization; company proprietary or trademarked; multi-word terms whose meaning could be inferred from the definitions of the component words; or terms whose meaning in the computer field could be directly inferred from their standard English meaning.

This glossary is an update and expansion of IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology (ANSI) [3].[1] It increases the number of terms from approximately 500 to 1300, and updates or refines the definitions of many terms included in the initial glossary. A few terms that were included in the initial glossary have been moved to other glossaries in the 610 series. Some definitions have been recast in a system, rather than software, context. Every effort has been made to preserve the fine work that went into the initial glossary.

2. Glossary Structure

Entries in the glossary are arranged alphabetically. An entry may consist of a single word, such as "software," a phrase, such as "test case," or an acronym, such as "CM." Phrases are given in their natural order (test plan) rather than in reversed order (plan, test).

Blanks precede all other characters in alphabetizing. Hyphens and slashes are treated as blanks. Alternative spellings are shown in parentheses.

If a term has more than one definition, the definitions are numbered. In most cases, noun definitions are given first, followed by verb and adjective definitions as applicable. Examples, notes, and illustrations have been added to clarify selected definitions.

The following cross-references are used to show a term's relationship to other terms in the dictionary:

- *Contrast with* refers to a term with an opposite or substantially different meaning.
- *Syn* refers to a synonymous term.
- *See also* refers to a related term.
- *See* refers to a preferred term or to a term where the desired definition can be found.

The word "deprecated" indicates a term or definition whose use is discouraged because such use is obsolete, misleading, or ambiguous. "DoD" refers to usage by the U.S. Department of Defense.

3. Definitions for Software Engineering Terms

1GL. Acronym for **first generation language.** *See:* **machine language.**

2GL. Acronym for **second generation language.** *See:* **assembly language.**

3GL. Acronym for **third generation language.** *See:* **high order language.**

4GL. Acronym for **fourth generation language.**

5GL. Acronym for **fifth generation language.**

[1] Numbers in brackets correspond to those in the Bibliography in Section 4.

abend. Abbreviation for **abnormal end.**

abnormal end (abend). Termination of a process prior to completion. *See also:* **abort; exception.**

abort. To terminate a process prior to completion. *See also:* **abend; exception.**

absolute address. An address that is permanently assigned to a device or storage location and that identifies the device or location without the need for translation or calculation. *Syn:* **explicit address; specific address.** *Contrast with:* **relative address; relocatable address; symbolic address.** *See also:* **absolute assembler; absolute code; absolute instruction; absolute loader.**

absolute assembler. An assembler that produces absolute code. *Contrast with:* **relocating assembler.**

absolute code. Code in which all addresses are absolute addresses. *Contrast with:* **relocatable code.** *Syn:* **specific code.**

absolute instruction. A computer instruction in which all addresses are absolute addresses. *See also:* **direct instruction; effective instruction; immediate instruction; indirect instruction.**

absolute loader. A loader that reads absolute machine code into main memory, beginning at the initial address assigned to the code by the assembler or compiler, and performs no address adjustments on the code. *Contrast with:* **relocating loader.**

abstract data type. A data type for which only the properties of the data and the operations to be performed on the data are specified, without concern for how the data will be represented or how the operations will be implemented.

abstraction. (1) A view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information. *See also:* **data abstraction.**
(2) The process of formulating a view as in (1).

acceptance criteria. The criteria that a system or component must satisfy in order to be accepted by a user, customer, or other authorized entity. *See also:* **requirement; test criteria.**

acceptance testing. (1) (IEEE Std 1012-1986 [12]) Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.
(2) Formal testing conducted to enable a user, customer, or other authorized entity to determine whether to accept a system or component.
Contrast with: **development testing.** *See also:* **operational testing; qualification testing.**

accuracy. (1) A qualitative assessment of correctness, or freedom from error.
(2) A quantitative measure of the magnitude of error.
Contrast with: **precision.**

active redundancy. In fault tolerance, the use of redundant elements operating simultaneously to prevent, or permit recovery from, failures. *Contrast with:* **standby redundancy.**

actual instruction.* *See:* **effective instruction.**
* Deprecated.

actual parameter. *See:* **argument (3).**

Ada. *Note:* Ada and other specific computer languages are defined in P610.13 [17].

adaptability. *See:* **flexibility.**

adaptation data. Data used to adapt a program to a given installation site or to given conditions in its operational environment.

adaptation parameter. A variable that is given a specific value to adapt a program to a given installation site or to given conditions in its operational environment; for example, the variable Installation_Site_Latitude.

adaptive maintenance. Software maintenance performed to make a computer program usable in a changed environment.

Contrast with: **corrective maintenance; perfective maintenance.**

address. (1) A number, character, or group of characters that identifies a given device or storage location.
(2) To refer to a device or storage location by an identifying number, character, or group of characters.
See also: **absolute address; effective address; implied addressing; indirect address; relative address; relocatable address; symbolic address; virtual address.**

address field. Any of the fields of a computer instruction that contain addresses, information necessary to derive addresses, or values of operands. *Syn:* **address part.** *Contrast with:* **operation field.**

address format. (1) The number and arrangement of address fields in a computer instruction. *See also:* ***n*-address instruction; *n*-plus-one address instruction.**
(2) The number and arrangement of elements within an address, such as the elements needed to identify a particular channel, device, disk sector, and record in magnetic disk storage.

address modification. Any arithmetic, logical, or syntactic operation performed on an address. *See also:* **effective address; indexed address; relative address; relocatable address.**

address part. *See:* **address field.**

address space. (1) The addresses that a computer program can access. *Note:* In some systems, this may be the set of physical storage locations that a program can access, disjoint from other programs, together with the set of virtual addresses referring to those storage locations, which may be accessible by other programs.
(2) The number of memory locations that a central processing unit can address.

addressing exception. An exception that occurs when a program calculates an address outside the bounds of the storage available to it. *See also:* **data exception; operation exception; overflow exception; protection exception; underflow exception.**

afferent. Pertaining to a flow of data or control from a subordinate module to a superordinate module in a software system. *Contrast with:* **efferent.**

algebraic language. A programming language that permits the construction of statements resembling algebraic expressions, such as $Y = X + 5$. For example, FORTRAN. *See also:* **algorithmic language; list processing language; logic programming language.**

algorithm. (1) A finite set of well-defined rules for the solution of a problem in a finite number of steps; for example, a complete specification of a sequence of arithmetic operations for evaluating sine x to a given precision.
(2) Any sequence of operations for performing a specific task.

algorithmic language. A programming language designed for expressing algorithms; for example, ALGOL. *See also:* **algebraic language; list processing language; logic programming language.**

allocated baseline. In configuration management, the initial approved specifications governing the development of configuration items that are part of a higher level configuration item. *Contrast with:* **developmental configuration; functional baseline; product baseline.** *See also:* **allocated configuration identification.**

allocated configuration identification. In configuration management, the current approved specifications governing the development of configuration items that are part of a higher level configuration item. Each specification defines the functional characteristics that are allocated from those of the higher level configuration item, establishes the tests required to demonstrate achievement of its allocated functional characteristics, delineates necessary interface requirements with other associated configuration items, and establishes design constraints, if any. *Contrast with:* **func-**

tional configuration identification; product configuration identification. *See also:* **allocated baseline.**

allocation. (1) The process of distributing requirements, resources, or other entities among the components of a system or program.
(2) The result of the distribution in (1).

anomaly. (IEEE Std 1012-1986 [12]) Anything observed in the documentation or operation of software that deviates from expectations based on previously verified software products or reference documents.

anticipatory buffering. A buffering technique in which data are stored in a buffer in anticipation of a need for the data. *See also:* **dynamic buffering; simple buffering.**

anticipatory paging. A storage allocation technique in which pages are transferred from auxiliary storage to main storage in anticipation of a need for those pages. *Contrast with:* **demand paging.**

application generator. A code generator that produces programs to solve one or more problems in a particular application area; for example, a payroll generator.

application-oriented language. A computer language with facilities or notations applicable primarily to a single application area; for example, a language for computer-assisted instruction or hardware design. *See also:* **authoring language; specification language; query language; simulation language.**

application software. Software designed to fulfill specific needs of a user; for example, software for navigation, payroll, or process control. *Contrast with:* **support software; system software.**

architectural design. (1) The process of defining a collection of hardware and software components and their interfaces to establish the framework for the development of a computer system. *See also:* **functional design.**
(2) The result of the process in (1).

architecture. The organizational structure of a system or component. *See also:* **component; module; subprogram; routine.**

argument. (1) An independent variable; for example, the variable m in the equation $E = mc^2$.
(2) A specific value of an independent variable; for example, the value $m = 24$ kg.
(3) A constant, variable, or expression used in a call to a software module to specify data or program elements to be passed to that module. *Syn:* **actual parameter.** *Contrast with:* **formal parameter.**

array. An n-dimensional ordered set of data items identified by a single name and one or more indices, so that each element of the set is individually addressable. For example, a matrix, table, or vector.

artificial intelligence. *Note:* P610.8 [15] defines terminology pertaining to artificial intelligence.

artificial language. *See:* **formal language.**

assemble. To translate a computer program expressed in an assembly language into its machine language equivalent. *Contrast with:* **compile; disassemble; interpret.**

assemble-and-go. An operating technique in which there are no stops between the assembling, linking, loading, and execution of a computer program.

assembled origin. The address of the initial storage location assigned to a computer program by an assembler, a compiler, or a linkage editor. *Contrast with:* **loaded origin.** *See also:* **offset (1); starting address.**

assembler. A computer program that translates programs expressed in assembly language into their machine language equivalents. *See also:* **absolute assembler; cross-assembler; relocating assembler.** *Contrast with:* **compiler; interpreter.**

assembler code. *See:* **assembly code.**

assembler language. *See:* **assembly language.**

assembly code. Computer instructions and data definitions expressed in a form that can be recognized and processed by an assembler. *Syn:* **assembler code.** *Contrast with:* **compiler code; interpretive code; machine code.**

assembly language. A programming language that corresponds closely to the instruction set of a given computer, allows symbolic naming of operations and addresses, and usually results in a one-to-one translation of program instructions into machine instructions. *Syn:* **assembler language; low level language; second generation language.** *Contrast with:* **fifth generation language; fourth generation language; high order language; machine language.** *Note:* Specific languages are defined in P610.13 [17].

assertion. A logical expression specifying a program state that must exist or a set of conditions that program variables must satisfy at a particular point during program execution. Types include input assertion, loop assertion, output assertion. *See also:* **invariant; proof of correctness.**

assignment statement. A computer program statement that assigns a value to a variable; for example, $Y := X - 5$. *Contrast with:* **control statement; declaration.** *See also:* **clear; initialize; reset.**

atomic type. A data type, each of whose members consists of a single, nondecomposable data item. *Syn:* **primitive type.** *Contrast with:* **composite type.**

attribute. A characteristic of an item; for example, the item's color, size, or type. *See also:* **quality attribute.**

audit. An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria. *See also:* **functional configuration audit; physical configuration audit.**

authoring language. A high level programming language used to develop courseware for computer-assisted instruction. *See also:* **authoring system.**

authoring system. A programming system that incorporates an authoring language.

automated verification system. (1) A software tool that accepts as input a computer program and a representation of its specification and produces, possibly with human help, a proof or disproof of the correctness of the program. (2) Any software tool that automates part or all of the verification process.

availability. The degree to which a system or component is operational and accessible when required for use. Often expressed as a probability. *See also:* **error tolerance; fault tolerance; robustness.**

back-to-back testing. Testing in which two or more variants of a program are executed with the same inputs, the outputs are compared, and errors are analyzed in case of discrepancies. *See also:* **mutation testing.**

background. In job scheduling, the computing environment in which low-priority processes or those not requiring user interaction are executed. *Contrast with:* **foreground.** *See also:* **background processing.**

background processing. The execution of a low-priority process while higher priority processes are not using computer resources, or the execution of processes that do not require user interaction. *Contrast with:* **foreground processing.**

backup. (1) A system, component, file, procedure, or person available to replace or help restore a primary item in the event of a failure or externally caused disaster.
(2) To create or designate a system, component, file, procedure, or person as in (1).

backup programmer. The assistant leader of a chief programmer team; responsibilities include contributing significant portions of the software being developed by the team, aiding the chief programmer in reviewing the work of other team members, substituting for the chief programmer when

necessary, and having an overall technical understanding of the software being developed. *See also:* **chief programmer.**

backward execution. *See:* **reversible execution.**

backward recovery. (1) The reconstruction of a file to a given state by reversing all changes made to the file since it was in that state.
(2) A type of recovery in which a system, program, database, or other system resource is restored to a previous state in which it can perform required functions.
Contrast with: **forward recovery.**

base address. An address used as a reference point to which a relative address is added to determine the address of the storage location to be accessed. *See also:* **indexed address; relative address; self-relative address.**

baseline. (1) A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
(2) A document or a set of such documents formally designated and fixed at a specific time during the life cycle of a configuration item. *Note:* Baselines, plus approved changes from those baselines, constitute the current configuration identification. *See also:* **allocated baseline; developmental configuration; functional baseline; product baseline.**
(3) Any agreement or result designated and fixed at a given time, from which changes require justification and approval.

baseline management. In configuration management, the application of technical and administrative direction to designate the documents and changes to those documents that formally identify and establish baselines at specific times during the life cycle of a configuration item.

batch. Pertaining to a system or mode of operation in which inputs are collected and processed all at one time, rather than being processed as they arrive, and a job, once started, proceeds to completion without additional input or user interaction. *Contrast with:* **conversational; interactive; online; real time.**

bathtub curve. A graph of the number of failures in a system or component as a function of time. The name is derived from the usual shape of the graph: a period of decreasing failures (the early-failure period), followed by a relatively steady period (the constant-failure period), followed by a period of increasing failures (the wearout-failure period).

benchmark. (1) A standard against which measurements or comparisons can be made.
(2) A procedure, problem, or test that can be used to compare systems or components to each other or to a standard as in (1).
(3) A recovery file.

big-bang testing. A type of integration testing in which software elements, hardware elements, or both are combined all at once into an overall system, rather than in stages.

binary digit (bit). (1) A unit of information that can be represented by either a zero or a one.
(2) An element of computer storage that can hold a unit of information as in (1).
(3) A numeral used to represent one of the two digits in the binary numeration system; zero (0) or one (1).
See also: **byte; word.**

bind. To assign a value to an identifier. For example, to assign a value to a parameter or to assign an absolute address to a symbolic address in a computer program. *See also:* **dynamic binding; static binding.**

bit. Acronym for **binary digit.**

bit steering. A microprogramming technique in which the meaning of a field in a microinstruction is dependent on the value of another field in the microinstruction. *Syn:* **immediate control.** *Contrast with:* **residual control.** *See also:* **two-level encoding.**

black box. (1) A system or component whose inputs, outputs, and general function are

known but whose contents or implementation are unknown or irrelevant. *Contrast with:* **glass box.**
(2) Pertaining to an approach that treats a system or component as in (1). *See also:* **encapsulation.**

black-box testing. *See:* **functional testing (1).**

block. (1) A group of contiguous storage locations, computer program statements, records, words, characters, or bits that are treated as a unit. *See also:* **block-structured language; delimiter.**
(2) To form a group as in (1). *Contrast with:* **deblock.**

block allocation. *See:* **paging (1).**

block diagram. A diagram of a system, computer, or device in which the principal parts are represented by suitably annotated geometrical figures to show both the functions of the parts and their functional relationships. *Syn:* **configuration diagram; system resources chart.** *See also:* **box diagram; bubble chart; flowchart; graph; input-process-output chart; structure chart.**

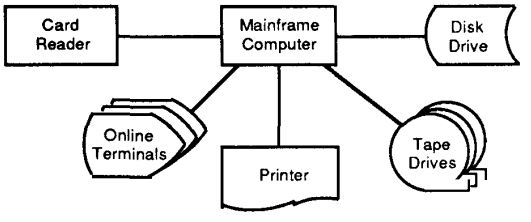

Fig 1
Block Diagram

block-structured language. A design or programming language in which sequences of statements, called blocks, are defined, usually with begin and end delimiters, and variables or labels defined in one block are not recognized outside that block. Examples include **Ada, ALGOL, PL/I.** *See also:* **structured programming language.**

blocking factor. The number or records, words, characters, or bits in a block.

boot. To initialize a computer system by clearing memory and reloading the operating system. Derived from **bootstrap.**

bootstrap. (1) A short computer program that is permanently resident or easily loaded into a computer and whose execution brings a larger program, such as an operating system or its loader, into memory.
(2) To use a program as in (1). *Syn:* **initial program load.**

bootstrap loader. A short computer program used to load a bootstrap.

bottom-up. Pertaining to an activity that starts with the lowest-level components of a hierarchy and proceeds through progressively higher levels; for example, bottom-up design; bottom-up testing. *Contrast with:* **top-down.** *See also:* **critical piece first.**

boundary value. A data value that corresponds to a minimum or maximum input, internal, or output value specified for a system or component. *See also:* **stress testing.**

box diagram. A control flow diagram consisting of a rectangle that is subdivided to show sequential steps, if-then-else conditions, repetition, and case conditions. *Syn:* **Chapin chart; Nassi-Shneiderman chart; program structure diagram.** *See also:* **block diagram; bubble chart; flowchart; graph; input-process-output chart; program structure diagram; structure chart.**

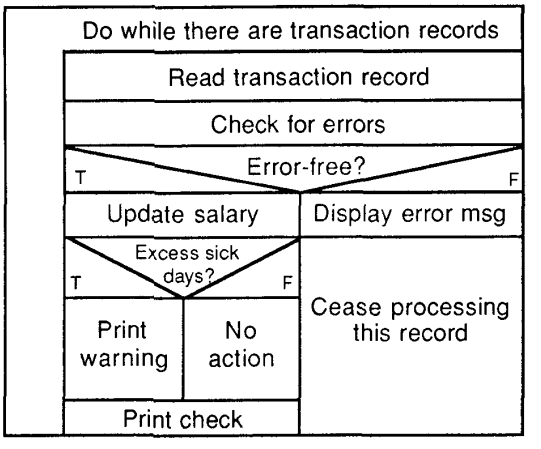

Fig 2
Box Diagram

branch. (1) A computer program construct in which one of two or more alternative sets of

program statements is selected for execution. *See also:* **case; jump; go to; if-then-else.**
(2) A point in a computer program at which one of two or more alternative sets of program statements is selected for execution. *Syn:* **branchpoint.**
(3) Any of the alternative sets of program statements in (1).
(4) To perform the selection in (1).

branch testing. Testing designed to execute each outcome of each decision point in a computer program. *Contrast with:* **path testing; statement testing.**

branchpoint. *See:* **branch (2).**

breakpoint. A point in a computer program at which execution can be suspended to permit manual or automated monitoring of program performance or results. Types include code breakpoint, data breakpoint, dynamic breakpoint, epilog breakpoint, programmable breakpoint, prolog breakpoint, static breakpoint. *Note:* A breakpoint is said to be set when both a point in the program and an event that will cause suspension of execution at that point are defined; it is said to be initiated when program execution is suspended.

bubble chart. A data flow, data structure, or other diagram in which entities are depicted with circles (bubbles) and relationships are represented by links drawn between the circles. *See also:* **block diagram; box diagram; flowchart; graph; input-process-output chart; structure chart.**

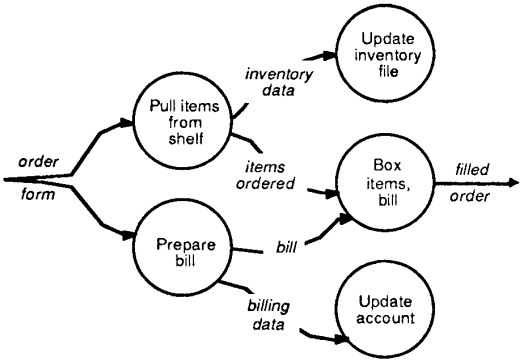

Fig 3
Bubble Chart

buffer. (1) A device or storage area used to store data temporarily to compensate for differences in rates of data flow, time of occurrence of events, or amounts of data that can be handled by the devices or processes involved in the transfer or use of the data.
(2) A routine that accomplishes the objectives in (1).
(3) To allocate, schedule, or use devices or storage areas as in (1). *See also:* **anticipatory buffering; dynamic buffering; simple buffering.**

bug. *See:* **error; fault.**

bug seeding. *See:* **error seeding.**

build. An operational version of a system or component that incorporates a specified subset of the capabilities that the final product will provide.

burn-in period. *See:* **early-failure period.**

busy. Pertaining to a system or component that is operational, in service, and in use. *See also:* **down; idle; up.**

busy time. In computer performance engineering, the period of time during which a system or component is operational, in service, and in use. *See also:* **down time; idle time; set-up time; up time.**

byte. (1) A group of adjacent binary digits operated upon as a unit and usually shorter than a computer word (frequently connotes a group of eight bits).
(2) An element of computer storage that can hold a group of bits as in (1).
See also: **bit; word.**

call. (1) A transfer of control from one software module to another, usually with the implication that control will be returned to the calling module. *Contrast with:* **go to.**
(2) A computer instruction that transfers control from one software module to another as in (1) and, often, specifies the parameters to be passed to and from the module.
(3) To transfer control from one software module to another as in (1) and, often, to pass

parameters to the other module. *Syn:* **cue.**
See also: **call by name; call by reference; call by value; call list; calling sequence.**

call by address. *See:* **call by reference.**

call by location. *See:* **call by reference.**

call by name. A method for passing parameters, in which the calling module provides to the called module a symbolic expression representing the parameter to be passed, and a service routine evaluates the expression and provides the resulting value to the called module. *Note:* Because the expression is evaluated each time its corresponding formal parameter is used in the called module, the value of the parameter may change during the execution of the called module. *Contrast with:* **call by reference; call by value.**

call by reference. A method for passing parameters, in which the calling module provides to the called module the address of the parameter to be passed. *Note:* With this method, the called module has the ability to change the value of the parameter stored by the calling module. *Syn:* **call by address; call by location.** *Contrast with:* **call by name; call by value.**

call by value. A method of passing parameters, in which the calling module provides to the called module the actual value of the parameter to be passed. *Note:* With this method, the called module cannot change the value of the parameter as stored by the calling module. *Contrast with:* **call by name; call by reference.**

call graph. A diagram that identifies the modules in a system or computer program and shows which modules call one another. *Note:* The result is not necessarily the same as that shown in a structure chart. *Syn:* **call tree; tier chart.** *Contrast with:* **structure chart.** *See also:* **control flow diagram; data flow diagram; data structure diagram; state diagram.**

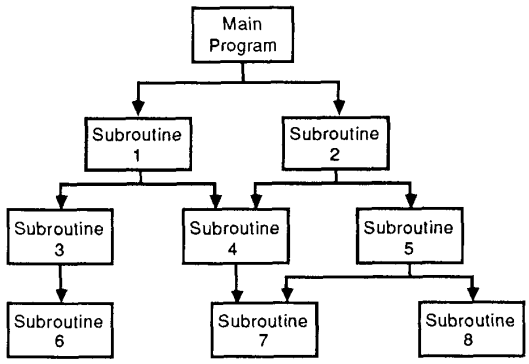

Fig 4
Call Graph

call list. The ordered list of arguments used in a call to a software module.

call trace. *See:* **subroutine trace.**

call tree. *See:* **call graph.**

calling sequence. A sequence of computer instructions and, possibly, data necessary to perform a call to another module.

CASE. Acronym for **computer-aided software engineering.**

case. A single-entry, single-exit multiple-way branch that defines a control expression, specifies the processing to be performed for each value of the control expression, and returns control in all instances to the statement immediately following the overall construct. *Syn:* **multiple exclusive selective construct.** *Contrast with:* **go to; jump; if-then-else.** *See also:* **multiple inclusive selective construct.**

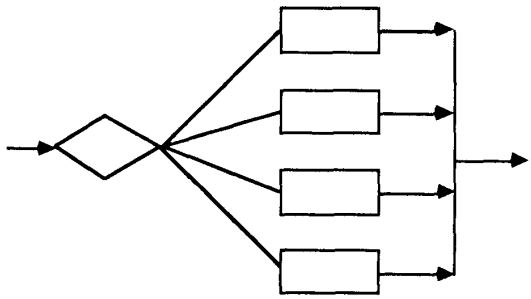

Fig 5
Case Construct

catastrophic failure. A failure of critical software.

CCB. (1) Acronym for **configuration control board.**
(2) Acronym for **change control board.** *See:* **configuration control board.**

CDR. Acronym for **critical design review.**

certification. (1) A written guarantee that a system or component complies with its specified requirements and is acceptable for operational use. For example, a written authorization that a computer system is secure and is permitted to operate in a defined environment.
(2) A formal demonstration that a system or component complies with its specified requirements and is acceptable for operational use.
(3) The process of confirming that a system or component complies with its specified requirements and is acceptable for operational use.

change control. *See:* **configuration control.**

change control board. *See:* **configuration control board.**

change dump. A selective dump of those storage locations whose contents have changed since some specified time or event. *Syn:* **differential dump.** *See also:* **dynamic dump; memory dump; postmortem dump; selective dump; snapshot dump; static dump.**

channel capacity. The maximum amount of information that can be transferred on a given channel per unit of time; usually measured in bits per second or in baud. *See also:* **memory capacity; storage capacity.**

Chapin chart. *See:* **box diagram.**

character. A letter, digit, or other symbol that is used to represent information.

character type. A data type whose members can assume the values of specified characters and can be operated on by character operators, such as concatenation. *Contrast with:* **enumeration type; integer type; logical type; real type.**

characteristic. (IEEE Std 1008-1987 [10]) *See:* **data characteristic; software characteristic.**

checkout. Testing conducted in the operational or support environment to ensure that a software product performs as required after installation.

checkpoint. A point in a computer program at which program state, status, or results are checked or recorded.

chief programmer. The leader of a chief programmer team; a senior-level programmer whose responsibilities include producing key portions of the software assigned to the team, coordinating the activities of the team, reviewing the work of the other team members, and having an overall technical understanding of the software being developed. *See also:* **backup programmer; chief programmer team.**

chief programmer team. A software development group that consists of a chief programmer, a backup programmer, a secretary/librarian, and additional programmers and specialists as needed, and that employs procedures designed to enhance group communication and to make optimum use of each member's skills. *See also:* **backup programmer; chief programmer; egoless programming.**

CI. Acronym for **configuration item.**

clear. To set a variable, register, or other storage location to zero, blank, or other null value. *See also:* **initialize; reset.**

closed loop. A loop that has no exit and whose execution can be interrupted only by intervention from outside the computer program or procedure in which the loop is located. *Contrast with:* **UNTIL; WHILE.**

closed subroutine. A subroutine that is stored at one given location rather than being copied into a computer program at each place that it is called. *Contrast with:* **open subroutine.**

CM. Acronym for **configuration management.**

code. (1) In software engineering, computer instructions and data definitions expressed in a programming language or in a form output by an assembler, compiler, or other translator. *See also:* **source code; object code; machine code; microcode.**
(2) To express a computer program in a programming language.
(3) A character or bit pattern that is assigned a particular meaning; for example, a status code.

code breakpoint. A breakpoint that is initiated upon execution of a given computer instruction. *Syn:* **control breakpoint.** *Contrast with:* **data breakpoint.** *See also:* **dynamic breakpoint; epilog breakpoint; programmable breakpoint; prolog breakpoint; static breakpoint.**

code generator. (1) A routine, often part of a compiler, that transforms a computer program from some intermediate level of representation (often the output of a root compiler or parser) into a form that is closer to the language of the machine on which the program will execute.
(2) A software tool that accepts as input the requirements or design for a computer program and produces source code that implements the requirements or design. *Syn:* **source code generator.** *See also:* **application generator.**

code inspection. *See:* **inspection.**

code of ethics standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of a set of moral principles dealing with accepted standards of conduct by, within, and among professionals.

code review. A meeting at which software code is presented to project personnel, managers, users, customers, or other interested parties for comment or approval. *Contrast with:* **design review; formal qualification review; requirements review; test readiness review.**

code trace. *See:* **execution trace.**

coding. (1) In software engineering, the process of expressing a computer program in a programming language.
(2) (IEEE Std 1002-1987 [9]) The transforming of logic and data from design specifications (design descriptions) into a programming language.
See also: **software development process.**

cohesion. The manner and degree to which the tasks performed by a single software module are related to one another. Types include coincidental, communicational, functional, logical, procedural, sequential, and temporal. *Syn:* **module strength.** *Contrast with:* **coupling.**

coincidental cohesion. A type of cohesion in which the tasks performed by a software module have no functional relationship to one another. *Contrast with:* **communicational cohesion; functional cohesion; logical cohesion; procedural cohesion; sequential cohesion; temporal cohesion.**

command. An expression that can be input to a computer system to initiate an action or affect the execution of a computer program; for example, the "log on" command to initiate a computer session.

command-driven. Pertaining to a system or mode of operation in which the user directs the system through commands. *Contrast with:* **menu-driven.**

command language. A language used to express commands to a computer system. *See also:* **command-driven.**

comment. Information embedded within a computer program, job control statements, or a set of data, that provides clarification to human readers but does not affect machine interpretation.

common. *See:* **common storage.**

common area. *See:* **common storage.**

common block. *See:* **common storage.**

common coupling. *See:* **common-environment coupling.**

common data. *See:* **global data.**

common-environment coupling. A type of coupling in which two software modules access a common data area. *Syn:* **common coupling.** *Contrast with:* **content coupling; control coupling; data coupling; hybrid coupling; pathological coupling.**

common storage. A portion of main storage that can be accessed by two or more modules in a software system. *Syn:* **common area; common block.** *See also:* **global data.**

communicational cohesion. A type of cohesion in which the tasks performed by a software module use the same input data or contribute to producing the same output data. *Contrast with:* **coincidental cohesion; functional cohesion; logical cohesion; procedural cohesion; sequential cohesion; temporal cohesion.**

compaction. In microprogramming, the process of converting a microprogram into a functionally equivalent microprogram that is faster or shorter than the original. *See also:* **local compaction; global compaction.**

comparator. A software tool that compares two computer programs, files, or sets of data to identify commonalities or differences. Typical objects of comparison are similar versions of source code, object code, data base files, or test results.

compatibility. (1) The ability of two or more systems or components to perform their required functions while sharing the same hardware or software environment.
(2) The ability of two or more systems or components to exchange information.
See also: **interoperability.**

compile. To translate a computer program expressed in a high order language into its machine language equivalent. *Contrast with:* **assemble; decompile; interpret.**

compile-and-go. An operating technique in which there are no stops between the compiling, linking, loading, and execution of a computer program.

compiler. A computer program that translates programs expressed in a high order language into their machine language equivalents. *Contrast with:* **assembler; interpreter.** *See also:* **cross-compiler; incremental compiler; root compiler.**

compiler code. Computer instructions and data definitions expressed in a form that can be recognized and processed by a compiler. *Contrast with:* **assembly code; interpretive code; machine code.**

compiler compiler. *See:* **compiler generator.**

compiler generator. A translator or interpreter used to construct part or all of a compiler. *Syn:* **compiler compiler; metacompiler.**

completion code. A code communicated to a job stream processor by a batch program to influence the execution of succeeding steps in the input stream.

complexity. (1) The degree to which a system or component has a design or implementation that is difficult to understand and verify. *Contrast with:* **simplicity.**
(2) Pertaining to any of a set of structure-based metrics that measure the attribute in (1).

component. One of the parts that make up a system. A component may be hardware or software and may be subdivided into other components. *Note:* The terms "module," "component," and "unit" are often used interchangeably or defined to be subelements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized.

component standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of data or program components.

component testing. Testing of individual hardware or software components or groups of related components. *Syn:* **module testing.** *See also:* **integration testing; interface testing; system testing; unit testing.**

composite type. A data type each of whose members is composed of multiple data items. For example, a data type called PAIRS whose members are ordered pairs (x,y). *Contrast with:* **atomic type.**

computer-aided software engineering (CASE). The use of computers to aid in the software engineering process. May include the application of software tools to software design, requirements tracing, code production, testing, document generation, and other software engineering activities.

computer instruction. (1) A statement in a programming language, specifying an operation to be performed by a computer and the addresses or values of the associated operands; for example, Move A to B. *See also:* **instruction format; instruction set.**
(2) Loosely, any executable statement in a computer program.

computer language. A language designed to enable humans to communicate with computers. *See also:* **design language; query language; programming language.** *Note:* P610.13 [17] defines specific computer languages.

computer performance evaluation. An engineering discipline that measures the performance of computer systems and investigates methods by which that performance can be improved. *See also:* **system profile; throughput; utilization; workload model.**

computer program. A combination of computer instructions and data definitions that enable computer hardware to perform computational or control functions. *See also:* **software.**

computer program abstract. A brief description of a computer program that provides sufficient information for potential users to determine the appropriateness of the program to their needs and resources.

computer program component (CPC).* *See:* **computer software component.**
*Deprecated.

computer program configuration item (CPCI).* *See:* **computer software configuration item.**
*Deprecated.

computer resource allocation. The assignment of computer resources to current and waiting jobs; for example, the assignment of main memory, input/output devices, and auxiliary storage to jobs executing concurrently in a computer system. *See also:* **dynamic resource allocation; storage allocation.**

computer resources. The computer equipment, programs, documentation, services, facilities, supplies, and personnel available for a given purpose. *See also:* **computer resource allocation.**

computer security. *Note:* P610.9 [16] defines terminology pertaining to computer security.

computer software component (CSC). A functionally or logically distinct part of a computer software configuration item, typically an aggregate of two or more software units.

computer software configuration item (CSCI). An aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process. *Contrast with:* **hardware configuration item.** *See also:* **configuration item.**

computer system. A system containing one or more computers and associated software.

computer word. *See:* **word.**

computing center. A facility designed to provide computer services to a variety of users through the operation of computers and auxiliary hardware and through services provided by the facility's staff.

concept phase. (1) (IEEE Std 1002-1987 [9]) The period of time in the software development cycle during which the user needs are described and evaluated through documentation (for example, statement of needs,

advance planning report, project initiation memo, feasibility studies, system definition, documentation, regulations, procedures, or policies relevant to the project).
(2) (IEEE Std 1012-1986 [12]) The initial phase of a software development project, in which the user needs are described and evaluated through documentation (for example, statement of needs, advance planning report, project initiation memo, feasibility studies, system definition, documentation, regulations, procedures, or policies relevant to the project).

concurrent. Pertaining to the occurrence of two or more activities within the same interval of time, achieved either by interleaving the activities or by simultaneous execution. *Syn:* **parallel (2).** *Contrast with:* **simultaneous.**

condition code. *See:* **status code.**

conditional branch.* *See:* **conditional jump.**
*Deprecated.

conditional jump. A jump that takes place only when specified conditions are met. *Contrast with:* **unconditional jump.**

configuration. (1) The arrangement of a computer system or component as defined by the number, nature, and interconnections of its constituent parts.
(2) In configuration management, the functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product. *See also:* **configuration item; form, fit and function; version.**

configuration audit. *See:* **functional configuration audit; physical configuration audit.**

configuration control. An element of configuration management, consisting of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification. *Syn:* **change control.** *Contrast with:* **configuration identification; configuration status accounting.** *See also:* **configuration control board; deviation; engineering change; interface control; notice of revision; specification change notice; waiver.**

configuration control board (CCB). A group of people responsible for evaluating and approving or disapproving proposed changes to configuration items, and for ensuring implementation of approved changes. *Syn:* **change control board.** *See also:* **configuration control.**

configuration diagram. *See:* **block diagram.**

configuration identification. (1) An element of configuration management, consisting of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation. *Contrast with:* **configuration control; configuration status accounting.**
(2) The current approved technical documentation for a configuration item as set forth in specifications, drawings, associated lists, and documents referenced therein. *See also:* **allocated configuration identification; functional configuration identification; product configuration identification; baseline.**

configuration index. A document used in configuration management, providing an accounting of the configuration items that make up a product. *See also:* **configuration item development record; configuration status accounting.**

configuration item (CI). An aggregation of hardware, software, or both, that is designated for configuration management and treated as a single entity in the configuration management process. *See also:* **hardware configuration item; computer software configuration item; configuration identification; critical item.**

configuration item development record. A document used in configuration management, describing the development status of a configuration item based on the results of configuration audits and design reviews. *See also:* **configuration index; configuration status accounting.**

configuration management (CM). A discipline applying technical and administra-

tive direction and surveillance to: identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. *See also:* **baseline; configuration identification; configuration control; configuration status accounting; configuration audit.**

configuration status accounting. An element of configuration management, consisting of the recording and reporting of information needed to manage a configuration effectively. This information includes a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of approved changes. *Contrast with:* **configuration control; configuration identification.** *See also:* **configuration index; configuration item development record.**

consecutive. Pertaining to the occurrence of two sequential events or items without the intervention of any other event or item; that is, one immediately after the other.

consistency. The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a system or component. *See also:* **traceability.**

constant. A quantity or data item whose value cannot change; for example, the data item FIVE, with an unchanging value of 5. *Contrast with:* **variable.** *See also:* **figurative constant; literal.**

constant-failure period. The period of time in the life cycle of a system or component during which hardware failures occur at an approximately uniform rate. *Contrast with:* **early-failure period; wearout-failure period.** *See also:* **bathtub curve.**

content coupling. A type of coupling in which some or all of the contents of one software module are included in the contents of another module. *Contrast with:* **common-environment coupling; control coupling; data coupling; hybrid coupling; pathological coupling.**

contiguous allocation. A storage allocation technique in which programs or data to be stored are allocated a block of storage of equal or greater size, so that logically contiguous programs and data are assigned physically contiguous storage locations. *Contrast with:* **paging (1).**

continuous iteration. A loop that has no exit.

control breakpoint. *See:* **code breakpoint.**

control coupling. A type of coupling in which one software module communicates information to another module for the explicit purpose of influencing the latter module's execution. *Contrast with:* **common-environment coupling; content coupling; data coupling; hybrid coupling; pathological coupling.**

control data. Data that select an operating mode, direct the sequential flow of a program, or otherwise directly influence the operation of software; for example, a loop control variable.

control flow. The sequence in which operations are performed during the execution of a computer program. *Syn:* **flow of control.** *Contrast with:* **data flow.**

control flow diagram. A diagram that depicts the set of all possible sequences in which operations may be performed during the execution of a system or program. Types include box diagram, flowchart, input-process-output chart, state diagram. *Contrast with:* **data flow diagram.** *See also:* **call graph; structure chart.**

control flow trace. *See:* **execution trace.**

control language. *See:* **job control language.**

control program. *See:* **supervisory program.**

control statement. A program statement that selects among alternative sets of program statements or affects the order in which operations are performed. For example, if-then-else, case. *Contrast with:* **assignment statement; declaration.**

control store. In a microprogrammed computer, the computer memory in which microprograms reside. *See also:* **microword; nanostore.**

control variable. *See:* **loop-control variable.**

conventions. (IEEE Std 983-1986 [7]) Requirements employed to prescribe a disciplined uniform approach to providing consistency in a software product, that is, uniform patterns or forms for arranging data. *See also:* **practices; standards.**

conversational. Pertaining to an interactive system or mode of operation in which the interaction between the user and the system resembles a human dialog. *Contrast with:* **batch.** *See also:* **interactive; on-line; real time.**

conversational compiler. *See:* **incremental compiler.**

conversion. Modification of existing software to enable it to operate with similar functional capability in a different environment; for example, converting a program from Fortran to Ada, converting a program that runs on one computer to run on another.

copy. (1) To read data from a source, leaving the source data unchanged, and to write the same data elsewhere in a physical form that may differ from that of the source. For example, to copy data from a magnetic disk onto a magnetic tape. *Contrast with:* **move.**
(2) The result of a copy process as in (1). For example, a copy of a data file.

core dump.* *See:* **memory dump.**
*Deprecated.

coroutine. A routine that begins execution at the point at which operation was last suspended, and that is not required to return control to the program or subprogram that called it. *Contrast with:* **subroutine.**

corrective maintenance. Maintenance performed to correct faults in hardware or software. *Contrast with:* **adaptive maintenance; perfective maintenance.**

correctness. (1) The degree to which a system or component is free from faults in its specification, design, and implementation.
(2) The degree to which software, documentation, or other items meet specified requirements.
(3) The degree to which software, documentation, or other items meet user needs and expectations, whether specified or not.

correctness proof. *See:* **proof of correctness.**

counter. A variable used to record the number of occurrences of a given event during the execution of a computer program; for example, a variable that records the number of times a loop is executed.

coupling. The manner and degree of interdependence between software modules. Types include common-environment coupling, content coupling, control coupling, data coupling, hybrid coupling, and pathological coupling. *Contrast with:* **cohesion.**

CPC. Acronym for **computer program component.** *See:* **computer software component.**

CPCI. Acronym for **computer program configuration item.** *See:* **computer software configuration item.**

crash. The sudden and complete failure of a computer system or component. *See also:* **hard failure.**

critical design review (CDR). (1) A review conducted to verify that the detailed design of one or more configuration items satisfy specified requirements; to establish the compatibility among the configuration items and other items of equipment, facilities, software, and personnel; to assess risk areas for each configuration item; and, as applicable, to assess the results of producibility analyses, review preliminary hardware product specifications, evaluate preliminary test planning, and evaluate the adequacy of preliminary operation and support documents. *See also:* **preliminary design review; system design review.**
(2) A review as in (1) of any hardware or software component.

critical item. In configuration management, an item within a configuration item that, because of special engineering or logistic considerations, requires an approved specification to establish technical or inventory control at the component level.

critical piece first. A system development approach in which the most critical aspects of a system are implemented first. The critical piece may be defined in terms of services provided, degree of risk, difficulty, or other criteria. *See also:* **bottom-up; top-down.**

critical software. (IEEE Std 1012-1986 [12]) Software whose failure could have an impact on safety, or could cause large financial or social loss.

criticality. The degree of impact that a requirement, module, error, fault, failure, or other item has on the development or operation of a system. *Syn:* **severity.**

cross-assembler. An assembler that executes on one computer but generates machine code for a different computer.

cross-compiler. A compiler that executes on one computer but generates machine code for a different computer.

cross-reference generator. A software tool that accepts as input the source code of a computer program and produces as output a listing that identifies each of the program's variables, labels, and other identifiers and indicates which statements in the program define, set, or use each one. *Syn:* **cross-referencer.**

cross-reference list. A list that identifies each of the variables, labels, and other identifiers in a computer program and indicates which statements in the program define, set, or use each one.

cross-referencer. *See:* **cross-reference generator.**

CSC. Acronym for **computer software component.**

CSCI. Acronym for **computer software configuration item.**

cue. *See:* **call (3).**

curriculum standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of a course of study on a body of knowledge that is offered by an educational institution.

cycle. (1) A period of time during which a set of events is completed. *See also:* **software development cycle; software life cycle.**
(2) A set of operations that is repeated regularly in the same sequence, possibly with variations in each repetition; for example, a computer's read cycle. *See also:* **pass.**

cycle stealing. The process of suspending the operation of a central processing unit for one or more cycles to permit the occurrence of other operations, such as transferring data from main memory in response to an output request from an input/output controller.

cyclic search. A storage allocation technique in which each search for a suitable block of storage begins with the block following the one last allocated.

data. (1) A representation of facts, concepts, or instructions in a manner suitable for communication, interpretation, or processing by humans or by automatic means. *See also:* **data type.** *Note:* IEEE Std 610.5-1990 [2] defines terminology pertaining to data management.
(2) Sometimes used as a synonym for documentation.

data abstraction. (1) The process of extracting the essential characteristics of data by defining data types and their associated functional characteristics and disregarding representation details. *See also:* **encapsulation; information hiding.**
(2) The result of the process in (1).

data breakpoint. A breakpoint that is initiated when a specified data item is accessed. *Syn:* **storage breakpoint.** *Contrast with:* **code breakpoint.** *See also:* **dynamic breakpoint; epilog breakpoint; programmable breakpoint; prolog breakpoint; static breakpoint.**

data characteristic. (IEEE Std 1008-1987 [10]) An inherent, possibly accidental, trait,

quality, or property of data (for example, arrival rates, formats, value ranges, or relationships between field values).

data coupling. A type of coupling in which output from one software module serves as input to another module. *Syn:* **input-output coupling.** *Contrast with:* **common-environment coupling; content coupling; control coupling; hybrid coupling; pathological coupling.**

data definition. *Note:* This term is defined in IEEE Std 610.5-1990 [2].

data exception. An exception that occurs when a program attempts to use or access data incorrectly. *See also:* **addressing exception; operation exception; overflow exception; protection exception; underflow exception.**

data flow. The sequence in which data transfer, use, and transformation are performed during the execution of a computer program. *Contrast with:* **control flow.**

data flow diagram (DFD). A diagram that depicts data sources, data sinks, data storage, and processes performed on data as nodes, and logical flow of data as links between the nodes. *Syn:* **data flowchart; data flow graph.** *Contrast with:* **control flow diagram; data structure diagram.**

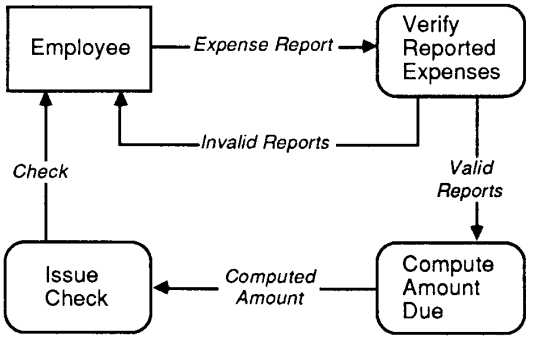

Fig 6
Data Flow Diagram

data flow graph. *See:* **data flow diagram.**

data flow trace. *See:* **variable trace.**

data flowchart (flow chart). *See:* **data flow diagram.**

data input sheet. User documentation that describes, in a worksheet format, the required and optional input data for a system or component. *See also:* **user manual.**

data-sensitive fault. A fault that causes a failure in response to some particular pattern of data. *Syn:* **pattern-sensitive fault.** *Contrast with:* **program-sensitive fault.**

data structure. A physical or logical relationship among data elements, designed to support specific data manipulation functions. *Note:* IEEE Std 610.5-1990 [2] defines specific data structures.

data structure-centered design. A software design technique in which the architecture of a system is derived from analysis of the structure of the data sets with which the system must deal. *See also:* **input-process-output; modular decomposition; object-oriented design; rapid prototyping; stepwise refinement; structure clash; structured design; transaction analysis; transform analysis.**

data structure diagram. A diagram that depicts a set of data elements, their attributes, and the logical relationships among them. *Contrast with:* **data flow diagram.** *See also:* **entity-relationship diagram.**

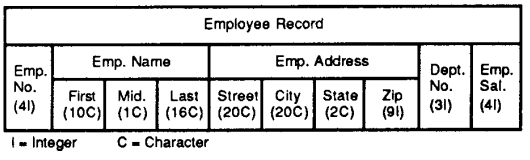

Fig 7
Data Structure Diagram

data trace. *See:* **variable trace.**

data type. A class of data, characterized by the members of the class and the operations that can be applied to them. For example, character type, enumeration type, integer type, logical type, real type. *See also:* **strong typing.**

database. A collection of interrelated data stored together in one or more computerized files. *Note:* IEEE Std 610.5-1990 [2] defines terminology pertaining to databases.

datum. Singular for **data.**

deadlock. A situation in which computer processing is suspended because two or more devices or processes are each awaiting resources assigned to the others. *See also:* **lockout.**

deassembler.* *See:* **disassembler.**
* Deprecated.

deblock. To separate the parts of a block. *Contrast with:* **block (2).**

debug. To detect, locate, and correct faults in a computer program. Techniques include use of breakpoints, desk checking, dumps, inspection, reversible execution, single-step operation, and traces.

decision table. A table used to show sets of conditions and the actions resulting from them.

declaration. A non-executable program statement that affects the assembler or compiler's interpretation of other statements in the program. For example, a statement that identifies a name, specifies what the name represents, and, possibly, assigns it an initial value. *Contrast with:* **assignment statement; control statement.** *See also:* **pseudo instruction.**

declarative language. A nonprocedural language that permits the user to declare a set of facts and to express queries or problems that use these facts. *See also:* **interactive language; rule-based language.**

decompile. To translate a compiled computer program from its machine language version into a form that resembles, but may not be identical to, the original high order language program. *Contrast with:* **compile.**

decompiler. A software tool that decompiles computer programs.

decoupling. The process of making software modules more independent of one another to decrease the impact of changes to, and errors in, the individual modules. *See also:* **coupling.**

delimiter. A character or set of characters used to denote the beginning or end of a group of related bits, characters, words, or statements.

delivery. Release of a system or component to its customer or intended user. *See also:* **software life cycle; system life cycle.**

demand paging. A storage allocation technique in which pages are transferred from auxiliary storage to main storage only when those pages are needed. *Contrast with:* **anticipatory paging.**

demodularization. In software design, the process of combining related software modules, usually to optimize system performance. *See also:* **downward compression; lateral compression; upward compression.**

demonstration. A dynamic analysis technique that relies on observation of system or component behavior during execution, without need for post-execution analysis, to detect errors, violations of development standards, and other problems. *See also:* **testing.**

derived type. A data type whose members and operations are taken from those of another data type according to some specified rule. *See also:* **subtype.**

description standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of product information or procedures provided to help understand, test, install, operate, or maintain the product.

design. (1) The process of defining the architecture, components, interfaces, and other characteristics of a system or component. *See also:* **architectural design; preliminary design; detailed design.**
(2) The result of the process in (1).

design description. A document that describes the design of a system or component.

Typical contents include system or component architecture, control logic, data structures, input/ output formats, interface descriptions, and algorithms. *Syn:* **design document; design specification.** *See also:* **product specification.** *Contrast with:* **requirements specification.**

design document. *See:* **design description.**

design element. (IEEE Std 990-1987 [8]) A basic component or building block in a design.

design entity. (IEEE Std 1016-1987 [13]) An element (component) of a design that is structurally and functionally distinct from other elements and that is separately named and referenced.

design inspection. *See:* **inspection.**

design language. A specification language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document a hardware or software design. Types include hardware design language, program design language. *See also:* **requirements specification language.**

design level. (IEEE Std 829-1983 [5]) The design decomposition of the software item (for example, system, subsystem, program, or module).

design phase. The period of time in the software life cycle during which the designs for architecture, software components, interfaces, and data are created, documented, and verified to satisfy requirements. *See also:* **detailed design; preliminary design.**

design requirement. A requirement that specifies or constrains the design of a system or system component. *Contrast with:* **functional requirement; implementation requirement; interface requirement; performance requirement; physical requirement.**

design review. A process or meeting during which a system, hardware, or software design is presented to project personnel, managers, users, customers, or other interested parties for comment or approval. Types include critical design review, preliminary design review, system design review. *Contrast with:* **code review; formal qualification review; requirements review; test readiness review.**

design specification. *See:* **design description.**

design standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of a design or a design description of data or program components.

design unit. (IEEE Std 990-1987 [8]) A logically related collection of design elements. In an Ada PDL, a design unit is represented by an Ada compilation unit.

design view. (IEEE Std 1016-1987 [13]) A subset of design entity attribute information that is specifically suited to the needs of a software project activity.

desk checking. A static analysis technique in which code listings, test results, or other documentation are visually examined, usually by the person who generated them, to identify errors, violations of development standards, or other problems. *See also:* **inspection; walk-through.**

destination address. The address of the device or storage location to which data is to be transferred. *Contrast with:* **source address.**

destructive read. A read operation that alters the data in the accessed location. *Contrast with:* **nondestructive read.**

detailed design. (1) The process of refining and expanding the preliminary design of a system or component to the extent that the design is sufficiently complete to be implemented. *See also:* **software development process.**
(2) The result of the process in (1).

development cycle. *See:* **software development cycle.**

development life cycle. *See:* **software development cycle.**

development specification. *See:* **requirements specification.**

development testing. Formal or informal testing conducted during the development of a system or component, usually in the development environment by the developer. *Contrast with:* **acceptance testing; operational testing.** *See also:* **qualification testing.**

developmental baseline.* *See:* **developmental configuration.**
*Deprecated.

developmental configuration. In configuration management, the software and associated technical documentation that define the evolving configuration of a computer software configuration item during development. *Note:* The developmental configuration is under the developer's control, and therefore is not called a baseline. *Contrast with:* **allocated baseline; functional baseline; product baseline.**

deviation. (1) A departure from a specified requirement.
(2) A written authorization, granted prior to the manufacture of an item, to depart from a particular performance or design requirement for a specific number of units or a specific period of time. *Note:* Unlike an engineering change, a deviation does not require revision of the documentation defining the affected item. *See also:* **configuration control.** *Contrast with:* **engineering change; waiver.**

device. A mechanism or piece of equipment designed to serve a purpose or perform a function.

DFD. Acronym for **data flow diagram.**

diagnostic. Pertaining to the detection and isolation of faults or failures; for example, a diagnostic message, a diagnostic manual.

diagnostic manual. A document that presents the information necessary to execute diagnostic procedures for a system or component, identify malfunctions, and remedy those malfunctions. Typically described are the diagnostic features of the system or component and the diagnostic tools available for its support. *See also:* **installation manual; operator manual; programmer manual; support manual; user manual.**

diagonal microinstruction. A microinstruction capable of specifying a limited number of simultaneous operations needed to carry out a machine language instruction. *Note:* Diagonal microinstructions fall, in size and functionality, between horizontal microinstructions and vertical microinstructions. The designation "diagonal" refers to this compromise rather than to any physical characteristic of the microinstruction. *Contrast with:* **horizontal microinstruction; vertical microinstruction.**

differential dump. *See:* **change dump.**

digraph. *See:* **directed graph.**

direct address. An address that identifies the storage location of an operand. *Syn:* **one-level address.** *Contrast with:* **immediate data; indirect address; *n*-level address.** *See also:* **direct instruction.**

direct insert subroutine. *See:* **open subroutine.**

direct instruction. A computer instruction that contains the direct addresses of its operands. *Contrast with:* **immediate instruction; indirect instruction.** *See also:* **absolute instruction; effective instruction.**

directed graph. A graph (sense 2) in which direction is implied in the internode connections. *Syn:* **digraph.** *Contrast with:* **undirected graph.**

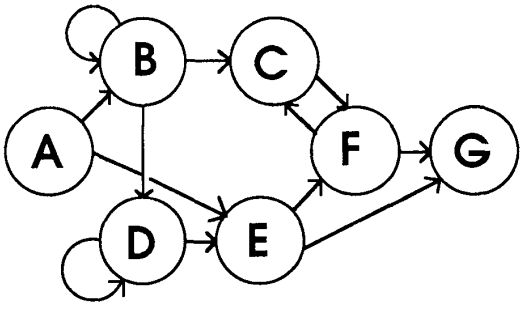

Fig 8
Directed Graph

directory. A list of data items and information about those data items. *Note:* IEEE Std 610.5-1990 [2] defines Data Management terms.

disassemble. To translate an assembled computer program from its machine language version into a form that resembles, but may not be identical to, the original assembly language program. *Contrast with:* **assemble.**

disassembler. A software tool that disassembles computer programs. *Syn:* **deassembler.**

discrete type. A data type whose members can assume any of a set of distinct values. A discrete type may be an enumeration type or an integer type.

diverse redundancy. *See:* **diversity.**

diversity. In fault tolerance, realization of the same function by different means. For example, use of different processors, storage media, programming languages, algorithms, or development teams. *See also:* **software diversity.**

do-nothing operation. *See:* **no-operation.**

document. (1) A medium, and the information recorded on it, that generally has permanence and can be read by a person or a machine. Examples in software engineering include project plans, specifications, test plans, user manuals.
(2) To create a document as in (1).
(3) To add comments to a computer program.

documentation. (1) A collection of documents on a given subject.
(2) Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results.
(3) The process of generating or revising a document.
(4) The management of documents, including identification, acquisition, processing, storage, and dissemination.

documentation tree. A diagram that depicts all of the documents for a given system and shows their relationships to one another. *See also:* **specification tree.**

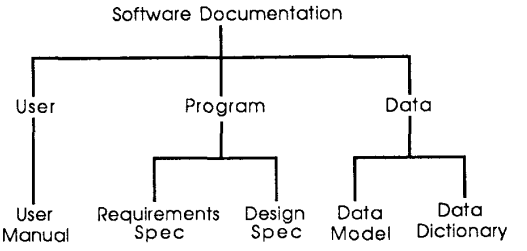

Fig 9
Documentation Tree

double-operand instruction. *See:* **two-address instruction.**

down. Pertaining to a system or component that is not operational or has been taken out of service. *Contrast with:* **up.** *See also:* **busy; crash; idle.**

down time. The period of time during which a system or component is not operational or has been taken out of service. *Contrast with:* **up time.** *See also:* **busy time; idle time; mean time to repair; set-up time.**

downward compatible. Pertaining to hardware or software that is compatible with an earlier or less complex version of itself; for example, a program that handles files created by an earlier version of itself. *Contrast with:* **upward compatible.**

downward compression. In software design, a form of demodularization in which a superordinate module is copied into the body of a subordinate module. *Contrast with:* **lateral compression; upward compression.**

driver. (1) A software module that invokes and, perhaps, controls and monitors the execution of one or more other software modules. *See also:* **test driver.**
(2) A computer program that controls a peripheral device and, sometimes, reformats data for transfer to and from the device.

dual coding. *See:* **software diversity.**

dump. (1) A display of some aspect of a computer program's execution state, usually the contents of internal storage or registers. Types include change dump, dynamic dump, memory dump, postmortem dump, selective dump, snapshot dump; static dump.
(2) A display of the contents of a file or device.
(3) To copy the contents of internal storage to an external medium.
(4) To produce a display or copy as in (1), (2), or (3).

dyadic selective construct. An if-then-else construct in which processing is specified for both outcomes of the branch. *Contrast with:* **monadic selective construct.**

dynamic. Pertaining to an event or process that occurs during computer program execution; for example, dynamic analysis, dynamic binding. *Contrast with:* **static.**

dynamic allocation. *See:* **dynamic resource allocation.**

dynamic analysis. The process of evaluating a system or component based on its behavior during execution. *Contrast with:* **static analysis.** *See also:* **demonstration; testing.**

dynamic binding. Binding performed during the execution of a computer program. *Contrast with:* **static binding.**

dynamic breakpoint. A breakpoint whose predefined initiation event is a runtime characteristic of the program, such as the execution of any twenty source statements. *Contrast with:* **static breakpoint.** *See also:* **code breakpoint; data breakpoint; epilog breakpoint; programmable breakpoint; prolog breakpoint.**

dynamic buffering. A buffering technique in which the buffer allocated to a computer program varies during program execution, based on current need. *Contrast with:* **simple buffering.**

dynamic dump. A dump that is produced during the execution of a computer program. *Contrast with:* **static dump.** *See also:* **change dump; memory dump; postmortem dump; selective dump; snapshot dump.**

dynamic error. An error that is dependent on the time-varying nature of an input. *Contrast with:* **static error.**

dynamic relocation. Relocation of a computer program during its execution.

dynamic resource allocation. A computer resource allocation technique in which the resources assigned to a program vary during program execution, based on current need.

dynamic restructuring. The process of restructuring a database, data structure, computer program, or set of system components during program execution.

dynamic storage allocation. A storage allocation technique in which the storage assigned to a computer program varies during program execution, based on the current needs of the program and of other executing programs.

E-R diagram. Acronym for **entity-relationship diagram.**

early-failure period. The period of time in the life cycle of a system or component during which hardware failures occur at a decreasing rate as problems are detected and repaired. *Contrast with:* **constant-failure period; wearout-failure period.** *Syn:* **burn-in period.** *See also:* **bathtub curve.**

echo. (1) To return a transmitted signal to its source, often with a delay to indicate that the signal is a reflection rather than the original.
(2) A returned signal, as in (1).

ECP. Acronym for **engineering change proposal.**

edit. To modify the form or format of computer code, data, or documentation; for example, to insert, rearrange, or delete characters.

editor. (1) *See:* **text editor.**
(2) *See:* **linkage editor.**

effective address. The address that results from performing any required indexing, indirect addressing, or other address modification on a specified address. *Note:* If the specified address requires no modification, it is also the effective address. *See also:* **generated address; indirect address; relative address.**

effective instruction. The computer instruction that results from performing any required indexing, indirect addressing, or other modification on the addresses in a specified computer instruction. *Note:* If the specified instruction requires no modification, it is also the effective instruction. *See also:* **absolute instruction; direct instruction; immediate instruction; indirect instruction.**

efferent. Pertaining to a flow of data or control from a superordinate module to a subordinate module in a software system. *Contrast with:* **afferent.**

efficiency. The degree to which a system or component performs its designated functions with minimum consumption of resources. *See also:* **execution efficiency; storage efficiency.**

egoless programming. A software development technique based on the concept of team, rather than individual, responsibility for program development. Its purpose is to prevent individual programmers from identifying so closely with their work that objective evaluation is impaired.

embedded computer system. A computer system that is part of a larger system and performs some of the requirements of that system; for example, a computer system used in an aircraft or rapid transit system.

embedded software. Software that is part of a larger system and performs some of the requirements of that system; for example, software used in an aircraft or rapid transit system.

emulation. (1) A model that accepts the same inputs and produces the same outputs as a given system. *See also:* **simulation.**

(2) The process of developing or using a model as in (1).

emulator. A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system. *See also:* **simulator.**

encapsulation. A software development technique that consists of isolating a system function or a set of data and operations on those data within a module and providing precise specifications for the module. *See also:* **data abstraction; information hiding.**

engineering. The application of a systematic, disciplined, quantifiable approach to structures, machines, products, systems, or processes.

engineering change. In configuration management, an alteration in the configuration of a configuration item or other designated item after formal establishment of its configuration identification. *See also:* **configuration control; engineering change proposal.** *Contrast with:* **deviation; waiver.**

engineering change proposal (ECP). In configuration management, a proposed engineering change and the documentation by which the change is described and suggested. *See also:* **configuration control.**

entity. In computer programming, any item that can be named or denoted in a program. For example, a data item, program statement, or subprogram.

entity attribute. (IEEE Std 1016-1987 [13]) A named characteristic or property of a design entity. It provides a statement of fact about the entity.

entity-relationship (E-R) diagram. A diagram that depicts a set of real-world entities and the logical relationships among them. *Syn:* **entity-relationship map.** *See also:* **data structure diagram.**

entity-relationship (E-R) map. *See:* **entity-relationship diagram.**

entrance. *See:* **entry point.**

entry. *See:* **entry point.**

entry point. A point in a software module at which execution of the module can begin. *Contrast with:* **exit.** *Syn:* **entrance; entry.** *See also:* **reentry point.**

enumeration type. A discrete data type whose members can assume values that are explicitly defined by the programmer. For example, a data type called COLORS with possible values RED, BLUE, and YELLOW. *Contrast with:* **character type; integer type; logical type; real type.**

epilog breakpoint. A breakpoint that is initiated upon exit from a given program or routine. *Syn:* **postamble breakpoint.** *Contrast with:* **prolog breakpoint.** *See also:* **code breakpoint; data breakpoint; dynamic breakpoint; programmable breakpoint; static breakpoint.**

equivalent faults. Two or more faults that result in the same failure mode.

error. (1) The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. For example, a difference of 30 meters between a computed result and the correct result.
(2) An incorrect step, process, or data definition. For example, an incorrect instruction in a computer program.
(3) An incorrect result. For example, a computed result of 12 when the correct result is 10.
(4) A human action that produces an incorrect result. For example, an incorrect action on the part of a programmer or operator.
Note: While all four definitions are commonly used, one distinction assigns definition 1 to the word "error," definition 2 to the word "fault," definition 3 to the word "failure," and definition 4 to the word "mistake." *See also:* **dynamic error; fatal error; indigenous error; semantic error; syntactic error; static error; transient error.**

error model. In software evaluation, a model used to estimate or predict the number of remaining faults, required test time, and similar characteristics of a system. *Syn:* **error prediction model.**

error prediction. A quantitative statement about the expected number or nature of faults in a system or component. *See also:* **error model; error seeding.**

error prediction model. *See:* **error model.**

error seeding. The process of intentionally adding known faults to those already in a computer program for the purpose of monitoring the rate of detection and removal, and estimating the number of faults remaining in the program. *Syn:* **bug seeding; fault seeding.** *See also:* **indigenous error.**

error tolerance. The ability of a system or component to continue normal operation despite the presence of erroneous inputs. *See also:* **fault tolerance; robustness.**

exception. An event that causes suspension of normal program execution. Types include addressing exception, data exception, operation exception, overflow exception, protection exception, underflow exception.

execute. To carry out an instruction, process, or computer program.

execution efficiency. The degree to which a system or component performs its designated functions with minimum consumption of time. *See also:* **execution time; storage efficiency.**

execution monitor. *See:* **monitor (1).**

execution time. The amount of elapsed time or processor time used in executing a computer program. *Note:* Processor time is usually less than elapsed time because the processor may be idle (for example, awaiting needed computer resources) or employed on other tasks during the execution of a program. *Syn:* **run time (3); running time.** *See also:* **overhead time.**

execution trace. A record of the sequence of instructions executed during the execution of a computer program. Often takes the form of a list of code labels encountered as the

program executes. *Syn:* **code trace; control-flow trace.** *See also:* **retrospective trace; subroutine trace; symbolic trace; variable trace.**

executive. *See:* **supervisory program.**

executive program. *See:* **supervisory program.**

executive state. *See:* **supervisor state.**

exit. A point in a software module at which execution of the module can terminate. *Contrast with:* **entry point.** *See also:* **return.**

exit routine. A routine that receives control when a specified event, such as an error, occurs.

expandability. *See:* **extendability.**

explicit address. *See:* **absolute address.**

extendability. The ease with which a system or component can be modified to increase its storage or functional capacity. *Syn:* **expandability; extensibility.** *See also:* **flexibility; maintainability.**

extensibility. *See:* **extendability.**

factoring. (1) The process of decomposing a system into a hierarchy of modules. *See also:* **modular decomposition.**
(2) The process of removing a function from a module and placing it into a module of its own.

fail safe. Pertaining to a system or component that automatically places itself in a safe operating mode in the event of a failure; for example, a traffic light that reverts to blinking red in all directions when normal operation fails. *Contrast with:* **fail soft.** *See also:* **fault secure; fault tolerance.**

fail soft. Pertaining to a system or component that continues to provide partial operational capability in the event of certain failures; for example, a traffic light that continues to alternate between red and green if the yellow light fails. *Contrast with:* **fail safe.** *See also:* **fault secure; fault tolerance.**

failure. The inability of a system or component to perform its required functions within specified performance requirements. *Note:* The fault tolerance discipline distinguishes between a human action (a mistake), its manifestation (a hardware or software fault), the result of the fault (a failure), and the amount by which the result is incorrect (the error). *See also:* **crash; dependent failure; exception; failure mode; failure rate; hard failure; incipient failure; independent failure; random failure; soft failure; stuck failure.**

failure mode. The physical or functional manifestation of a failure. For example, a system in failure mode may be characterized by slow operation, incorrect outputs, or complete termination of execution.

failure rate. The ratio of the number of failures of a given category to a given unit of measure; for example, failures per unit of time, failures per number of transactions, failures per number of computer runs. *Syn:* **failure ratio.**

failure ratio. *See:* **failure rate.**

fatal error. An error that results in the complete inability of a system or component to function.

fault. (1) A defect in a hardware device or component; for example, a short circuit or broken wire.
(2) An incorrect step, process, or data definition in a computer program. *Note:* This definition is used primarily by the fault tolerance discipline. In common usage, the terms "error" and "bug" are used to express this meaning. *See also:* **data-sensitive fault; program sensitive fault; equivalent faults; fault masking; intermittent fault.**

fault dictionary. A list of faults in a system or component, and the tests that have been designed to detect them.

fault masking. A condition in which one fault prevents the detection of another.

fault secure. Pertaining to a system or component in which no failures are produced from a prescribed set of faults. *See also:* **fault tolerance; fail safe; fail soft.**

fault seeding. *See:* **error seeding.**

fault tolerance. (1) The ability of a system or component to continue normal operation despite the presence of hardware or software faults. *See also:* **error tolerance; fail safe; fail soft; fault secure; robustness.**
(2) The number of faults a system or component can withstand before normal operation is impaired.
(3) Pertaining to the study of errors, faults, and failures, and of methods for enabling systems to continue normal operation in the presence of faults. *See also:* **recovery; redundancy; restart.**

fault tolerant. Pertaining to a system or component that is able to continue normal operation despite the presence of faults.

FCA. Acronym for **functional configuration audit.**

feasibility. The degree to which the requirements, design, or plans for a system or component can be implemented under existing constraints.

feature. (IEEE Std 1008-1987 [10]) *See:* **software feature.**

fetch. To locate and load computer instructions or data from storage. *See also:* **move; store.**

fifth generation language (5GL). A computer language that incorporates the concepts of knowledge-based systems, expert systems, inference engines, and natural language processing. *Contrast with:* **assembly language; fourth generation language; high order language; machine language.** *Note:* Specific languages are defined in P610.13 [17].

figurative constant. A data name that is reserved for a specific constant in a programming language. For example, the data name THREE may be reserved to represent the value 3. *See also:* **literal.**

file. A set of related records treated as a unit. For example, in stock control, a file could consist of a set of invoice records.

finite state machine. A computational model consisting of a finite number of states and transitions between those states, possibly with accompanying actions.

firmware. The combination of a hardware device and computer instructions and data that reside as read-only software on that device. *Notes:* (1) This term is sometimes used to refer only to the hardware device or only to the computer instructions or data, but these meanings are deprecated. (2) The confusion surrounding this term has led some to suggest that it be avoided altogether.

first generation language (1GL). *See:* **machine language.**

flag. A variable that is set to a prescribed state, often "true" or "false," based on the results of a process or the occurrence of a specified condition. *See also:* **indicator; semaphore.**

flexibility. The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed. *Syn:* **adaptability.** *See also:* **extendability; maintainability.**

flow diagram. *See:* **flowchart.**

flow of control. *See:* **control flow.**

flowchart (flow chart). A control flow diagram in which suitably annotated geometrical figures are used to represent operations, data, or equipment, and arrows are used to indicate the sequential flow from one to another. *Syn:* **flow diagram.** *See also:* **block diagram; box diagram; bubble chart; graph; input-process-output chart; structure chart.**

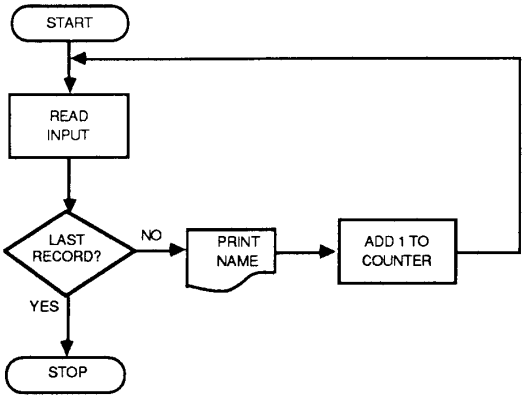

Fig 10
Flowchart

flowcharter. A software tool that accepts as input a design or code representation of a program and produces as output a flowchart of the program.

foreground. In job scheduling, the computing environment in which highpriority processes or those requiring user interaction are executed. *Contrast with:* **background.** *See also:* **foreground processing.**

foreground processing. The execution of a high-priority process while lower-priority processes await the availability of computer resources, or the execution of processes that require user interaction. *Contrast with:* **background processing.**

form, fit, and function. In configuration management, that configuration comprising the physical and functional characteristics of an item as an entity, but not including any characteristics of the elements making up the item. *See also:* **configuration identification.**

formal language. A language whose rules are explicitly established prior to its use. Examples include programming languages and mathematical languages. *Contrast with:* **natural language.**

formal parameter. A variable used in a software module to represent data or program elements that are to be passed to the module by a calling module. *Contrast with:* **argument (3).**

formal qualification review (FQR). The test, inspection, or analytical process by which a group of configuration items comprising a system are verified to have met specific contractual performance requirements. *Contrast with:* **code review; design review; requirements review; test readiness review.**

formal specification. (1) A specification written and approved in accordance with established standards.
(2) A specification written in a formal notation, often for use in proof of correctness.

formal testing. Testing conducted in accordance with test plans and procedures that have been reviewed and approved by a customer, user, or designated level of management. *Contrast with:* **informal testing.**

forward recovery. (1) The reconstruction of a file to a given state by updating an earlier version, using data recorded in a chronological record of changes made to the file.
(2) A type of recovery in which a system, program, database, or other system resource is restored to a new, not previously occupied state in which it can perform required functions.
Contrast with: **backward recovery.**

four-address instruction. A computer instruction that contains four address fields. For example, an instruction to add the contents of locations A, B, and C, and place the result in location D. *Contrast with:* **one-address instruction; two-address instruction; three-address instruction; zero-address instruction.**

four-plus-one address instruction. A computer instruction that contains five address fields, the fifth containing the address of the instruction to be executed next. For example, an instruction to add the contents of locations A, B, and C, place the results in location D, then execute the instruction at location E. *Contrast with:* **one-plus-one address instruction; two-plus-one address instruction; three-plus-one address instruction.**

fourth generation language (4GL). A computer language designed to improve the productivity achieved by high order (third generation) languages and, often, to make computing power available to non-programmers. Features typically include an integrated database management system, query language, report generator, and screen definition facility. Additional features may include a graphics generator, decision support function, financial modeling, spreadsheet capability, and statistical analysis functions. *Contrast with:* **machine language; assembly language; high order language; fifth generation language.** *Note:* Specific languages are defined in P610.13 [17].

FQR. Acronym for **formal qualification review.**

function. (1) A defined objective or characteristic action of a system or component. For example, a system may have inventory control as its primary function. *See also:* **functional requirement; functional specification; functional testing.**
(2) A software module that performs a specific action, is invoked by the appearance of its name in an expression, may receive input values, and returns a single value. *See also:* **subroutine.**

function field. *See:* **operation field.**

functional baseline. In configuration management, the initial approved technical documentation for a configuration item. *Contrast with:* **allocated baseline; developmental configuration; product baseline.** *See also:* **functional configuration item.**

functional cohesion. A type of cohesion in which the tasks performed by a software module all contribute to the performance of a single function. *Contrast with:* **coincidental cohesion; communicational cohesion; logical cohesion; procedural cohesion; sequential cohesion; temporal cohesion.**

functional configuration audit (FCA). An audit conducted to verify that the development of a configuration item has been completed satisfactorily, that the item has achieved the performance and functional characteristics specified in the functional or allocated configuration identification, and that its operational and support documents are complete and satisfactory. *See also:* **configuration management; physical configuration audit.**

functional configuration identification. In configuration management, the current approved technical documentation for a configuration item. It prescribes all necessary functional characteristics, the tests required to demonstrate achievement of specified functional characteristics, the necessary interface characteristics with associated configuration items, the configuration item's key functional characteristics and its key lower level configuration items, if any, and design constraints. *Contrast with:* **allocated configuration identification; product configuration identification.** *See also:* **functional baseline.**

functional decomposition. A type of modular decomposition in which a system is broken down into components that correspond to system functions and subfunctions. *See also:* **hierarchical decomposition; stepwise refinement.**

functional design. (1) The process of defining the working relationships among the components of a system. *See also:* **architectural design.**
(2) The result of the process in (1).

functional language. A programming language used to express programs as a sequence of functions and function calls. Examples include LISP.

functional requirement. A requirement that specifies a function that a system or system component must be able to perform. *Contrast with:* **design requirement; implementation requirement; interface requirement; performance requirement; physical requirement.**

functional specification. A document that specifies the functions that a system or component must perform. Often part of a requirements specification.

functional testing. (1) Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. *Syn:* **black-box testing.** *Contrast with:* **structural testing.**
(2) Testing conducted to evaluate the compliance of a system or component with specified functional requirements. *See also:* **performance testing.**

garbage collection. In computer resource management, a synonym for memory compaction (1).

generality. The degree to which a system or component performs a broad range of functions. *See also:* **reusability.**

generated address. An address that has been calculated during the execution of a computer program. *Syn:* **synthetic address.** *See also:* **absolute address; effective address; relative address; indirect address.**

generic program unit. A software module that is defined in a general manner and that requires substitution of specific data, instructions, or both in order to be used in a computer program. *See also:* **instantiation.**

glass box. (1) A system or component whose internal contents or implementation are known. *Syn:* **white box.** *Contrast with:* **black box.**
(2) Pertaining to an approach that treats a system or component as in (1).

glass-box testing. *See:* **structural testing.**

global compaction. In microprogramming, compaction in which microoperations may be moved beyond the boundaries of the single entry, single exit sequential blocks in which they occur. *Contrast with:* **local compaction.**

global data. Data that can be accessed by two or more non-nested modules of computer program without being explicitly passed as parameters between the modules. *Syn:* **common data.** *Contrast with:* **local data.**

global variable. A variable that can be accessed by two or more non-nested modules of a computer program without being explicitly passed as a parameter between the modules. *Contrast with:* **local variable.**

go to. A computer program statement that causes a jump. *Contrast with:* **call; case; if-then-else.** *See also:* **branch.**

graph. (1) A diagram that represents the variation of a variable in comparison with that of one or more other variables; for example, a graph showing a bathtub curve.

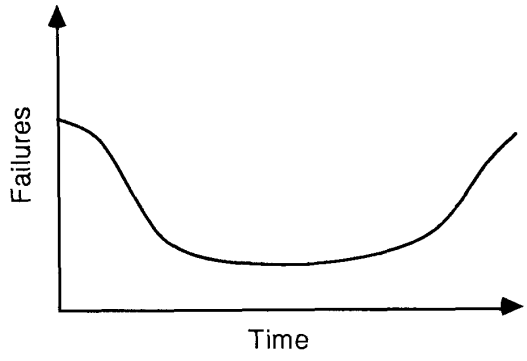

Fig 11
Graph (1)

(2) A diagram or other representation consisting of a finite set of nodes and internode connections called edges or arcs. *See also:* **block diagram; box diagram; bubble chart; directed graph; flowchart; input-process-output chart; structure chart.**

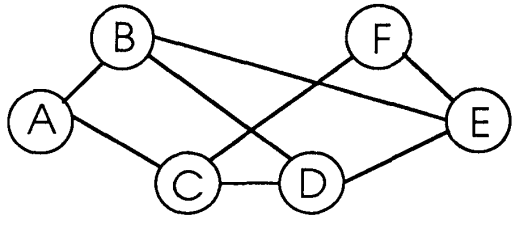

Fig 12
Graph (2)

Grosch's law. A guideline formulated by H. R. J. Grosch, stating that the computing power of a computer increases proportionally to the square of the cost of the computer. *See also:* **computer performance evaluation.**

halt. (1) Most commonly, a synonym for **stop.**
(2) Less commonly, a synonym for **pause.**

hard failure. A failure that results in complete shutdown of a system. *Contrast with:* **soft failure.**

hardware. Physical equipment used to process, store, or transmit computer programs or data. *Contrast with:* **software.**

hardware configuration item (HWCI). An aggregation of hardware that is designated for configuration management and treated as a single entity in the configuration management process. *Contrast with:* **computer software configuration item.** *See also:* **configuration item.**

hardware design language (HDL). A language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document a hardware design. *See also:* **program design language.**

hardware monitor. (1) A device that measures or records specified events or characteristics of a computer system; for example, a device that counts the occurrences of various electrical events or measures the time between such events.
(2) A software tool that records or analyzes hardware events during the execution of a computer program.
See also: **monitor; software monitor.**

HDL. Acronym for **hardware design language.** *See:* **design language.**

header. (1) A block of comments placed at the beginning of a computer program or routine.
(2) Identification or control information placed at the beginning of a file or message. *Contrast with:* **trailer.**

hierarchical decomposition. A type of modular decomposition in which a system is broken down into a hierarchy of components through a series of top-down refinements. *See also:* **functional decomposition; stepwise refinement.**

hierarchical input-process-output (HIPO). *See:* **input-process-output.**

hierarchical modeling. A technique used in computer performance evaluation, in which a computer system is represented as a hierarchy of subsystems, the subsystems are analyzed to determine their performance characteristics, and the results are used to evaluate the performance of the overall system.

hierarchy. A structure in which components are ranked into levels of subordination; each component has zero, one, or more subordinates; and no component has more than one superordinate component. *See also:* **hierarchical decomposition; hierarchical modeling.**

hierarchy chart. *See:* **structure chart.**

high level language. *See:* **high order language.**

high order language (HOL). A programming language that requires little knowledge of the computer on which a program will run, can be translated into several different machine languages, allows symbolic naming of operations and addresses, provides features designed to facilitate expression of data structures and program logic, and usually results in several machine instructions for each program statement. Examples include Ada, COBOL, FORTRAN, ALGOL, PASCAL. *Syn:* **high level language; higher order language; third generation language.** *Contrast with:* **assembly language; fifth generation language; fourth generation language; machine language.** *Note:* Specific languages are defined in P610.13 [17].

higher order language. *See:* **high order language.**

HLL. Acronym for **high level language.** *See:* **high order language.**

HMI. Acronym for **human-machine interface.** *See:* **user interface.**

HOL. Acronym for **high order language.**

homogeneous redundancy. In fault tolerance, realization of the same function with identical means, for example, use of two identical processors. *Contrast with:* **diversity.**

horizontal microinstruction. A microinstruction that specifies a set of simultaneous operations needed to carry out a given machine language instruction. *Note:* Horizontal microinstructions are relatively long, often 64 bits or more, and are called "horizontal" because the set of simultaneous

operations that they specify are written on a single line, rather than being listed sequentially down the page. *Contrast with:* **diagonal microinstruction; vertical microinstruction.**

host machine. (1) A computer used to develop software intended for another computer. *Contrast with:* **target machine (1).**
(2) A computer used to emulate another computer. *Contrast with:* **target machine (2).**
(3) The computer on which a program or file is installed.
(4) In a computer network, a computer that provides processing capabilities to users of the network.

housekeeping operation. A computer operation that establishes or reestablishes a set of initial conditions to facilitate the execution of a computer program; for example, initializing storage areas, clearing flags, rewinding tapes, opening and closing files. *Syn:* **overhead operation.**

human-machine interface (HMI). *See:* **user interface.**

HWCI. Acronym for **hardware configuration item.**

hybrid coupling. A type of coupling in which different subsets of the range of values that a data item can assume are used for different and unrelated purposes in different software module. *Contrast with:* **common-environment coupling; content coupling; control coupling; data coupling; pathological coupling.**

identifier. The name, address, label, or distinguishing index of an object in a computer program.

idle. Pertaining to a system or component that is operational and in service, but not in use. *See also:* **busy; down; up.**

idle time. The period of time during which a system or component is operational and in service, but not in use. *Syn:* **standby time.** *See also:* **busy time; down time; set-up time; up time.**

if-then-else. A single-entry, single-exit two-way branch that defines a condition, specifies the processing to be performed if the condition is met and, optionally, if it is not, and returns control in both instances to the statement immediately following the overall construct. *Contrast with:* **case; jump; go to.** *See also:* **dyadic selective construct; monadic selective construct.**

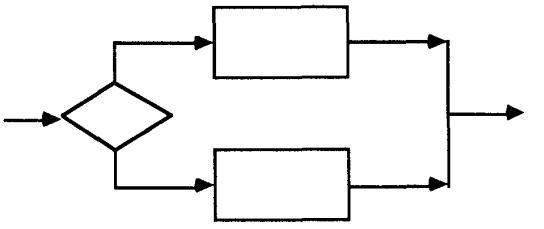

Fig 13
If-Then-Else Construct

immediate address.* *See:* **immediate data.**
* Deprecated.

immediate control. *See:* **bit steering.**

immediate data. Data contained in the address field of a computer instruction. *Contrast with:* **direct address; indirect address; *n*-level address.** *See also:* **immediate instruction.**

immediate instruction. A computer instruction whose address fields contain the values of the operands rather than the operands' addresses. *Contrast with:* **direct instruction; indirect instruction.** *See also:* **absolute instruction; effective instruction; immediate data.**

imperative construct. A sequence of one or more steps not involving branching or iteration.

imperative statement. *See:* **instruction.**

implementation. (1) The process of translating a design into hardware components, software components, or both. *See also:* **coding.**
(2) The result of the process in (1).

implementation phase. The period of time in the software life cycle during which a software product is created from design documentation and debugged.

implementation requirement. A requirement that specifies or constrains the coding or construction of a system or system component. *Contrast with:* **design requirement; functional requirement; interface requirement; performance requirement; physical requirement.**

implied addressing. A method of addressing in which the operation field of an computer instruction implies the address of the operands. For example, if a computer has only one accumulator, an instruction that refers to the accumulator needs no address information describing it. Types include one-ahead addressing, repetitive addressing. *See also:* **direct address; indirect address; relative address.**

incident. (IEEE Std 1008-1987 [10]) *See:* **software test incident.**

incipient failure. A failure that is about to occur.

incremental compiler. A compiler that completes as much of the translation of each source statement as possible during the input or scanning of the source statement. Typically used for on-line computer program development and checkout. *Syn:* **conversational compiler; interactive compiler; on-line compiler.**

incremental development. A software development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product. *Contrast with:* **waterfall model.** *See also:* **data structure-centered design; input-process-output; modular decomposition; object-oriented design; rapid prototyping; spiral model; stepwise refinement; structured design; transaction analysis; transform analysis.**

independent verification and validation (IV&V). Verification and validation performed by an organization that is technically, managerially, and financially independent of the development organization.

indexed address. An address that must be added to the contents of an index register to obtain the address of the storage location to be accessed. *See also:* **offset (2); relative address; self-relative address.**

indicator. A device or variable that can be set to a prescribed state based on the results of a process or the occurrence of a specified condition. For example, a flag or semaphore.

indigenous error. A computer program error that has not been purposely inserted as part of an error-seeding process.

indirect address. An address that identifies the storage location of another address. The designated storage location may contain the address of the desired operand or another indirect address; the chain of addresses eventually leads to the operand. *Syn:* **multilevel address.** *Contrast with:* **direct address; immediate data.** *See also:* **indirect instruction; *n*-level address.**

indirect instruction. A computer instruction that contains indirect addresses for its operands. *Contrast with:* **direct instruction; immediate instruction.** *See also:* **absolute instruction; effective instruction.**

inductive assertion method. A proof of correctness technique in which assertions are written describing program inputs, outputs, and intermediate conditions, a set of theorems is developed relating satisfaction of the input assertions to satisfaction of the output assertions, and the theorems are proved or disproved using proof by induction.

infant mortality. The set of failures that occur during the early-failure period of a system or component.

informal testing. Testing conducted in accordance with test plans and procedures that have not been reviewed and approved by

a customer, user, or designated level of management. *Contrast with:* **formal testing.**

information hiding. A software development technique in which each module's interfaces reveal as little as possible about the module's inner workings and other modules are prevented from using information about the module that is not in the module's interface specification. *See also:* **encapsulation.**

inherited error. An error carried forward from a previous step in a sequential process.

initial program load. *See:* **bootstrap.**

initial program loader. A bootstrap loader used to load that part of an operating system needed to load the remainder of the operating system.

initialize. To set a variable, register, or other storage location to a starting value. *See also:* **clear; reset.**

inline code. A sequence of computer instructions that is physically contiguous with the instructions that logically precede and follow it.

input. (1) Pertaining to data received from an external source.
(2) Pertaining to a device, process, or channel involved in receiving data from an external source.
(3) To receive data from an external source.
(4) To provide data from an external source.
(5) Loosely, input data.
Contrast with: **output.**

input assertion. A logical expression specifying one or more conditions that program inputs must satisfy in order to be valid. *Contrast with:* **loop assertion; output assertion.** *See also:* **inductive assertion method.**

input-output coupling. *See:* **data coupling.**

input-process-output. A software design technique that consists of identifying the steps involved in each process to be performed and identifying the inputs to and outputs from each step. *Note:* A refinement called hierarchical input-process-output identifies the steps, inputs, and outputs at both general and detailed levels of detail. *See also:* **data structurecentered design; input-process-output chart; modular decomposition; object-oriented design; rapid prototyping; stepwise refinement; structured design; transaction analysis; transform analysis.**

input-process-output (IPO) chart. A diagram of a software system or module, consisting of a rectangle on the left listing inputs, a rectangle in the center listing processing steps, a rectangle on the right listing outputs, and arrows connecting inputs to processing steps and processing steps to outputs. *See also:* **block diagram; box diagram; bubble chart; flowchart; graph; structure chart.**

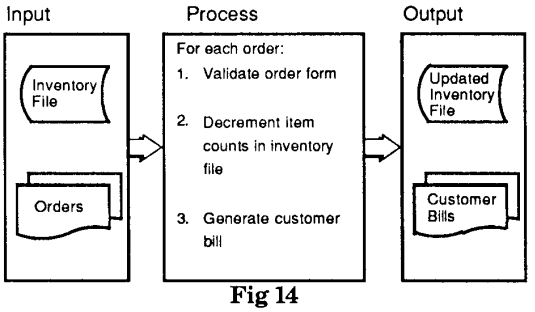

Fig 14
Input-Process-Output Chart

inspection. A static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. Types include code inspection; design inspection.

installation and checkout phase. The period of time in the software life cycle during which a software product is integrated into its operational environment and tested in this environment to ensure that it performs as required.

installation manual. A document that provides the information necessary to install a system or component, set initial parameters, and prepare the system or component for operational use. *See also:* **diagnostic manual; operator manual;**

programmer manual; support manual; user manual.

instantiation. The process of substituting specific data, instructions, or both into a generic program unit to make it usable in a computer program.

instruction. *See:* **computer instruction.**

instruction counter. A register that indicates the location of the next computer instruction to be executed. *Syn:* **program counter.**

instruction cycle. The process of fetching a computer instruction from memory and executing it. *See also:* **instruction time.**

instruction format. The number and arrangement of fields in a computer instruction. *See also:* **address field; address format; operation field.**

instruction length. The number of words, bytes, or bits needed to store a computer instruction. *See also:* **instruction format.**

instruction modifier. A word or part of a word used to alter a computer instruction.

instruction repertoire. *See:* **instruction set.**

instruction set. The complete set of instructions recognized by a given computer or provided by a given programming language. *Syn:* **instruction repertoire.**

instruction time. The time it takes a computer to fetch an instruction from memory and execute it. *See also:* **instruction cycle.**

instrument. In software and system testing, to install or insert devices or instructions into hardware or software to monitor the operation of a system or component.

instrumentation. Devices or instructions installed or inserted into hardware or software to monitor the operation of a system or component.

integer type. A data type whose members can assume only integer values and can be operated on only by integer arithmetic operations, such as addition, subtraction, and multiplication. *Contrast with:* **character type; enumeration type; logical type; real type.**

integration. The process of combining software components, hardware components, or both into an overall system.

integration testing. Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them. *See also:* **component testing; interface testing; system testing; unit testing.**

integrity. The degree to which a system or component prevents unauthorized access to, or modification of, computer programs or data.

interactive. Pertaining to a system or mode of operation in which each user entry causes a response from or action by the system. *Contrast with:* **batch.** *See also:* **conversational; on-line; real time.**

interactive compiler. *See:* **incremental compiler.**

interactive language. A nonprocedural language in which a program is created as a result of interactive dialog between the user and the computer system. The system provides questions, forms, and so on, to aid the user in expressing the results to be achieved. *See also:* **declarative language; rule-based language.**

interface. 1) A shared boundary across which information is passed.
(2) A hardware or software component that connects two or more other components for the purpose of passing information from one to the other.
(3) To connect two or more components for the purpose of passing information from one to the other.
(4) To serve as a connecting or connected component as in (2).

interface control. (1) (IEEE Std 828-1983 [4]) In configuration management, the process of: (a) identifying all functional and physical

characteristics relevant to the interfacing of two or more configuration items provided by one or more organizations, and (b) ensuring that proposed changes to these characteristics are evaluated and approved prior to implementation.
(2) (DOD usage) In configuration management, the administrative and technical procedures and documentation necessary to identify functional and physical characteristics between and within configuration items provided by different developers, and to resolve problems concerning the specified interfaces. *See also:* **configuration control.**

interface requirement. A requirement that specifies an external item with which a system or system component must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction. *Contrast with:* **design requirement; functional requirement; implementation requirement; performance requirement; physical requirement.**

interface specification. A document that specifies the interface characteristics of an existing or planned system or component.

interface testing. Testing conducted to evaluate whether systems or components pass data and control correctly to one another. *See also:* **component testing; integration testing; system testing; unit testing.**

interleave. To alternate the elements of one sequence with the elements of one or more other sequences so that each sequence retains its identity; for example, to alternately perform the steps of two different tasks in order to achieve concurrent operation of the tasks.

intermittent fault. A temporary or unpredictable fault in a component. *See also:* **random failure; transient error.**

interoperability. The ability of two or more systems or components to exchange information and to use the information that has been exchanged. *See also:* **compatibility.**

interpret. To translate and execute each statement or construct of a computer program before translating and executing the next. *Contrast with:* **assemble; compile.**

interpreter. A computer program that translates and executes each statement or construct of a computer program before translating and executing the next. *Contrast with:* **assembler; compiler.**

interpretive code. Computer instructions and data definitions expressed in a form that can be recognized and processed by an interpreter. *Contrast with:* **assembly code; compiler code; machine code.**

interrupt. (1) The suspension of a process to handle an event external to the process. *Syn:* **interruption.** *See also:* **interrupt latency; interrupt mask; interrupt priority; interrupt service routine; priority interrupt.**
(2) To cause the suspension of a process.
(3) Loosely, an interrupt request.

interrupt latency. The delay between a computer system's receipt of an interrupt request and its handling of the request. *See also:* **interrupt priority.**

interrupt mask. A mask used to enable or disable interrupts by retaining or suppressing bits that represent interrupt requests.

interrupt priority. The importance assigned to a given interrupt request. This importance determines whether the request will cause suspension of the current process and, if there are several outstanding interrupt requests, which will be handled first.

interrupt request. A signal or other input requesting that the currently executing process be suspended to permit performance of another process.

interrupt service routine. A routine that responds to interrupt requests by storing the contents of critical registers, performing the processing required by the interrupt request, restoring the register contents, and restarting the interrupted process.

interruption. *See:* **interrupt.**

invariant. An assertion that should always be true for a specified segment or at a specified point of a computer program.

IPO chart. Acronym for **input-process-output chart.**

IPSE. Acronym for **integrated programming support environment.** *See:* **programming support environment.**

iteration. (1) The process of performing a sequence of steps repeatedly. *See also:* **loop; recursion.**
(2) A single execution of the sequence of steps in (1).

iterative construct. *See:* **loop.**

IV&V. Acronym for **independent verification and validation.**

JCL. Acronym for **job control language.**

job. A user-defined unit of work that is to be accomplished by a computer. For example, the compilation, loading, and execution of a computer program. *See also:* **job control language; job step; job stream.**

job control language (JCL). A language used to identify a sequence of jobs, describe their requirements to an operating system, and control their execution.

job function. (IEEE Std 1002-1987 [9]) A group of engineering processes that is identified as a unit for the purposes of work organization, assignment, or evaluation. Examples are design, testing, or configuration management.

job step. A user-defined portion of a job, explicitly identified by a job control statement. A job consists of one or more job steps.

job stream. A sequence of programs or jobs set up so that a computer can proceed from one to the next without the need for operator intervention. *Syn:* **run stream.**

jump. (1) To depart from the implicit or declared order in which computer program statements are being executed. *Syn:* **transfer.**
(2) A program statement that causes a departure as in (1). *Contrast with:* **case; if-then-else.** *See also:* **branch; go to.**
(3) The departure described in (1). *See also:* **conditional jump; unconditional jump.**

kernel. (1) That portion of an operating system that is kept in main memory at all times. *Syn:* **nucleus; resident control program.**
(2) A software module that encapsulates an elementary function or functions of a system. *See also:* **security kernel.**

KOPS. Acronym for kilo-operations per second; that is, thousands of operations per second. A measure of computer processing speed. *See also:* **MFLOPS; MIPS.**

label. (1) A name or identifier assigned to a computer program statement to enable other statements to refer to that statement.
(2) One or more characters, within or attached to a set of data, that identify or describe the data.

language. (1) A systematic means of communicating ideas by the use of conventionalized signs, sounds, gestures, or marks and rules for the formation of admissible expressions.
(2) (IEEE Std 830-1984 [6]) A means of communication, with syntax and semantics, consisting of a set of representations, conventions, and associated rules used to convey information.
See also: **computer language.**

language processor. A computer program that translates, interprets, or performs other tasks required to process statements expressed in a given language. *See also:* **assembler; compiler; interpreter; translator.**

language standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of a language used to describe a requirements specification, a design, or test data.

latency. The time interval between the instant at which an instruction control unit issues a

call for data and the instant at which the transfer of data is started.

lateral compression. In software design, a form of demodularization in which two or more modules that execute one after the other are combined into a single module. *Contrast with:* **downward compression; upward compression.**

leading decision. A loop control that is executed before the loop body. *Contrast with:* **trailing decision.** *See also:* **WHILE.**

library. *See:* **software library.**

licensing standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of an authorization given by an official or a legal authority to an individual or organization to do or own a specific thing.

life cycle. *See:* **software life cycle; system life cycle.**

link. (1) To create a load module from two or more independently translated object modules or load modules by resolving cross-references among them. *See also:* **linkage editor.**
(2) A part of a computer program, often a single instruction or address, that passes control and parameters between separate modules of the program. *Syn:* **linkage.**
(3) To provide a link as in (2).

linkage. *See:* **link (2).**

linkage editor. A computer program that creates a single load module from two or more independently translated object modules or load modules by resolving cross-references among the modules and, possibly, by relocating elements. May be part of a loader. *Syn:* **linker.** *See also:* **linking loader.**

linker. *See:* **linkage editor.**

linking loader. A computer program that reads one or more object modules into main memory in preparation for execution, creates a single load module by resolving cross-references among the separate modules, and, in some cases, adjusts the addresses to reflect the storage locations into which the code has been loaded. *See also:* **absolute loader; relocating loader; linkage editor.**

list. (1) A set of data items, each of which has the same data definition.
(2) To print or otherwise display a set of data items.
Note: IEEE Std 610.5-1990 [2] defines Data Management terms.

list processing language. A programming language designed to facilitate the manipulation of data expressed in the form of lists. Examples are LISP and IPL. *See also:* **algebraic language; algorithmic language; logic programming language.**

listing. An ordered display or printout of data items, program statements, or other information.

literal. In a source program, an explicit representation of the value of an item; for example, the word FAIL in the instruction: If $x = 0$ then print "FAIL". *See also:* **immediate data; figurative constant.**

load. (1) To read machine code into main memory in preparation for execution and, in some cases, to perform address adjustment and linking of modules. *See also:* **loader.**
(2) To copy computer instructions or data from external storage to internal storage or from internal storage to registers. *Contrast with:* **store (2).** *See also:* **fetch; move.**

load-and-go. An operating technique in which there are no stops between the loading and execution phases of a computer program.

load map. A computer-generated list that identifies the location or size of all or selected parts of memory-resident code or data.

load module. A computer program or subprogram in a form suitable for loading into main storage for execution by a computer; usually the output of a linkage editor. *See also:* **object module.**

loaded origin. The address of the initial storage location of a computer program at the time the program is loaded into main memory. *Contrast with:* **assembled origin.** *See also:* **offset (1); starting address.**

loader. (1) A computer program that reads machine code into main memory in preparation for execution and, in some cases, adjusts the addresses and links the modules. Types include absolute loader, linking loader, relocating loader. *See also:* **bootstrap; linkage editor.**
(2) Any program that reads programs or data into main memory.

local compaction. In microprogramming, compaction in which microoperations are not moved beyond the boundaries of the single entry, single exit sequential blocks in which they occur. *Contrast with:* **global compaction.**

local data. Data that can be accessed by only one module or set of nested modules in a computer program. *Contrast with:* **global data.**

local variable. A variable that can be accessed by only one module or set of nested modules in a computer program. *Contrast with:* **global variable.**

lockout. A computer resource allocation technique in which shared resources (especially data) are protected by permitting access by only one device or process at a time. *See also:* **deadlock; semaphore.**

logic programming language. A programming language used to express programs in terms of control constructs and a restricted predicate calculus; for example, PROLOG. *See also:* **algebraic language; algorithmic language; list processing language.**

logical cohesion. A type of cohesion in which the tasks performed by a software module perform logically similar functions; for example, processing of different types of input data. *Contrast with:* **coincidental cohesion; communicational cohesion; functional cohesion; procedural cohesion; sequential cohesion; temporal cohesion.**

logical trace. An execution trace that records only branch or jump instructions. *See also:* **execution trace; retrospective trace; subroutine trace; symbolic trace; variable trace.**

logical type. A data type whose members can assume only logical values (usually TRUE and FALSE) and can be operated on only by logical operators, such as AND, OR, and NOT. *Contrast with:* **character type; enumeration type; integer type; real type.**

loop. (1) A sequence of computer program statements that is executed repeatedly until a given condition is met or while a given condition is true. *Syn:* **iterative construct.** *See also:* **loop body; loop control; UNTIL; WHILE.**
(2) To execute a sequence of computer program statements as in (1).

loop assertion. A logical expression specifying one or more conditions that must be met each time a particular point in a program loop is executed. *Syn:* **loop invariant.** *Contrast with:* **input assertion; output assertion.** *See also:* **inductive assertion method.**

loop body. The part of a loop that accomplishes the loop's primary purpose. *Contrast with:* **loop control.**

loop control. The part of a loop that determines whether to exit from the loop. *Contrast with:* **loop body.** *See also:* **leading decision; trailing decision.**

loop-control variable. A program variable used to determine whether to exit from a loop.

loop invariant. *See:* **loop assertion.**

loopback testing. Testing in which signals or data from a test device are input to a system or component, and results are returned to the test device for measurement or comparison.

low level language. *See:* **assembly language.**

machine address.* *See:* **absolute address.**
*Deprecated.

machine code. Computer instructions and data definitions expressed in a form that can be recognized by the processing unit of a computer. *Contrast with:* **assembly code; compiler code; interpretive code.**

machine dependent. Pertaining to software that relies on features unique to a particular type of computer and therefore executes only on computers of that type. *Contrast with:* **machine independent.**

machine independent. Pertaining to software that does not rely on features unique to a particular type of computer, and therefore executes on computers of more than one type. *Contrast with:* **machine dependent.** *See also:* **portability.**

machine language. A language that can be recognized by the processing unit of a computer. Such a language usually consists of patterns of 1s and 0s, with no symbolic naming of operations or addresses. *Syn:* **first-generation language; machine-oriented language.** *Contrast with:* **assembly language; fifth-generation language; fourth generation language; high order language; symbolic language.**

machine-oriented language. *See:* **machine language.**

machine readable. Pertaining to data in a form that can be automatically input to a computer; for example, data encoded on a diskette.

macro. In software engineering a predefined sequence of computer instructions that is inserted into a program, usually during assembly or compilation, at each place that its corresponding macroinstruction appears in the program. *Syn:* **macro definition.** *See also:* **macroinstruction; macrogenerator; open subroutine.**

macro definition. *See:* **macro.**

macro generating program. *See:* **macrogenerator.**

macro library. A collection of macros available for use by a macrogenerator. *See also:* **system library.**

macroassembler. An assembler that includes, or performs the functions of, a macrogenerator.

macrogenerator. A routine, often part of an assembler or compiler, that replaces each macroinstruction in a source program with the predefined sequence of instructions that the macroinstruction represents. *Syn:* **macro generating program.**

macroinstruction. A source code instruction that is replaced by a predefined sequence of source instructions, usually in the same language as the rest of the program and usually during assembly or compilation. *See also:* **macro; macrogenerator.**

macroprocessor. A routine or set of routines provided in some assemblers and compilers to support the definition and use of macros.

macroprogramming. Computer programming using macros and macroinstructions.

main program. A software component that is called by the operating system of a computer and that usually calls other software components. *See also:* **routine; subprogram.**

maintainability. (1) The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment. *See also:* **extendability; flexibility.**
(2) The ease with which a hardware system or component can be retained in, or restored to, a state in which it can perform its required functions.

maintenance. (1) The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment. *Syn:* **software maintenance.** *See also:* **adaptive maintenance; corrective maintenance; perfective maintenance.**
(2) The process of retaining a hardware system or component in, or restoring it to, a state in which it can perform its required functions. *See also:* **preventive maintenance.**

maintenance manual. *See:* **support manual.**

man-machine interface (MMI). *See:* **user interface.**

manufacture. In software engineering, the process of copying software to disks, chips, or other devices for distribution to customers or users.

manufacturing phase. (IEEE Std 1002-1987 [9]) The period of time in the software life cycle during which the basic version of a software product is adapted to a specified set of operational environments and is distributed to a customer base.

map program. A software tool, often part of a compiler or assembler, that generates a load map.

mask. A pattern of bits or characters designed to be logically combined with an unknown data item to retain or suppress portions of the data item; for example, the bit string "00000011" when logically ANDed with an eight-bit data item, gives a result that retains the last two bits of the data item and has zero in all the other bit positions. *See also:* **interrupt mask.**

master library. A software library containing master copies of software and documentation from which working copies can be made for distribution and use. *Contrast with:* **production library; software development library; software repository; system library.**

master state. *See:* **supervisor state.**

mean time between failures (MTBF). The expected or observed time between consecutive failures in a system or component. *See also:* **up time.**

mean time to repair (MTTR). The expected or observed time required to repair a system or component and return it to normal operations. *See also:* **down time.**

measurement standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of evaluating a process of product.

memory capacity. The maximum number of items that can be held in a given computer memory; usually measured in words or bytes. *See also:* **channel capacity; storage capacity.**

memory compaction. (1) A storage allocation technique in which the contents of all allocated storage areas are moved to the beginning of the storage space and the remaining storage blocks are combined into a single block. *Syn:* **garbage collection.** (2) A storage allocation technique in which contiguous blocks of nonallocated storage are combined to form single blocks.

memory dump. A display of the contents of all or part of a computer's internal storage, usually in binary, octal, or hexadecimal form. *See also:* **change dump; dynamic dump; port mortem dump; selective dump; snapshot dump; static dump.**

memory map. A diagram that shows where programs and data are stored in a computer's memory.

menu by-pass. In a menu-driven system, a feature that permits advanced users to perform functions in a command-driven mode without selecting options from the menus.

menu-driven. Pertaining to a system or mode of operation in which the user directs the system through menu selections. *See also:* **menu by-pass.** *Contrast with:* **command-driven.**

metacompiler. *See:* **compiler generator.**

metalanguage. A language used to specify some or all aspects of a language; for example, Backus-Naur form. *See also:* **stratified language; unstratified language.**

method standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of the orderly process or procedure used in the engineering of a product or performing a service.

metric. A quantitative measure of the degree to which a system, component, or process

possesses a given attribute. *See also:* **quality metric.**

MFLOPS. Acronym for millions of floating point operations per second. A measure of computer processing speed. *See also:* **KOPS; MIPS.**

microarchitecture. The microword definition, data flow, timing constraints, and precedence constraints that characterize a given microprogrammed computer.

microcode. A collection of microinstructions, comprising part of, all of, or a set of microprograms.

microcode assembler. A computer program that translates microprograms from symbolic form to binary form.

microinstruction. In microprogramming, an instruction that specifies one or more of the basic operations needed to carry out a machine language instruction. Types include diagonal microinstruction; horizontal microinstruction; vertical microinstruction. *See also:* **microcode; microoperation; microprogram.**

microoperation. In microprogramming, one of the basic operations needed to carry out a machine language instruction. *See also:* **microinstruction.**

microprogram. A sequence of instructions, called microinstructions, specifying the basic operations needed to carry out a machine language instruction.

microprogrammable computer. A microprogrammed computer in which microprograms can be created or altered by the user.

microprogrammed computer. A computer in which machine language instructions are implemented by microprograms rather than by hard-wired logic. *Note:* A microprogrammed computer may or may not be a microcomputer; the concepts are not related despite the similarity of the terms. *See also:* **microarchitecture; microprogrammable computer.**

microprogramming. The process of designing and implementing the control logic of a computer by identifying the basic operations needed to carry out each machine language instruction and representing these operations as sequences of instructions in a special memory called control store. This method is an alternative to hard wiring the control signals necessary to carry out each machine language instruction. Techniques include bit steering, compaction, residual control, single-level encoding, two-level encoding. *See also:* **microcode; microinstruction; microprogram.**

microword. An addressable element in the control store of a microprogrammed computer.

minimum delay programming. A programming technique in which storage locations for computer instructions and data are chosen so that access time is minimized.

MIPS. Acronym for million instructions per second. A measure of computer processing speed. *See also:* **KOPS; MFLOPS.**

mistake. A human action that produces an incorrect result. *Note:* The fault tolerance discipline distinguishes between the human action (a mistake), its manifestation (a hardware or software fault), the result of the fault (a failure), and the amount by which the result is incorrect (the error).

mixed mode. Pertaining to an expression that contains two or more different data types. For example, $Y := X + N$, where X and Y are floating point variables and N is an integer variable. *Syn:* **mixed type.**

mixed type. *See:* **mixed mode.**

MMI. Acronym for **man-machine interface.** *See:* **user interface.**

modular. Composed of discrete parts. *See also:* **modular decomposition; modular programming.**

modular decomposition. The process of breaking a system into components to facilitate design and development; an element of

modular programming. *Syn:* **modularization.** *See also:* **cohesion; coupling; demodularization; factoring; functional decomposition; hierarchical decomposition; packaging.**

modular programming. A software development technique in which software is developed as a collection of modules. *See also:* **data structure-centered design; input-process-output; modular decomposition; object-oriented design; rapid prototyping; stepwise refinement; structured design; transaction analysis; transform analysis.**

modularity. The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components. *See also:* **cohesion; coupling.**

modularization. *See:* **modular decomposition.**

module. (1) A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to, or output from, an assembler, compiler, linkage editor, or executive routine.
(2) A logically separable part of a program.
Note: The terms "module," "component," and "unit" are often used interchangeably or defined to be sub-elements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized.

module strength. *See:* **cohesion.**

module testing. *See:* **component testing.**

monadic selective construct. An if-then-else construct in which processing is specified for only one outcome of the branch, the other outcome resulting in skipping this processing. *Contrast with:* **dyadic selective construct.**

monitor. A software tool or hardware device that operates concurrently with a system or component and supervises, records, analyzes, or verifies the operation of the system or component. *Syn:* **execution monitor.** *See also:* **hardware monitor; software monitor.**

move. (1) To read data from a source, altering the contents of the source location, and to write the same data elsewhere in a physical form that may differ from that of the source. For example, to move data from one file to another. *Contrast with:* **copy.**
(2) Sometimes, a synonym for **copy.**
See also: **fetch; load; store.**

MTBF. Acronym for **mean time between failures.**

MTTR. Acronym for **mean time to repair.**

multiaddress instruction. A computer instruction that contains more than one address field. *Syn:* **multiple-address instruction.** *Contrast with:* **one-address instruction.**

multilevel address. *See:* **indirect address.**

multilevel storage. *See:* **virtual storage.**

multiple-address instruction. *See:* **multiaddress instructions.**

multiple exclusive selective construct. *See:* **case.**

multiple inclusive selective construct. A special instance of the case construct in which two or more different values of the control expression result in the same processing. For example, values 1 and 2 cause one branch, 3 and 4 cause another, and so on.

multiprocessing. A mode of operation in which two or more processes are executed concurrently by separate processing units that have access (usually) to a common main storage. *Contrast with:* **multiprogramming.** *See also:* **multitasking; time sharing.**

multiprogramming. A mode of operation in which two or more computer programs are executed in an interleaved manner by a single processing unit. *Contrast with:* **multiprocessing.** *See also:* **multitasking; time sharing.**

multitasking. A mode of operation in which two or more tasks are executed in an

interleaved manner. *See also:* **multiprocessing; multiprogramming; time sharing.**

mutation. *See:* **program mutation.**

mutation testing. A testing methodology in which two or more program mutations are executed using the same test cases to evaluate the ability of the test cases to detect differences in the mutations.

***n*-address instruction.** A computer instruction that contains n address fields, where *n* may be any non-negative integer. *See also:* **one-address instruction; two-address instruction; etc.** *Contrast with:* ***n*-plus-one address instruction.**

***n*-level address.** An indirect address that specifies the first of a chain of *n* storage locations, the first *n*-1 of which contain the address of the next location in the chain and the last of which contains the desired operand. For example, a two-level address. *Contrast with:* **direct address; immediate data.**

***n*-plus-one address instruction.** A computer instruction that contains *n*+1 address fields, the last containing the address of the instruction to be executed next. *See also:* **one-plus-one address instruction; two-plus-one address instruction; etc.** *Contrast with:* ***n*-address instruction.**

nanocode. A collection of nanoinstructions.

nanoinstruction. In a two-level implementation of microprogramming, an instruction that specifies one or more of the basic operations needed to carry out a microinstruction.

nanostore. In a two-level implementation of microprogramming, a secondary control store in which nanoinstructions reside.

Nassi-Shneiderman chart. *See:* **box diagram.**

natural language. A language whose rules are based on usage rather than being preestablished prior to the language's use. Examples include German and English. *Contrast with:* **formal language.**

nest. To incorporate a computer program construct into another construct of the same kind. For example, to nest one subroutine, block, or loop within another; to nest one data structure within another.

no-op. Abbreviation for **no-operation.**

no-operation. A computer operation whose execution has no effect except to advance the instruction counter to the next instruction. Used to reserve space in a program or, if executed repeatedly, to wait for a given event. Often abbreviated no-op. *Syn:* **do-nothing operation.**

node. (1) In a diagram, a point, circle, or other geometric figure used to represent a state, event, or other item of interest. *See also:* graph (2).
(2) *Note:* The meaning of this term in the context of computer networks is covered in P610.7-1990 [14].

nomenclature standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of a system or set of names, or designations, or symbols.

nondestructive read. A read operation that does not erase the data in the accessed location. *Contrast with:* **destructive read.**

nonprocedural language. A language in which the user states what is to be achieved without having to state specific instructions that the computer must execute in a given sequence. *Contrast with:* **procedural language.** *See also:* **declarative language; interactive language; rule-based language.**

NOR. (1) In configuration management, an acronym for **notice of revision.**
(2) *Note:* The meaning of this term as a logical operator is given in IEEE Std 610.1/1084-1986 [1, 11].

notation standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of formal interfaces within a profession.

notice of revision (NOR). A form used in configuration management to propose revisions to a drawing or list, and, after

approval, to notify users that the drawing or list has been, or will be, revised accordingly. *See also:* **configuration control; engineering change; specification change notice.**

nucleus. *See:* **kernel (1).**

object. (1) Pertaining to the outcome of an assembly or compilation process. *See also:* **object code; object module; object program.**
(2) A program constant or variable.
(3) An encapsulation of data and services that manipulate that data. *See also:* **object-oriented design.**

object code. Computer instructions and data definitions in a form output by an assembler or compiler. An object program is made up of object code. *Contrast with:* **source code.**

object language. *See:* **target language.**

object module. A computer program or subprogram that is the output of an assembler or compiler. *See also:* **load module; object program.**

object-oriented design. A software development technique in which a system or component is expressed in terms of objects and connections between those objects. *See also:* **data structure-centered design; input-process-output; modular decomposition; rapid prototyping; stepwise refinement; structured design; transaction analysis; transform analysis.**

object-oriented language. A programming language that allows the user to express a program in terms of objects and messages between those objects. Examples include Smalltalk and LOGO.

object program. A computer program that is the output of an assembler or compiler. *Contrast with:* **source program.** *Syn:* **target program.** *See also:* **object module.**

occupational title standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of the general areas of work or profession.

off-line. Pertaining to a device or process that is not under the direct control of the central processing unit of a computer. *Contrast with:* **on-line (2).**

offset. (1) The difference between the loaded origin and the assembled origin of a computer program. *Syn:* **relocation factor.**
(2) A number that must be added to a relative address to determine the address of the storage location to be accessed. This number may be the difference defined in (1) or another number defined in the program. *See also:* **base address; indexed address; relative address; self-relative address.**

on-line. (1) Pertaining to a system or mode of operation in which input data enter the computer directly from the point of origin or output data are transmitted directly to the point where they are used. For example, an airline reservation system. *Contrast with:* **batch.** *See also:* **conversational; interactive; real time.**
(2) Pertaining to a device or process that is under the direct control of the central processing unit of a computer.
Contrast with: **off-line.**

on-line compiler. *See:* **incremental compiler.**

one-address instruction. A computer instruction that contains one address field. For example, an instruction to load the contents of location A. *Syn:* **single-address instruction; single-operand instruction.** *Contrast with:* **multiaddress instruction; two-address instruction; three-address instruction; four-address instruction; zero-address instruction.**

one-ahead addressing. A method of implied addressing in which the operands for a computer instruction are understood to be in the storage locations following the locations of the operands used for the last instruction executed. *Contrast with:* **repetitive addressing.**

one-level address. *See:* **direct address.**

one-plus-one address instruction. A computer instruction that contains two address fields, the second containing the address of the

instruction to be executed next. For example, an instruction to load the contents of location A, then execute the instruction at location B. *Contrast with:* **two-plus-one address instruction; three-plus-one address instruction; four-plus-one address instruction.**

op code (opcode). *See:* **operation code.**

open subroutine. A subroutine that is copied into a computer program at each place that it is called. *Syn:* **direct insert subroutine.** *Contrast with:* **closed subroutine.** *See also:* **inline code; macro.**

operand. A variable, constant, or function upon which an operation is to be performed. For example, in the expression $A = B + 3$, B and 3 are the operands.

operating system. A collection of software, firmware, and hardware elements that controls the execution of computer programs and provides such services as computer resource allocation, job control, input/output control, and file management in a computer system.

operation. (1) In computer mathematics, the action specified by an operator on one or more operands. For example, in the expression $A = B + 3$, the process of adding B to 3 to obtain A.
(2) In programming, a defined action that can be performed by a computer system; for example, addition, comparison, branching. *Note:* Unlike the mathematical meaning, such an operation may not involve an operator or operands; for example, the operation Halt.
(3) The process of running a computer system in its intended environment to perform its intended functions.

operation and maintenance phase. The period of time in the software life cycle during which a software product is employed in its operational environment, monitored for satisfactory performance, and modified as necessary to correct problems or to respond to changing requirements.

operation code. A character or set of characters that specifies a computer operation; for example, the code BNZ to designate the operation "branch if not zero." *Syn:* **op code.**

operation exception. An exception that occurs when a program encounters an invalid operation code. *See also:* **addressing exception; data exception; overflow exception; protection exception; underflow exception.**

operation field. The field of a computer instruction that specifies the operation to be performed. *Syn:* **function field; operation part.** *Contrast with:* **address field.**

operation part. *See:* **operation field.**

operational. (1) Pertaining to a system or component that is ready for use in its intended environment.
(2) Pertaining to a system or component that is installed in its intended environment.
(3) Pertaining to the environment in which a system or component is intended to be used.

operational testing. Testing conducted to evaluate a system or component in its operational environment. *Contrast with:* **development testing.** *See also:* **acceptance testing; qualification testing.**

operator. (1) A mathematical or logical symbol that represents an action to be performed in an operation. For example, in the expression $A = B + 3$, + is the operator, representing addition.
(2) A person who operates a computer system.

operator field. *See:* **operation field.**

operator manual. A document that provides the information necessary to initiate and operate a system or component. Typically described are procedures for preparation, operation, monitoring, and recovery. *Note:* An operator manual is distinguished from a user manual when a distinction is made between those who operate a computer system (mounting tapes, etc.) and those who use the system for its intended purpose. *See also:* **diagnostic manual; installation manual; programmer manual; support manual; user manual.**

order clash. In software design, a type of structure clash in which a program must deal with two or more data sets that have been sorted in different orders. *See also:* **data structure-centered design.**

origin. The address of the initial storage location assigned to a computer program in main memory. *See also:* **assembled origin; loaded origin.** *Contrast with:* **starting address.**

output. (1) Pertaining to data transmitted to an external destination.
(2) Pertaining to a device, process, or channel involved in transmitting data to an external destination.
(3) To transmit data to an external destination.
(4) Loosely, output data.
Contrast with: **input.**

output assertion. A logical expression specifying one or more conditions that program outputs must satisfy in order for the program to be correct. *Contrast with:* **input assertion; loop assertion.** *See also:* **inductive assertion method.**

overflow exception. An exception that occurs when the result of an arithmetic operation exceeds the size of the storage location designated to receive it. *See also:* **addressing exception; data exception; operation exception; protection exception; underflow exception.**

overhead operation. *See:* **housekeeping operation.**

overhead time. The amount of time a computer system spends performing tasks that do not contribute directly to the progress of any user task; for example, time spent tabulating computer resource usage for billing purposes.

overlay. (1) A storage allocation technique in which computer program segments are loaded from auxiliary storage to main storage when needed, overwriting other segments not currently in use.
(2) A computer program segment that is maintained in auxiliary storage and loaded into main storage when needed, overwriting other segments not currently in use.
(3) To load a computer program segment from auxiliary storage to main storage in such a way that other segments of the program are overwritten.

overlay supervisor. A routine that controls the sequencing and positioning of overlays.

overload. To assign an operator, identifier, or literal more than one meaning, depending upon the data types associated with it at any given time during program execution.

pack. To store data in a compact form in a storage medium, using known characteristics of the data and medium in such a way as to permit recovery of the data. *Contrast with:* **unpack.**

package. A separately compilable software component consisting of related data types, data objects, and subprograms. *See also:* **data abstraction; encapsulation; information hiding.**

packaging. In software development, the assignment of modules to segments to be handled as distinct physical units for execution by a computer.

padding. (1) The technique of filling out a fixed-length block of data with dummy characters, words, or records.
(2) Dummy characters, words, or records used to fill out a fixed-length block of data.

page. (1) A fixed-length segment of data or of a computer program treated as a unit in storage allocation. *See also:* **paging.**
(2) In a virtual storage system, a fixed-length segment of data or of a computer program that has a virtual address and is transferred as a unit between main and auxiliary storage.
(3) A screenful of information on a video display terminal.

page breakage. A portion of main storage that is unused when the last page of data or of a computer program does not fill the entire block of storage allocated to it. *See also:* **paging.**

page frame. A block of main storage having the size of, and used to hold, a page. *See also:* **paging.**

page swapping. The exchange of pages between main storage and auxiliary storage. *See also:* **paging.**

page table. A table that identifies the location of pages in storage and gives significant attributes of those pages. *See also:* **paging.**

page turning. *See:* **paging (3).**

page zero. In the paging method of storage allocation, the first page in a series of pages.

pager. A routine that initiates and controls the transfer of pages between main and auxiliary storage. *See also:* **paging.**

paging. (1) A storage allocation technique in which programs or data are divided into fixed-length blocks called pages, main storage is divided into blocks of the same length called page frames, and pages are stored in page frames, not necessarily contiguously or in logical order. *Syn:* **block allocation.** *Contrast with:* **contiguous allocation.**
(2) A storage allocation technique in which programs or data are divided into fixed-length blocks called pages, main storage is divided into blocks of the same length called page frames, and pages are transferred between main and auxiliary storage as needed. *See also:* **anticipatory paging; demand paging; virtual storage.**
(3) The transfer of pages as in (2). *Syn:* **page turning.**
See also: **page; page breakage; page frame; page swapping; page table; page zero; pager; working set.**

parallel. (1) Pertaining to the simultaneous transfer, occurrence, or processing of the individual parts of a whole, such as the bits of a character, using separate facilities for the various parts. *Contrast with:* **serial (1).**
(2) *See:* **concurrent.**

parallel construct. A program construct consisting of two or more procedures that can occur simultaneously.

parameter. (1) A variable that is given a constant value for a specified application. *See also:* **adaptation parameter.**
(2) A constant, variable, or expression that is used to pass values between software modules. *See also:* **argument; formal parameter.**

parse. To determine the syntactic structure of a language unit by decomposing it into more elementary subunits and establishing the relationships among the subunits. For example, to decompose blocks into statements, statements into expressions, expressions into operators and operands.

parser. A software tool that parses computer programs or other text, often as the first step of assembly, compilation, interpretation, or analysis.

partial correctness. In proof of correctness, a designation indicating that a program's output assertions follow logically from its input assertions and processing steps. *Contrast with:* **total correctness.**

partitioning. (IEEE Std 830-1984 [6]) Decomposition; the separation of the whole into its parts.

pass. A single cycle in the processing of a set of data, usually performing part of an overall process. For example, a pass of an assembler through a source program; a pass of a sort program through a set of data.

pass/fail criteria. (IEEE Std 829-1983 [5]) Decision rules used to determine whether a software item or a software feature passes or fails a test. *See also:* **test criteria.**

patch. (1) A modification made directly to an object program without reassembling or recompiling from the source program.
(2) A modification made to a source program as a last-minute fix or afterthought.
(3) Any modification to a source or object program.
(4) To perform a modification as in (1), (2), or (3).

path. (1) In software engineering, a sequence of instructions that may be performed in the

execution of a computer program.
(2) In file access, a hierarchical sequence of directory and subdirectory names specifying the storage location of a file.

path analysis. Analysis of a computer program to identify all possible paths through the program, to detect incomplete paths, or to discover portions of the program that are not on any path.

path condition. A set of conditions that must be met in order for a particular program path to be executed.

path expression. A logical expression indicating the input conditions that must be met in order for a particular program path to be executed.

path testing. Testing designed to execute all or selected paths through a computer program. *Contrast with:* **branch testing; statement testing.**

pathological coupling. A type of coupling in which one software module affects or depends upon the internal implementation of another. *Contrast with:* **common-environment coupling; content coupling; control coupling; data coupling; hybrid coupling.**

pattern-sensitive fault. *See:* **data-sensitive fault.**

pause. To suspend the execution of a computer program. *Syn:* **halt (2).** *Contrast with:* **stop.**

PCA. Acronym for **physical configuration audit.**

PDL. Acronym for **program design language.**

PDR. Acronym for **preliminary design review.**

perfective maintenance. Software maintenance performed to improve the performance, maintainability, or other attributes of a computer program. *Contrast with:* **adaptive maintenance; corrective maintenance.**

performance. The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage.

performance requirement. A requirement that imposes conditions on a functional requirement; for example a requirement that specifies the speed, accuracy, or memory usage with which a given function must be performed. *Contrast with:* **design requirement; functional requirement; implementation requirement; interface requirement; physical requirement.**

performance specification. A document that specifies the performance characteristics that a system or component must possess. These characteristics typically include speed, accuracy, and memory usage. Often part of a requirements specification.

performance testing. Testing conducted to evaluate the compliance of a system or component with specified performance requirements. *See also:* **functional testing.**

Petri net. An abstract, formal model of information flow, showing static and dynamic properties of a system. A Petri net is usually represented as a graph having two types of nodes (called places and transitions) connected by arcs, and markings (called tokens) indicating dynamic properties.

physical configuration audit (PCA). An audit conducted to verify that a configuration item, as built, conforms to the technical documentation that defines it. *See also:* **functional configuration audit.**

physical requirement. A requirement that specifies a physical characteristic that a system or system component must possess; for example, material, shape, size, weight. *Contrast with:* **design requirement; functional requirement; implementation requirement; interface requirement; performance requirement.**

pipeline. A software or hardware design technique in which the output of one process serves as input to a second, the output of the second process serves as input to a third, and

so on, often with simultaneity within a single cycle time.

plan standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of a scheme for accomplishing defined objectives or work within specified resources.

playback. *See:* **reversible execution.**

pointer. A data item that specifies the location of another data item; for example, a data item that specifies the address of the next employee record to be processed.

port-to-port time. The elapsed time between the application of a stimulus to an input interface and the appearance of the response at an output interface. *See also:* **response time; think time; turnaround time.**

portability. The ease with which a system or component can be transferred from one hardware or software environment to another. *Syn:* **transportability.** *See also:* **machine independent.**

post-tested iteration. *See:* **UNTIL.**

postamble breakpoint. *See:* **epilog breakpoint.**

postmortem dump. A dump that is produced upon abnormal termination of a computer program. *See also:* **change dump; dynamic dump; memory dump; selective dump; snapshot dump; static dump.**

postprocessor. A computer program or routine that carries out some final processing step after the completion of the primary process; for example, a routine that reformats data for output. *Contrast with:* **preprocessor.**

practices. (IEEE Std 983-1986 [7]) Requirements employed to prescribe a disciplined uniform approach to the software development process. *See also:* **conventions; standards.**

pragma. *See:* **pseudo-instruction.**

pre-tested iteration. *See:* **WHILE.**

preamble breakpoint. *See:* **prolog breakpoint.**

precision. The degree of exactness or discrimination with which a quantity is stated; for example, a precision of 2 decimal places versus a precision of 5 decimal places. *Contrast with:* **accuracy.**

precompiler. A computer program or routine that processes source code and generates equivalent code that is acceptable to a compiler. For example, a routine that converts structured FORTRAN to ANSI-standard FORTRAN. *See also:* **preprocessor.**

preliminary design. (1) The process of analyzing design alternatives and defining the architecture, components, interfaces, and timing and sizing estimates for a system or component. *See also:* **detailed design.**
(2) The result of the process in (1).

preliminary design review (PDR). (1) A review conducted to evaluate the progress, technical adequacy, and risk resolution of the selected design approach for one or more configuration items; to determine each design's compatibility with the requirements for the configuration item; to evaluate the degree of definition and assess the technical risk associated with the selected manufacturing methods and processes; to establish the existence and compatibility of the physical and functional interfaces among the configuration items and other items of equipment, facilities, software and personnel; and, as applicable, to evaluate the preliminary operational and support documents. *See also:* **critical design review; system design review.**
(2) A review as in (1) of any hardware or software component.

preprocessor. A computer program or routine that carries out some processing step prior to the primary process; for example, a precompiler or other routine that reformats code or data for processing. *Contrast with:* **postprocessor.**

prestore. To store data that are required by a computer program or routine before the program or routine is entered.

prettyprinting. The use of indentation, blank lines, and other visual cues to show the logical structure of a program.

preventive maintenance. Maintenance performed for the purpose of preventing problems before they occur.

primitive type. *See:* **atomic type.**

priority. The level of importance assigned to an item.

priority interrupt. An interrupt performed to permit execution of a process that has a higher priority than the process currently executing.

private type. A data type whose structure and possible values are defined but are not revealed to the user of the type. *See also:* **information hiding.**

privileged instruction. A computer instruction that can be executed only by a supervisory program.

privileged state. *See:* **supervisor state.**

problem-oriented language. A programming language designed for the solution of a given class of problems. Examples are list processing languages, information retrieval languages, simulation languages.

problem state. In the operation of a computer system, a state in which programs other than the supervisory program can execute. *Syn:* **slave state; user state.** *Contrast with:* **supervisor state.**

procedural cohesion. A type of cohesion in which the tasks performed by a software module all contribute to a given program procedure, such as an iteration or decision process. *Contrast with:* **coincidental cohesion; communicational cohesion; functional cohesion; logical cohesion; sequential cohesion; temporal cohesion.**

procedural language. A programming language in which the user states a specific set of instructions that the computer must perform in a given sequence. All widely-used programming languages are of this type. *Syn:* **procedure-oriented language.** *Contrast with:* **nonprocedural language.** *See also:* **algebraic language; algorithmic language; list processing language; logic programming language.**

procedure. (1) A course of action to be taken to perform a given task.
(2) A written description of a course of action as in (1); for example, a documented test procedure.
(3) A portion of a computer program that is named and that performs a specific action.

procedure-oriented language. *See:* **procedural language.**

process. (1) A sequence of steps performed for a given purpose; for example, the software development process.
(2) An executable unit managed by an operating system scheduler. *See also:* **task; job.**
(3) To perform operations on data.

process management. (IEEE Std 1002-1987 [9]) The direction, control, and coordination or work performed to develop a product or perform a service. Example is quality assurance.

process standard. (IEEE Std 1002-1987 [9]) A standard that deals with the series of actions or operations used in making or achieving a product.

product analysis. (IEEE Std 1002-1987 [9]) The process of evaluating a product by manual or automated means to determine if the product has certain characteristics.

product baseline. In configuration management, the initial approved technical documentation (including, for software, the source code listing) defining a configuration item during the production, operation, maintenance, and logistic support of its life cycle. *Contrast with:* **allocated baseline; developmental configuration; functional baseline.** *See also:* **product configuration identification.**

product configuration identification. The current approved or conditionally approved technical documentation defining a configuration item during the production, operation, maintenance, and logistic support

phases of its life cycle. It prescribes all necessary physical or form, fit and function characteristics of a configuration item, the selected functional characteristics designated for production acceptance testing, and the production acceptance tests. *Contrast with:* **allocated configuration identification; functional configuration identification.** *See also:* **product baseline.**

product engineering. (IEEE Std 1002-1987 [9]) The technical processes to define, design, and construct or assemble a product.

product management. (IEEE Std 1002-1987 [9]) The definition, coordination, and control of the characteristics of a product during its development cycle. Example is configuration management.

product specification. (1) A document that specifies the design that production copies of a system or component must implement. *Note:* For software, this document describes the as-built version of the software. *See also:* **design description.**
(2) A document that describes the characteristics of a planned or existing product for consideration by potential customers or users.

product standard. (IEEE Std 1002-1987 [9]) A standard that defines what constitutes completeness and acceptability of items that are used or produced, formally or informally, during the software engineering process.

product support. (IEEE Std 1002-1987 [9]) The providing of information, assistance, and training to install and make software operational in its intended environment and to distribute improved capabilities to users.

production library. A software library containing software approved for current operational use. *Contrast with:* **master library; software development library; software repository; system library.**

professional standard. (IEEE Std 1002-1987 [9]) A standard that identifies a profession as a discipline and distinguishes it from other professions.

program. (1) *See:* **computer program.**
(2) To write a computer program.

program counter. *See:* **instruction counter.**

program definition language. *See:* **program design language.**

program design language (PDL). A specification language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document a program design. *See also:* **hardware design language; pseudo code.**

program flowchart (flow chart). *See:* **flowchart.**

program instruction. A computer instruction in a source program. *Note:* A program instruction is distinguished from a computer instruction that results from assembly, compilation, or other interpretation process.

program library. *See:* **software library.**

program listing. A printout or other human readable display of the source and, sometimes, object statements that make up a computer program.

program mutation. (1) A computer program that has been purposely altered from the intended version to evaluate the ability of test cases to detect the alteration. *See also:* **mutation testing.**
(2) The process of creating an altered program as in (1).

program network chart. A diagram that shows the relationship between two or more computer programs.

program-sensitive fault. A fault that causes a failure when some particular sequence of program steps is executed. *Contrast with:* **data-sensitive fault.**

program status word (PSW). (1) A computer word that contains information specifying the current status of a computer program. The information may include error indicators, the address of the next instruction to be executed, currently enabled interrupts,

and so on.
(2) A special-purpose register that contains a program status word as in (1).
Syn: **status word.**

program structure diagram. *See:* **structure chart.**

program support library. *See:* **software development library.**

program synthesis. The use of software tools to aid in the transformation of a program specification into a program that realizes that specification.

programmable breakpoint. A breakpoint that automatically invokes a previously specified debugging process when initiated. *See also:* **code breakpoint; data breakpoint; dynamic breakpoint; epilog breakpoint; prolog breakpoint; static breakpoint.**

programmer manual. A document that provides the information necessary to develop or modify software for a given computer system. Typically described are the equipment configuration, operational characteristics, programming features, input/output features, and compilation or assembly features of the computer system. *See also:* **diagnostic manual; installation manual; operator manual; support manual; user manual.**

programming language. A language used to express computer programs. *See also:* **assembly language; high order language; machine language.** *Contrast with:* **query language; specification language.**

programming support environment. An integrated collection of software tools accessed via a single command language to provide programming support capabilities throughout the software life cycle. The environment typically includes tools for specifying, designing, editing, compiling, loading, testing, configuration management, and project management. Sometimes called integrated programming support environment. *See also:* **scaffolding.**

programming system. A set of programming languages and the support software (editors, compilers, linkers, etc.) necessary for using these languages with a given computer system.

project file. A central repository of material pertinent to a project. Contents typically include memos, plans, technical reports, and related items. *Syn:* **project notebook.**

project library. *See:* **software development library.**

project notebook. *See:* **project file.**

project plan. A document that describes the technical and management approach to be followed for a project. The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way that the project will be organized. For example, a software development plan.

prolog breakpoint. A breakpoint that is initiated upon entry into a program or routine. *Syn:* **preamble breakpoint.** *Contrast with:* **epilog breakpoint.** *See also:* **code breakpoint; data breakpoint; dynamic breakpoint; programmable breakpoint; static breakpoint.**

prompt. (1) A symbol or message displayed by a computer system, requesting input from the user of the system.
(2) To display a symbol or message as in (1).

proof of correctness. (1) A formal technique used to prove mathematically that a computer program satisfies its specified requirements. *See also:* **assertion; formal specification; inductive assertion method; partial correctness; total correctness.**
(2) A proof that results from applying the technique in (1).

protection exception. An exception that occurs when a program attempts to write into a protected area in storage. *See also:* **addressing exception; data exception; operation exception; overflow exception; underflow exception.**

protocol. A set of conventions that govern the interaction of processes, devices, and other components within a system.

prototype. A preliminary type, form, or instance of a system that serves as a model for later stages or for the final, complete version of the system.

prototyping. A hardware and software development technique in which a preliminary version of part or all of the hardware or software is developed to permit user feedback, determine feasibility, or investigate timing or other issues in support of the development process. *See also:* **rapid prototyping.**

pseudo code (pseudocode). A combination of programming language constructs and natural language used to express a computer program design. For example:

```
IF the data arrives faster than expected,
  THEN reject every third input.
  ELSE process all data received.
ENDIF
```

pseudo instruction. A source language instruction that provides information or direction to the assembler or compiler and is not translated into a target language instruction. For example, an instruction specifying the desired format of source code listings. *Syn:* **pragma; pseudo-op; pseudo operation.**

pseudo operation. *See:* **pseudo instruction.**

pseudo-op. *See:* **pseudo instruction.**

PSW. Acronym for **program status word.**

QA. Acronym for **quality assurance.**

QC. Acronym for **quality control.**

qualification. The process of determining whether a system or component is suitable for operational use.

qualification testing. Testing conducted to determine whether a system or component is suitable for operational use. *See also:* **acceptance testing; development testing; operational testing.**

quality. (1) The degree to which a system, component, or process meets specified requirements.
(2) The degree to which a system, component, or process meets customer or user needs or expectations.

quality assurance (QA). (1) A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.
(2) A set of activities designed to evaluate the process by which products are developed or manufactured. *Contrast with:* **quality control (1).**

quality attribute. A feature or characteristic that affects an item's quality. *Syn:* **quality factor.** *Note:* In a hierarchy of quality attributes, higher level attributes may be called quality factors, lower level attributes called quality attributes.

quality control (QC). *Note:* This term has no standardized meaning in software engineering at this time. Candidate definitions are: (1) A set of activities designed to evaluate the quality of developed or manufactured products. *Contrast with:* **quality assurance (2).**
(2) The process of verifying one's own work or that of a co-worker.
(3) Synonym for **quality assurance.**

quality factor. *See:* **quality attribute.** *Note:* In a hierarchy of quality attributes, higher level attributes may be called quality factors, lower level attributes called quality attributes.

quality metric. (1) A quantitative measure of the degree to which an item possesses a given quality attribute.
(2) A function whose inputs are software data and whose output is a single numerical value that can be interpreted as the degree to which the software possesses a given quality attribute.

query language. A language used to access information stored in a database. *Contrast with:* **programming language; specification language.**

queue. A list in which items are appended to the last position of the list and retrieved from

the first position of the list. *Note:* IEEE Std 610.5-1990 [2] defines Data Management terms.

quiescing. The process of bringing a device or system to a halt by rejecting new requests for work.

random failure. A failure whose occurrence is unpredictable except in a probabilistic or statistical sense. *See also:* **intermittent fault; transient error.**

rapid prototyping. A type of prototyping in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process. *Contrast with:* **waterfall model.** *See also:* **data structure-centered design; incremental development; input-process-output; modular decomposition; object-oriented design; spiral model; stepwise refinement; structured design; transaction analysis; transform analysis.**

read. To access data from a storage device or data medium. *See also:* **destructive read; nondestructive read.** *Contrast with:* **write.**

real address. The address of a storage location in the main storage part of a virtual storage system. *Contrast with:* **virtual address.**

real storage. The main storage portion of a virtual storage system. *Contrast with:* **virtual storage.**

real time. Pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process. *Contrast with:* **batch.** *See also:* **conversational; interactive; interrupt; on-line.**

real type. A data type whose members can assume real numbers as values and can be operated on by real number arithmetic operations, such as addition, subtraction, multiplication, division, and square root. *Contrast with:* **character type; enumeration type; integer type; logical type.**

record. A set of related data items treated as a unit. For example, in stock control, the data for each invoice could constitute one record.

recovery. The restoration of a system, program, database, or other system resource to a state in which it can perform required functions. *See also:* **backward recovery; checkpoint; forward recovery.**

recursion. (1) A process in which a software module calls itself. *See also:* **simultaneous recursion.**
(2) The process of defining or generating a process or data structure in terms of itself.

recursive. (1) Pertaining to a software module that calls itself.
(2) Pertaining to a process or data structure that is defined or generated in terms of itself.

redundancy. In fault tolerance, the presence of auxiliary components in a system to perform the same or similar functions as other elements for the purpose of preventing or recovering from failures. *See also:* **active redundancy; diversity; homogeneous redundancy; standby redundancy.**

reenterable. *See:* **reentrant.**

reentrant. Pertaining to a software module that can be entered as part of one process while also in execution as part of another process and still achieve the desired results. *Syn:* **reenterable.**

reentry point. The place in a software module at which the module is reentered following a call to another module.

regression testing. Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

relative address. An address that must be adjusted by the addition of an offset to determine the address of the storage location

to be accessed. *Contrast with:* **absolute address.** *See also:* **base address; indexed address; self-relative address.**

relative loader. *See:* **relocating loader.**

reliability. The ability of a system or component to perform its required functions under stated conditions for a specified period of time. *See also:* **availability; MTBF.**

reliability growth. The improvement in reliability that results from correction of faults.

relocatable. Pertaining to code that can be loaded into any part of main memory. The starting address is established by the loader, which then adjusts the addresses in the code to reflect the storage locations into which the code has been loaded. *See also:* **relocating loader.**

relocatable address. An address that is to be adjusted by the loader when the computer program containing the address is loaded into memory. *Contrast with:* **absolute address.**

relocatable code. Code containing addresses that are to be adjusted by the loader to reflect the storage locations into which the code is loaded. *Contrast with:* **absolute code.**

relocate. To move machine code from one portion of main memory to another and to adjust the addresses so that the code can be executed in its new location.

relocating assembler. An assembler that produces relocatable code. *Contrast with:* **absolute assembler.**

relocating loader. A loader that reads relocatable code into main memory and adjusts the addresses in the code to reflect the storage locations into which the code has been loaded. *Syn:* **relative loader.** *Contrast with:* **absolute loader.**

relocation dictionary. The part of an object module or load module that identifies the addresses that must be adjusted when a relocation occurs.

relocation factor. *See:* **offset (1).**

remote batch entry. *See:* **remote job entry.**

remote job entry (RJE). Submission of jobs through a remote input device connected to a computer through a data link. *Syn:* **remote batch entry.**

repeatability. *See:* **test repeatability.**

repetitive addressing. A method of implied addressing in which the operation field of an computer instruction is understood to address the operands of the last instruction executed. *Contrast with:* **one-ahead addressing.**

replay. *See:* **reversible execution.**

report standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of describing results of engineering and management activities.

representation standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of portraying aspects of an engineering or management product.

requirement. (1) A condition or capability needed by a user to solve a problem or achieve an objective.
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.
(3) A documented representation of a condition or capability as in (1) or (2).
See also: **design requirement; functional requirement; implementation requirement; interface requirement; performance requirement; physical requirement.**

requirement standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of a requirements specification.

requirements analysis. (1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements.
(2) The process of studying and refining

system, hardware, or software requirements.

requirements phase. The period of time in the software life cycle during which the requirements for a software product are defined and documented.

requirements review. A process or meeting during which the requirements for a system, hardware item, or software item are presented to project personnel, managers, users, customers, or other interested parties for comment or approval. Types include system requirements review, software requirements review. *Contrast with:* **code review; design review; formal qualification review; test readiness review.**

requirements specification. A document that specifies the requirements for a system or component. Typically included are functional requirements, performance requirements, interface requirements, design requirements, and development standards. *Contrast with:* **design description.** *See also:* **functional specification; performance specification.**

requirements specification language. A specification language with special constructs and, sometimes, verification protocols, used to develop, analyze, and document hardware or software requirements. *See also:* **design language.**

rescue point. *See:* **restart point.**

reserved word. A word in a programming language whose meaning is fixed by the rules of that language and which, in certain or all contexts, cannot be used by the programmer for any purpose other than its intended one. Examples include IF, THEN, WHILE.

reset. To set a variable, register, or other storage location back to a prescribed state. *See also:* **clear; initialize.**

resident control program. *See:* **kernel (1).**

residual control. A microprogramming technique in which the meaning of a field in a microinstruction depends on the value in an auxiliary register. *Contrast with:* **bit steering.** *See also:* **two-level encoding.**

resource allocation. *See:* **computer resource allocation.**

resource management. (IEEE Std 1002-1987 [9]) The identification, estimation, allocation, and monitoring of the means used to develop a product or perform a service. Example is estimating.

response time. The elapsed time between the end of an inquiry or command to an interactive computer system and the beginning of the system's response. *See also:* **port-to-port time; think time; turnaround time.**

restart. To cause a computer program to resume execution after a failure, using status and results recorded at a checkpoint.

restart point. A point in a computer program at which execution can be restarted following a failure. *Syn:* **rescue point.**

retirement. (1) Permanent removal of a system or component from its operational environment.
(2) Removal of support from an operational system or component.
See also: **software life cycle; system life cycle.**

retirement phase. The period of time in the software life cycle during which support for a software product is terminated.

retrospective trace. A trace produced from historical data recorded during the execution of a computer program. *Note:* This differs from an ordinary trace, which is produced cumulatively during program execution. *See also:* **execution trace; subroutine trace; symbolic trace; variable trace.**

return. (1) To transfer control from a software module to the module that called it. *See also:* **return code.**
(2) To assign a value to a parameter that is accessible by a calling module; for example, to assign the value 25 to parameter AGE for use by a calling module. *See also:* **return value.**

(3) A computer instruction or process that performs the transfer in (1).

return code. A code used to influence the execution of a calling module following a return from a called module.

return value. A value assigned to a parameter by a called module for access by the calling module.

reusability. The degree to which a software module or other work product can be used in more than one computer program or software system. *See also:* **generality.**

reusable. Pertaining to a software module or other work product that can be used in more than one computer program or software system.

reverse execution. *See:* **reversible execution.**

reversible execution. A debugging technique in which a history of program execution is recorded and then replayed under the user's control, in either the forward or backward direction. *Syn:* **backward execution; playback; replay; reverse execution.**

review. A process or meeting during which a work product, or set of work products, is presented to project personnel, managers, users, customers, or other interested parties for comment or approval. Types include code review, design review, formal qualification review, requirements review, test readiness review.

RJE. Acronym for **remote job entry.**

robustness. The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions. *See also:* **error tolerance; fault tolerance.**

roll in. To transfer data or computer program segments from auxiliary storage to main storage. *Contrast with:* **roll out.** *See also:* **swap.**

roll out. To transfer data or computer program segments from main storage to auxiliary storage for the purpose of freeing main storage for other uses. *Contrast with:* **roll in.** *See also:* **swap.**

root compiler. A compiler whose output is a machine independent, intermediate-level representation of a program. A root compiler, when combined with a code generator, comprises a full compiler.

routine. A subprogram that is called by other programs and subprograms. *Note:* The terms "routine," "subprogram," and "subroutine" are defined and used differently in different programming languages; the preceding definition is advanced as a proposed standard. *See also:* **coroutine; subroutine.**

rule-based language. A nonprocedural language that permits the user to state a set of rules and to express queries or problems that use these rules. *See also:* **declarative language; interactive language.**

run. (1) In software engineering, a single, usually continuous, execution of a computer program. *See also:* **run time.**
(2) To execute a computer program.

run stream. *See:* **job stream.**

run time. (1) The instant at which a computer program begins to execute.
(2) The period of time during which a computer program is executing.
(3) *See:* **execution time.**

running time. *See:* **execution time.**

scaffolding. Computer programs and data files built to support software development and testing, but not intended to be included in the final product. For example, dummy routines or files, test case generators, software monitors, stubs. *See also:* **programming support environment.**

scheduler. A computer program, usually part of an operating system, that schedules, initiates, and terminates jobs.

SCN. Acronym for **specification change notice.**

SDD. (1) Acronym for **software design description.**
(2) (DoD) Acronym for software design document. *See:* **software design description.**

SDP. Acronym for **software development plan.**

SDR. Acronym for **system design review.**

second generation language (2GL). *See:* **assembly language.**

security kernel. A small, self-contained collection of key security-related statements that works as a privileged part of an operating system, specifying and enforcing criteria that must be met for programs and data to be accessed.

segment. (1) One of the subsystems or combinations of subsystems that make up an overall system; for example, the accounts payable segment of a financial system.
(2) In storage allocation, a self-contained portion of a computer program that can be executed without maintaining the entire program in main storage. *See also:* **page.**
(3) A collection of data that is stored or transferred as a unit.
(4) In path analysis, a sequence of computer program statements between two consecutive branch points.
(5) To divide a system, computer program, or data file into segments as in (1), (2), or (3).

selective choice construct. *See:* **branch.**

selective dump. A dump of designated storage location areas only. *See also:* **change dump; dynamic dump; memory dump; postmortem dump; snapshot dump; static dump.**

selective trace. A variable trace that involves only selected variables. *See also:* **execution trace; retrospective trace; subroutine trace; symbolic trace; variable trace.**

self-descriptiveness. The degree to which a system or component contains enough information to explain its objectives and properties. *See also:* **maintainability; testability; usability.**

self-documented. Pertaining to source code that contains comments explaining its objectives, operation, and other information useful in understanding and maintaining the code.

self-relative address. An address that must be added to the address of the instruction in which it appears to obtain the address of the storage location to be accessed. *See also:* **base address; indexed address; offset; relative address.**

semantic error. An error resulting from a misunderstanding of the relationship of symbols or groups of symbols to their meanings in a given language. *Contrast with:* **syntactic error.**

semantics. The relationships of symbols or groups of symbols to their meanings in a given language. *Contrast with:* **syntax.**

semaphore. A shared variable used to synchronize concurrent processes by indicating whether an action has been completed or an event has occurred. *See also:* **flag; indicator.**

sequential. Pertaining to the occurrence of two or more events or activities in such a manner that one must finish before the next begins. *Syn:* **serial (2).** *See also:* **consecutive.**

sequential cohesion. A type of cohesion in which the output of one task performed by a software module serves as input to another task performed by the module. *Contrast with:* **coincidental cohesion; communicational cohesion; functional cohesion; logical cohesion; procedural cohesion; temporal cohesion.**

sequential construct. *See:* **serial construct.**

serial. (1) Pertaining to the sequential transfer, occurrence, or processing of the individual parts of a whole, such as the bits of a character, using the same facilities for successive parts. *Contrast with:* **parallel (1).**
(2) *See:* **sequential.**

serial construct. A program construct consisting of a sequence of steps not involving a decision or loop. *Syn:* **sequential construct.**

set-up time. The period of time during which a system or component is being prepared for a specific operation. *See also:* **busy time; down time; idle time; up time.**

severity. *See:* **criticality.**

shell. A computer program or routine that provides an interface between the user and a computer system or program.

simple buffering. A buffering technique in which a buffer is allocated to a computer program for the duration of the program's execution. *Contrast with:* **dynamic buffering.**

simplicity. The degree to which a system or component has a design and implementation that is straightforward and easy to understand. *Contrast with:* **complexity.**

simulation. (1) A model that behaves or operates like a given system when provided a set of controlled inputs. *See also:* **emulation.**
(2). The process of developing or using a model as in (1).

simulator. A device, computer program, or system that behaves or operates like a given system when provided a set of controlled inputs. *See also:* **emulator.**

simultaneous. Pertaining to the occurrence of two or more events at the same instant of time. *Contrast with:* **concurrent.**

simultaneous recursion. A situation in which two software modules call each other.

single-address instruction. *See:* **one-address instruction.**

single-level encoding. A microprogramming technique in which different microoperations are encoded as different values in the same field of a microinstruction. *Contrast with:* **two-level encoding.**

single-operand instruction. *See:* **one-address instruction.**

single-step execution. *See:* **single-step operation.**

single-step operation. A debugging technique in which a single computer instruction, or part of an instruction, is executed in response to an external signal. *Syn:* **single-step execution; step-by-step operation.**

sizing. The process of estimating the amount of computer storage or the number of source lines required for a software system or component. *Contrast with:* **timing.**

slave state. *See:* **problem state.**

snapshot dump. A dynamic dump of the contents of one or more specified storage areas. *See also:* **change dump; dynamic dump; memory dump; postmortem dump; selective dump; static dump.**

soft error.* *See:* **transient error.**
* Deprecated.

soft failure. A failure that permits continued operation of a system with partial operational capability. *Contrast with:* **hard failure.**

software. Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system. *See also:* **application software; support software; system software.** *Contrast with:* **hardware.**

software characteristic. (IEEE Std 1008-1987 [10]) An inherent, possibly accidental, trait, quality, or property of software (for example, functionality, performance, attributes, design constraints, number of states, lines or branches).

software configuration management. *See:* **configuration management.**

software design description (SDD). (1) (IEEE Std 1012-1986 [12]) A representation of software created to facilitate analysis, planning, implementation, and decision making. The software design description is used as a medium for communicating software design information, and may be

thought of as a blueprint or model of the system.
(2) (IEEE Std 1016-1987 [13]) A representation of a software system created to facilitate analysis, planning, implementation, and decision making. A blueprint or model of the software system. The SDD is used as the primary medium for communicating software design information.

software development cycle. The period of time that begins with the decision to develop a software product and ends when the software is delivered. This cycle typically includes a requirements phase, design phase, implementation phase, test phase, and sometimes, installation and checkout phase. *Contrast with:* **software life cycle.**
Notes: (1) The phases listed above may overlap or be performed iteratively, depending upon the software development approach used.
(2) This term is sometimes used to mean a longer period of time, either the period that ends when the software is no longer being enhanced by the developer, or the entire software life cycle.

software development file (SDF). A collection of material pertinent to the development of a given software unit or set of related units. Contents typically include the requirements, design, technical reports, code listings, test plans, test results, problem reports, schedules, and notes for the units. *Syn:* **software development folder; software development notebook; unit development folder.**

software development folder. *See:* **software development file.**

software development library. A software library containing computer readable and human readable information relevant to a software development effort. *Syn:* **project library; program support library.** *Contrast with:* **master library; production library; software repository; system library.**

software development notebook. *See:* **software development file.**

software development plan (SDP). A project plan for a software development project.

software development process. The process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use. *Note:* These activities may overlap or be performed iteratively. *See also:* **incremental development; rapid prototyping; spiral model; waterfall model.**

software diversity. A software development technique in which two or more functionally identical variants of a program are developed from the same specification by different programmers or programming teams with the intent of providing error detection, increased reliability, additional documentation, or reduced probability that programming or compiler errors will influence the end results. *See also:* **diversity.**

software engineering. (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
(2) The study of approaches as in (1).

software engineering environment. The hardware, software, and firmware used to perform a software engineering effort. Typical elements include computer equipment, compilers, assemblers, operating systems, debuggers, simulators, emulators, test tools, documentation tools, and database management systems.

software feature. (1) (IEEE Std 829-1983 [5]) A distinguishing characteristic of a software item (for example, performance, portability, or functionality).
(2) (IEEE Std 1008-1987 [10]) A software characteristic specified or implied by requirements documentation (for example, functionality, performance, attributes, or design constraints).

software item. (IEEE Std 829-1983 [5]) Source code, object code, job control code, control data, or a collection of these items.

software library. A controlled collection of software and related documentation designed to aid in software development, use, or maintenance. Types include master library, production library, software development library, software repository, system library. *Syn:* **program library.**

software life cycle. The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and, sometimes, retirement phase. *Note:* These phases may overlap or be performed iteratively. *Contrast with:* **software development cycle.**

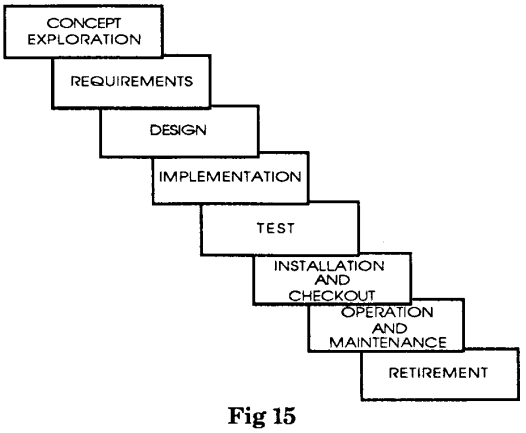

Fig 15
Sample Software Life Cycle

software maintenance. *See:* **maintenance (1).**

software monitor. A software tool that executes concurrently with another program and provides detailed information about the execution of the other program. *See also:* **hardware monitor; monitor.**

software product. (1) The complete set of computer programs, procedures, and possibly associated documentation and data designated for delivery to a user.
(2) Any of the individual items in (1).

software quality assurance. *See:* **quality assurance.**

software quality metric. *See:* **quality metric.**

software repository. A software library providing permanent, archival storage for software and related documentation. *Contrast with:* **master library; production library; software development library; system library.**

software requirements review (SRR). (1) A review of the requirements specified for one or more software configuration items to evaluate their responsiveness to and interpretation of the system requirements and to determine whether they form a satisfactory basis for proceeding into preliminary design of the configuration items. *See also:* **system requirements review.** *Note:* This review is called software specification review by the U.S. Department of Defense.
(2) A review as in (1) for any software component.

software requirements specification (SRS). (IEEE Std 1012-1986 [12]) Documentation of the essential requirements (functions, performance, design constraints, and attributes) of the software and its external interfaces.

software specification review (SSR). *See:* **software requirements review.**

software test incident. (IEEE Std 1008-1987 [10]) Any event occurring during the execution of a software test that requires investigation.

software tool. A computer program used in the development, testing, analysis, or maintenance of a program or its documentation. Examples include comparator, cross-reference generator, decompiler, driver, editor, flowcharter, monitor, test case generator, timing analyzer.

source address. The address of a device or storage location from which data is to be transferred. *Contrast with:* **destination address.**

source code. Computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler, or other

translator. *Note:* A source program is made up of source code. *Contrast with:* **object code.**

source code generator. *See:* **code generator (2).**

source language. The language in which the input to a machine-aided translation process is represented. For example, the language used to write a computer program. *Contrast with:* **target language.**

source program. A computer program that must be compiled, assembled, or otherwise translated in order to be executed by a computer. *Contrast with:* **object program.**

specific address. *See:* **absolute address.**

specific code. *See:* **absolute code.**

specification. A document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system or component, and, often, the procedures for determining whether these provisions have been satisfied. *See also:* **formal specification; product specification; requirements specification.**

specification change notice (SCN). A document used in configuration management to propose, transmit, and record changes to a specification. *See also:* **configuration control; engineering change; notice of revision.**

specification language. A language, often a machine-processible combination of natural and formal language, used to express the requirements, design, behavior, or other characteristics of a system or component. For example, a design language or requirements specification language. *Contrast with:* **programming language; query language.**

specification tree. A diagram that depicts all of the specifications for a given system and shows their relationships to one another. *See also:* **documentation tree.**

spiral model. A model of the software development process in which the constituent activities, typically requirements analysis, preliminary and detailed design, coding, integration, and testing, are performed iteratively until the software is complete. *Contrast with:* **waterfall model.** *See also:* **incremental development; rapid prototyping.**

spool. To read input data, or write output data, to auxiliary or main storage for later processing or output, in order to permit input/output devices to operate concurrently with job execution. Derived from the acronym SPOOL for Simultaneous Peripheral Output On Line.

spooler. A program that initiates and controls spooling.

SRR. (1) Acronym for **software requirements review.**
(2) (DoD) Acronym for system requirements review.

SRS. Acronym for **software requirements specification.**

SSR. Acronym for **software specification review.** *See:* **software requirements review.**

stand-alone. Pertaining to hardware or software that is capable of performing its function without being connected to other components; for example, a stand-alone word processing system.

standards. (IEEE Std 983-1986 [7]) Mandatory requirements employed and enforced to prescribe a disciplined uniform approach to software development, that is, mandatory conventions and practices are in fact standards. *See also:* **practices; standards.**

standby redundancy. In fault tolerance, the use of redundant elements that are left inoperative until a failure occurs in a primary element. *Contrast with:* **active redundancy.**

standby time. *See:* **idle time.**

starting address. The address of the first instruction of a computer program in main storage. *Note:* This address may or may not be the same as the program's origin, depending upon whether there are data

preceding the first instruction. *Contrast with:* **origin.** *See also:* **assembled origin; loaded origin.**

state. (1) A condition or mode of existence that a system, component, or simulation may be in; for example, the pre-flight state of an aircraft navigation program or the input state of given channel.
(2) The values assumed at a given instant by the variables that define the characteristics of a system, component, or simulation.

state data. (IEEE Std 1008-1987 [10]) Data that defines an internal state of the test unit and is used to establish that state or compare with existing states.

state diagram. A diagram that depicts the states that a system or component can assume, and shows the events or circumstances that cause or result from a change from one state to another.

state transition diagram. *See:* **state diagram.**

statement. In a programming language, a meaningful expression that defines data, specifies program actions, or directs the assembler or compiler. *See also:* **assignment statement; control statement; declaration.**

statement testing. Testing designed to execute each statement of a computer program. *Contrast with:* **branch testing; path testing.**

static. Pertaining to an event or process that occurs without computer program execution; for example, static analysis, static binding. *Contrast with:* **dynamic.**

static analysis. The process of evaluating a system or component based on its form, structure, content, or documentation. *Contrast with:* **dynamic analysis.** *See also:* **inspection; walk-through.**

static binding. Binding performed prior to the execution of a computer program and not subject to change during program execution. *Contrast with:* **dynamic binding.**

static breakpoint. A breakpoint that can be set at compile time, such as entry into a given routine. *Contrast with:* **dynamic breakpoint.** *See also:* **code breakpoint; data breakpoint; epilog breakpoint; programmable breakpoint; prolog breakpoint.**

static dump. A dump that is produced before or after the execution of a computer program. *Contrast with:* **dynamic dump.** *See also:* **change dump; memory dump; postmortem dump; selective dump; snapshot dump.**

static error. An error that is independent of the time-varying nature of an input. *Contrast with:* **dynamic error.**

status code. A code used to indicate the results of a computer program operation. For example, a code indicating a carry, an overflow, or a parity error. *Syn:* **condition code.**

step-by-step operation. *See:* **single-step operation.**

stepwise refinement. A software development technique in which data and processing steps are defined broadly at first and then further defined with increasing detail. *See also:* **data structure-centered design; input-processoutput; modular decomposition; object-oriented design; rapid prototyping; structured design; transaction analysis; transform analysis.**

stop. To terminate the execution of a computer program. *Syn:* **halt (1).** *Contrast with:* **pause.**

storage allocation. An element of computer resource allocation, consisting of assigning storage areas to specific jobs and performing related procedures, such as transfer of data between main and auxiliary storage, to support the assignments made. *See also:* **buffer; contiguous allocation; cyclic search; memory compaction; overlay; paging; virtual storage.**

storage breakpoint. *See:* **data breakpoint.**

storage capacity. The maximum number of items that can be held in a given storage

device; usually measured in words or bytes. *See also:* **channel capacity; memory capacity.**

storage efficiency. The degree to which a system or component performs its designated functions with minimum consumption of available storage. *See also:* **execution efficiency.**

store. (1) To place or retain data in a storage device.
(2) To copy computer instructions or data from a register to internal storage or from internal storage to external storage. *Contrast with:* **load (2).** *See also:* **fetch; move.**

straight-line code. A sequence of computer instructions in which there are no loops.

straight-line coding. A programming technique in which loops are avoided by stating explicitly and in full all of the instructions that would be involved in the execution of each loop. *See also:* **unwind.**

stratified language. A language that cannot be used as its own metalanguage. Examples include FORTRAN, COBOL. *Contrast with:* **unstratified language.**

stress testing. Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements. *See also:* **boundary value.**

strong typing. A feature of some programming languages that requires the type of each data item to be declared, precludes the application of operators to inappropriate data types, and prevents the interaction of data items of incompatible types.

structural testing. Testing that takes into account the internal mechanism of a system or component. Types include branch testing, path testing, statement testing. *Syn:* **glass-box testing; white-box testing.** *Contrast with:* **functional testing (1).**

structure chart. A diagram that identifies modules, activities, or other entities in a system or computer program and shows how larger or more general entities break down into smaller, more specific entities. *Note:* The result is not necessarily the same as that shown in a call graph. *Syn:* **hierarchy chart; program structure chart.** *Contrast with:* **call graph.**

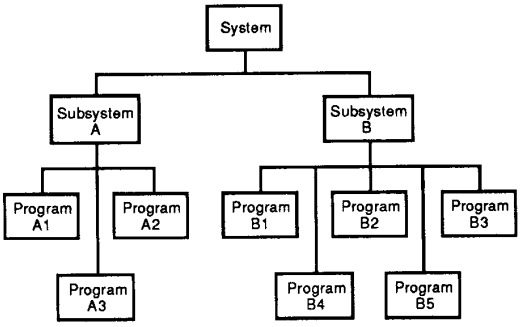

Fig 16
Structure Chart

structure clash. In software design, a situation in which a module must deal with two or more data sets that have incompatible data structures. *See also:* **data structure-centered design; order clash.**

structured design. (1) Any disciplined approach to software design that adheres to specified rules based on principles such as modularity, top-down design, and stepwise refinement of data, system structures, and processing steps. *See also:* **data structure-centered design; input-process-output; modular decomposition; object-oriented design; rapid prototyping; stepwise refinement; transaction analysis; transform analysis.**
(2) The result of applying the approach in (1).

structured program. A computer program constructed of a basic set of control structures, each having one entry and one exit. The set of control structures typically includes: sequence of two or more instructions, conditional selection of one of two or more sequences of instructions, and repetition of a sequence of instructions. *See also:* **structured design.**

structured programming. Any software development technique that includes struc-

tured design and results in the development of structured programs.

structured programming language. A programming language that provides the structured program constructs, namely, single-entry-single-exit sequences, branches, and loops, and facilitates the development of structured programs. *See also:* **block-structured language.**

stub. (1) A skeletal or special-purpose implementation of a software module, used to develop or test a module that calls or is otherwise dependent on it.
(2) A computer program statement substituting for the body of a software module that is or will be defined elsewhere.

subprogram. A separately compilable, executable component of a computer program. *Note:* The terms "routine," "subprogram," and "subroutine" are defined and used differently in different programming languages; the preceding definition is advanced as a proposed standard. *See also:* **coroutine; main program; routine; subroutine.**

subroutine. A routine that returns control to the program or subprogram that called it. *Note:* The terms "routine," "subprogram," and "subroutine" are defined and used differently in different programming languages; the preceding definition is advanced as a proposed standard. *Contrast with:* **coroutine.** *See also:* **closed subroutine; open subroutine.**

subroutine trace. A record of all or selected subroutines or function calls performed during the execution of a computer program and, optionally, the values of parameters passed to and returned by each subroutine or function. *Syn:* **call trace.** *See also:* **execution trace; retrospective trace; subroutine trace; symbolic trace; variable trace.**

subsystem. A secondary or subordinate system with a larger system.

subtype. A subset of a data type, obtained by constraining the set of possible values of the data type. *Note:* The operations applicable to the subtype are the same as those of the original data type. *See also:* **derived type.**

supervisor. *See:* **supervisory program.**

supervisor state. In the operation of a computer system, a state in which the supervisory program is executing. This state usually has higher priority than, and precludes the execution of, application programs. *Syn:* **executive state; master state; privileged state.** *Contrast with:* **problem state.**

supervisory program. A computer program, usually part of an operating system, that controls the execution of other computer programs and regulates the flow of work in a computer system. *Syn:* **control program; executive; executive program; supervisor.** *See also:* **supervisor state.**

support. The set of activities necessary to ensure that an operational system or component fulfills its original requirements and any subsequent modifications to those requirements. For example, software or hardware maintenance, user training. *See also:* **software life cycle; system life cycle.**

support manual. A document that provides the information necessary to service and maintain an operational system or component throughout its life cycle. Typically described are the hardware and software that make up the system or component and procedures for servicing, repairing, or reprogramming it. *Syn:* **maintenance manual.** *See also:* **diagnostic manual; installation manual; operator manual; programmer manual; user manual.**

support software. Software that aids in the development or maintenance of other software, for example, compilers, loaders, and other utilities. *Contrast with:* **application software.** *See also:* **system software.**

swap. (1) An exchange of the contents of two storage areas, usually an area of main storage with an area of auxiliary storage. *See also:* **roll in; roll out.**
(2) To perform an exchange as in (1).

symbol table. A table that presents program symbols and their corresponding addresses, values, and other attributes.

symbolic address. An address expressed as a name or label that must be translated to the absolute address of the device or storage location to be accessed. *Contrast with:* **absolute address.**

symbolic execution. A software analysis technique in which program execution is simulated using symbols, such as variable names, rather than actual values for input data, and program outputs are expressed as logical or mathematical expressions involving these symbols.

symbolic language. A programming language that expresses operations and addresses in symbols convenient to humans rather than in machine language. Examples are assembly language, high order language. *Contrast with:* **machine language.**

symbolic trace. A record of the source statements and branch outcomes that are encountered when a computer program is executed using symbolic, rather than actual, values for input data. *See also:* **execution trace; retrospective trace; subroutine trace; variable trace.**

syntactic error. A violation of the structural or grammatical rules defined for a language; for example, using the statement $B + C = A$ in Fortran, rather than the correct $A = B + C$. *Syn:* **syntax error.** *Contrast with:* **semantic error.**

syntax. The structural or grammatical rules that define how the symbols in a language are to be combined to form words, phrases, expressions, and other allowable constructs. *Contrast with:* **semantics.**

syntax error. *See:* **syntactic error.**

synthetic address. *See:* **generated address.**

system. A collection of components organized to accomplish a specific function or set of functions.

system design review (SDR). A review conducted to evaluate the manner in which the requirements for a system have been allocated to configuration items, the system engineering process that produced the allocation, the engineering planning for the next phase of the effort, manufacturing considerations, and the planning for production engineering. *See also:* **critical design review; preliminary design review.**

system development cycle. The period of time that begins with the decision to develop a system and ends when the system is delivered to its end user. *Note:* This term is sometimes used to mean a longer period of time, either the period that ends when the system is no longer being enhanced, or the entire system life cycle. *Contrast with:* **system life cycle.** *See also:* **software development cycle.**

system flowchart (flow chart). *See:* **flowchart.**

system library. A software library containing system-resident software that can be accessed for use or incorporated into other programs by reference; for example, a macro library. *Contrast with:* **master library; production library; software development library; software repository.**

system life cycle. The period of time that begins when a system is conceived and ends when the system is no longer available for use. *See also:* **system development cycle; software life cycle.**

system model. In computer performance evaluation, a representation of a system depicting the relationships between workloads and performance measures in the system. *See also:* **workload model.**

system profile. A set of measurements used in computer performance evaluation, describing the proportion of time each of the major resources in a computer system is busy, divided by the time that resource is available.

system requirements review (SRR). A review conducted to evaluate the completeness and adequacy of the requirements defined for a

system; to evaluate the system engineering process that produced those requirements; to assess the results of system engineering studies; and to evaluate system engineering plans. *See also:* **software requirements review.**

system resources chart. *See:* **block diagram.**

system software. Software designed to facilitate the operation and maintenance of a computer system and its associated programs; for example, operating systems, assemblers, utilities. *Contrast with:* **application software.** *See also:* **support software.**

system testing. Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. *See also:* **component testing; integration testing; interface testing; unit testing.**

target language. The language in which the output from a machine-aided translation process is represented. For example, the language output by an assembler or compiler. *Syn:* **object language.** *Contrast with:* **source language.**

target machine. (1) The computer on which a program is intended to execute. *Contrast with:* **host machine (1).**
(2) A computer being emulated by another computer. *Contrast with:* **host machine (2).**

target program. *See:* **object program.**

task. (1) A sequence of instructions treated as a basic unit of work by the supervisory program of an operating system.
(2) In software design, a software component that can operate in parallel with other software components.

taxonomy. (IEEE Std 1002-1987 [9]) A scheme that partitions a body of knowledge and defines the relationships among the pieces. It is used for classifying and understanding the body of knowledge.

technical management. (IEEE Std 1002-1987 [9]) The application of technical and administrative resources to plan, organize, and control engineering functions.

technical standard. (IEEE Std 1002-1987 [9]) A standard that describes the characteristics of applying accumulated technical or management skills and methods in the creation of a product or performing a service.

techniques. (IEEE Std 983-1986 [7]) Technical and managerial procedures that aid in the evaluation and improvement of the software development process.

temporal cohesion. A type of cohesion in which the tasks performed by a software module are all required at a particular phase of program execution; for example, a module containing all of a program's initialization tasks. *Contrast with:* **coincidental cohesion; communicational cohesion; functional cohesion; logical cohesion; procedural cohesion; sequential cohesion.**

termination construct. A program construct that results in a halt or exit.

test. (1) An activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component.
(2) To conduct an activity as in (1).
(3) (IEEE Std 829-1983 [5]) A set of one or more test cases.
(4) (IEEE Std 829-1983 [5]) A set of one or more test procedures.
(5) (IEEE Std 829-1983 [5]) A set of one or more test cases and procedures.

test bed. An environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.

test case. (1) A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.
(2) (IEEE Std 829-1983 [5]) Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.

See also: **test case generator; test case specification.**

test case generator. A software tool that accepts as input source code, test criteria, specifications, or data structure definitions; uses these inputs to generate test input data; and, sometimes, determines expected results. *Syn:* **test data generator; test generator.**

test case specification. A document that specifies the test inputs, execution conditions, and predicted results for an item to be tested. *Syn:* **test description; test specification.** *See also:* **test incident report; test item transmittal report; test log; test plan; test procedure; test report.**

test coverage. The degree to which a given test or set of tests addresses all specified requirements for a given system or component.

test criteria. The criteria that a system or component must meet in order to pass a given test. *See also:* **acceptance criteria; pass-fail criteria.**

test data generator. *See:* **test case generator.**

test description. *See:* **test case specification.**

test design. (IEEE Std 829-1983 [5]) Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests.

test documentation. Documentation describing plans for, or results of, the testing of a system or component. Types include test case specification, test incident report, test log, test plan, test procedure, test report.

test driver. A software module used to invoke a module under test and, often, provide test inputs, control and monitor execution, and report test results. *Syn:* **test harness.**

test generator. *See:* **test case generator.**

test harness. *See:* **test driver.**

test incident report. A document that describes an event that occurred during testing which requires further investigation. *See also:* **test case specification; test item transmittal report; test log; test plan; test procedure; test report.**

test item. (IEEE Std 829-1983 [5]) A software item which is an object of testing.

test item transmittal report. (IEEE Std 829-1983 [5]) A document that identifies one or more items submitted for testing. It contains current status and location information. *See also:* **test case specification; test incident report; test log; test plan; test procedure; test report.**

test log. A chronological record of all relevant details about the execution of a test. *See also:* **test case specification; test incident report; test item transmittal report; test plan; test procedure; test report.**

test objective. (IEEE Std 1008-1987 [10]) An identified set of software features to be measured under specified conditions by comparing actual behavior with the required behavior described in the software documentation.

test phase. The period of time in the software life cycle during which the components of a software product are evaluated and integrated, and the software product is evaluated to determine whether or not requirements have been satisfied.

test plan. (1) (IEEE Std 829-1983 [5]) A document describing the scope, approach, resources, and schedule of intended test activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.
(2) A document that describes the technical and management approach to be followed for testing a system or component. Typical contents identify the items to be tested, tasks to be performed, responsibilities, schedules and required resources for the testing activity.
See also: **test case specification; test incident report; test item transmittal report; test log; test procedure; test report.**

test procedure. (1) Detailed instructions for the set-up, execution, and evaluation of results for a given test case.
(2) A document containing a set of associated instructions as in (1).
(3) (IEEE Std 829-1983 [5]) Documentation specifying a sequence of actions for the execution of a test.
Syn: **test procedure specification; test script.**
See also: **test case specification; test incident report; test item transmittal report; test log; test plan; test report.**

test procedure specification. *See:* **test procedure.**

test readiness review (TRR). (1) A review conducted to evaluate preliminary test results for one or more configuration items; to verify that the test procedures for each configuration item are complete, comply with test plans and descriptions, and satisfy test requirements; and to verify that a project is prepared to proceed to formal testing of the configuration items.
(2) A review as in (1) for any hardware or software component.
Contrast with: **code review; formal qualification review; design review; requirements review.**

test repeatability. An attribute of a test, indicating that the same results are produced each time the test is conducted.

test report. A document that describes the conduct and results of the testing carried out for a system or component. *Syn:* **test summary report.** *See also:* **test case specification; test incident report; test item transmittal report; test log; test plan; test procedure.**

test script. *See*: **test procedure.**

test set architecture. (IEEE Std 1008-1987 [10]) The nested relationships between sets of test cases that directly reflect the hierarchic decomposition of the test objectives.

test specification. *See:* **test case specification.**

test summary report. (IEEE Std 829-1983 [5]) A document summarizing testing activities and results. It also contains an evaluation of the corresponding test items. *See also:* **test case specification; test incident report; test item transmittal report; test log; test plan; test procedure; test report.**

test unit. (IEEE Std 1008-1987 [10]) A set of one or more computer program modules together with associated control data, (for example, tables), usage procedures, and operating procedures that satisfy the following conditions: (a) All modules are from a single computer program; (b) At least one of the new or changed modules in the set has not completed the unit test; (c) The set of modules together with its associated data and procedures are the sole object of a testing process.

testability. (1) The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.
(2) The degree to which a requirement is stated in terms that permit establishment of test criteria and performance of tests to determine whether those criteria have been met.

testing. (1) The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component.
(2) (IEEE Std 829-1983 [5]) The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software items.
See also: **acceptance testing; benchmark; checkout; component testing; development testing; dynamic analysis; formal testing; functional testing; informal testing; integration testing; interface testing; loopback testing; mutation testing; operational testing; performance testing; qualification testing; regression testing; stress testing; structural testing; system testing; unit testing.**

text editor. A computer program, often part of a word processing system, that allows a user to enter, alter, and view text. *Syn:* **editor.**

think time. The elapsed time between the end of a prompt or message generated by an interactive system and the beginning of a human user's response. *See also:* **port-to-port time; response time; turnaround time.**

third generation language (3GL). *See:* **high order language.**

thrashing. A state in which a computer system is expending most or all of its resources on overhead operations, such as swapping data between main and auxiliary storage, rather than on intended computing functions.

three-address instruction. A computer instruction that contains three address fields. For example, an instruction to add the contents of locations A and B, and place the results in location C. *Contrast with:* **one-address instruction; two-address instruction; four-address instruction; zero-address instruction.**

three-plus-one address instruction. A computer instruction that contains four address fields, the fourth containing the address of the instruction to be executed next. For example, an instruction to add the contents of locations A and B, place the results in location C, then execute the instruction at location D. *Contrast with:* **one-plus-one address instruction; two-plus-one address instruction; four-plus-one address instruction.**

throughput. The amount of work that can be performed by a computer system or component in a given period of time; for example, number of jobs per day. *See also:* **turnaround time; workload model.**

tier chart. *See:* **call graph.**

time out. (1) A condition that occurs when a predetermined amount of time elapses without the occurrence of an expected event. For example, the condition that causes termination of an on-line process if no user input is received within a specified period of time.
(2) To experience the condition in (1).

time sharing. A mode of operation that permits two or more users to execute computer programs concurrently on the same computer system by interleaving the execution of their program. *Note:* Time sharing may be implemented by time slicing, priority-based interrupts, or other scheduling methods.

time slicing. A mode of operation in which two or more processes are each assigned a small, fixed amount of continuous processing time on the same processor, and the processes execute in a round-robin manner, each for its allotted time, until all are completed.

timing. The process of estimating or measuring the amount of execution time required for a software system or component. *Contrast with:* **sizing.**

timing analyzer. A software tool that estimates or measures the execution time of a computer program or portion of a computer program, either by summing the execution times of the instructions along specified paths or by inserting probes at specified points in the program and measuring the execution time between probes.

top-down. Pertaining to an activity that starts with the highest level component of a hierarchy and proceeds through progressively lower levels; for example, top-down design; top-down testing. *Contrast with:* **bottom-up.** *See also:* **critical piece first.**

total correctness. In proof of correctness, a designation indicating that a program's output assertions follow logically from its input assertions and processing steps, and that, in addition, the program terminates under all specified input conditions. *Contrast with:* **partial correctness.**

trace. (1) A record of the execution of a computer program, showing the sequence of instructions executed, the names and values of variables, or both. Types include execution trace, retrospective trace, subroutine trace, symbolic trace, variable trace.
(2) To produce a record as in (1).
(3) To establish a relationship between two or more products of the development process; for example, to establish the relationship between a given requirement and the design element that implements that requirement.

traceability. (1) The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another; for example, the degree to which the requirements and design of a given software component match. *See also:* **consistency.**
(2) The degree to which each element in a software development product establishes its reason for existing; for example, the degree to which each element in a bubble chart references the requirement that it satisfies.

traceability matrix. A matrix that records the relationship between two or more products of the development process; for example, a matrix that records the relationship between the requirements and the design of a given software component.

trailer. Identification or control information placed at the end of a file or message. *Contrast with:* **header (2).**

trailing decision. A loop control that is executed after the loop body. *Contrast with:* **leading decision.** *See also:* **UNTIL.**

transaction. In software engineering, a data element, control element, signal, event, or change of state that causes, triggers, or initiates an action or sequence of actions.

transaction analysis. A software development technique in which the structure of a system is derived from analyzing the transactions that the system is required to process. *Syn:* **transaction-centered design.** *See also:* **data structure-centered design; input-process-output; modular decomposition; object-oriented design; rapid prototyping; stepwise refinement; structured design; transform analysis.**

transaction-centered design. *See:* **transaction analysis.**

transaction matrix. A matrix that identifies possible requests for database access and relates each request to information categories or elements in the database.

transfer. (1) To send data from one place and receive it at another.
(2) To relinquish control by one process and assume it at another, either with expectation of return (*see* **call**) or without such expectation (*see* **jump**).

transform analysis. A software development technique in which the structure of a system is derived from analyzing the flow of data through the system and the transformations that must be performed on the data. *Syn:* **transformation analysis; transform-centered design.** *See also:* **data structure-centered design; input-process-output; modular decomposition; object-oriented design; rapid prototyping; stepwise refinement; structured design; transaction analysis.**

transform-centered design. *See:* **transform analysis.**

transformation analysis. *See:* **transform analysis.**

transient error. An error that occurs once, or at unpredictable intervals. *See also:* **intermittent fault; random failure.**

translator. A computer program that transforms a sequence of statements expressed in one language into an equivalent sequence of statements expressed in another language. *See also:* **assembler; compiler.**

transportability. *See:* **portability.**

trap. (1) A conditional jump to an exception or interrupt handling routine, often automatically activated by hardware, with the location from which the jump occurred recorded.
(2) To perform the operation in (1).

TRR. Acronym for **test readiness review.**

turnaround time. The elapsed time between the submission of a job to a batch processing system and the return of completed output. *See also:* **port-to-port time; response time; think time.**

turnkey. Pertaining to a hardware or software system delivered in a complete, operational state.

two-address instruction. A computer instruction that contains two address fields. For example, an instruction to add the contents of A to the contents of B. *Syn:* **double-operand instruction.** *Contrast with:* **one-address instruction; three-address instruction; four-address instruction; zero-address instruction.**

two-level address. An indirect address that specifies the storage location containing the address of the desired operand. *See also:* ***n*-level address.**

two-level encoding. A microprogramming technique in which different microoperations may be encoded identically into the same field of a microinstruction, and the one that is executed depends upon the value in another field internal or external to the microinstruction. *See also:* **bit steering; residual control.** *Contrast with:* **single-level encoding.**

two-plus-one address instruction. A computer instruction that contains three address fields, the third containing the address of the instruction to be executed next. For example, an instruction to add the contents of A to the contents of B, then execute the instruction at location C. *Contrast with:* **one-plus-one address instruction; three-plus-one address instruction; four-plus-one address instruction.**

type. *See:* **data type.**

UDF. Acronym for **unit development folder.** *See:* **software development file.**

unconditional branch.* *See:* **unconditional jump.**
*Deprecated.

unconditional jump. A jump that takes place regardless of execution conditions. *Contrast with:* **conditional jump.**

underflow exception. An exception that occurs when the result of an arithmetic operation is too small a fraction to be represented by the storage location designated to receive it. *See also:* **addressing exception; data exception; operation exception; overflow exception; protection exception.**

undirected graph. A graph (sense 2) in which no direction is implied in the internode connections. *Contrast with:* **directed graph.**

unit. (1) A separately testable element specified in the design of a computer software component.
(2) A logically separable part of a computer program.
(3) A software component that is not subdivided into other components.
(4) (IEEE Std 1008-1987 [10]) *See:* **test unit.**
Note: The terms "module," "component," and "unit" are often used interchangeably or defined to be sub-elements of one another in different ways depending upon the context. The relationship of these terms is not yet standardized.

unit development folder (UDF). *See:* **software development file.**

unit requirements documentation. (IEEE Std 1008-1987 [10]) Documentation that sets forth the functional, interface, performance, and design constraint requirements for a test unit.

unit testing. Testing of individual hardware or software units or groups of related units. *See also:* **component testing; integration testing; interface testing; system testing.**

unpack. To recover the original form of one or more data items from packed data. *Contrast with:* **pack.**

unstratified language. A language that can be used as its own metalanguage; for example, English, German. *Contrast with:* **stratified language.**

UNTIL. A single-entry, single-exit loop, in which the loop control is executed after the loop body. *Syn:* **post-tested iteration.** *Contrast with:* **closed loop; WHILE.** *See also:* **trailing decision.**

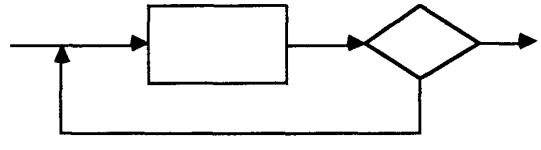

Fig 17
UNTIL Construct

unwind. In programming, to state explicitly and in full all of the instructions involved in multiple executions of a loop. *See also:* **straightline coding.**

up. Pertaining to a system or component that is operational and in service. Such a system is either busy or idle. *Contrast with:* **down.** *See also:* **busy; idle.**

up time. The period of time during which a system or component is operational and in service; that is, the sum of busy time and idle time. *Contrast with:* **down time.** *See also:* **busy time; idle time; mean time between failures; setup time.**

upward compatible. Pertaining to hardware or software that is compatible with a later or more complex version of itself; for example, a program that handles files created by a later version of itself. *Contrast with:* **downward compatible.**

upward compression. In software design, a form of demodularization in which a subordinate module is copied in-line into the body of a superordinate module. *Contrast with:* **lateral compression; downward compression.**

usability. The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.

user documentation. Documentation describing the way in which a system or component is to be used to obtain desired results. *See also:* **data input sheet; user manual.**

user friendly. Pertaining to a computer system, device, program, or document designed with ease of use as a primary objective.

user guide. *See:* **user manual.**

user interface. An interface that enables information to be passed between a human user and hardware or software components of a computer system.

user manual. A document that presents the information necessary to employ a system or component to obtain desired results. Typically described are system or component capabilities, limitations, options, permitted inputs, expected outputs, possible error messages, and special instructions. *Note:* A user manual is distinguished from an operator manual when a distinction is made between those who operate a computer system (mounting tapes, etc.) and those who use the system for its intended purpose. *Syn:* **user guide.** *See also:* **data input sheet; diagnostic manual; installation manual; operator manual; programmer manual; support manual; user manual.**

user state. *See:* **problem state.**

utility. A software tool designed to perform some frequently used support function. For example, a program to copy magnetic tapes.

utilization. In computer performance evaluation, a ratio representing the amount of time a system or component is busy divided by the time it is available. *See also:* **busy time; idle time ; up time.**

V&V. Acronym for **verification and validation.**

validation. The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements. *Contrast with:* **verification.**

value trace. *See:* **variable trace.**

variable. A quantity or data item whose value can change; for example, the variable Current_time. *Contrast with:* **constant.** *See also:* **global variable; local variable.**

variable trace. A record of the name and values of variables accessed or changed during the execution of a computer program. *Syn:* **data-flow trace; data trace; value trace.** *See also:* **execution trace; retrospective trace; subroutine trace; symbolic trace.**

variant. In fault tolerance, a version of a program resulting from the application of software diversity.

VDD. Acronym for **version description document.**

verification. (1) The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. *Contrast with:* **validation.**
(2) Formal proof of program correctness. *See:* **proof of correctness.**

verification and validation (V&V). The process of determining whether the requirements for a system or component are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with specified requirements. *See also:* **independent verification and validation.**

version. (1) An initial release or re-release of a computer software configuration item, associated with a complete compilation or recompilation of the computer software configuration item.
(2) An initial release or complete re-release of a document, as opposed to a revision resulting from issuing change pages to a previous release.
See also: **configuration control; version description document.**

version description document (VDD). A document that accompanies and identifies a given version of a system or component. Typical contents include an inventory of system or component parts, identification of changes incorporated into this version, and installation and operating information unique to the version described.

vertical microinstruction. A microinstruction that specifies one of a sequence of operations needed to carry out a machine language instruction. *Note:* Vertical microinstructions are relatively short, 12 to 24 bits, and are called "vertical" because a sequence of such instruction, normally listed vertically on a page, are required to carry out a single machine language instruction. *Contrast with:* **diagonal microinstruction; horizontal microinstruction.**

virtual address. In a virtual storage system, the address assigned to an auxiliary storage location to allow that location to be accessed as though it were part of main storage. *Contrast with:* **real address.**

virtual memory. *See:* **virtual storage.**

virtual storage. A storage allocation technique in which auxiliary storage can be addressed as though it were part of main storage. Portions of a user's program and data are placed in auxiliary storage, and the operating system automatically swaps them in and out of main storage as needed. *Syn:* **multilevel storage, virtual memory.** *Contrast with:* **real storage.** *See also:* **virtual address; paging (2).**

waiver. A written authorization to accept a configuration item or other designated item which, during production or after having been submitted for inspection, is found to depart from specified requirements, but is nevertheless considered suitable for use as is or after rework by an approved method. *See also:* **configuration control.** *Contrast with:* **deviation; engineering change.**

walk-through. A static analysis technique in which a designer or programmer leads members of the development team and other interested parties through a segment of documentation or code, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems.

waterfall model. A model of the software development process in which the constituent activities, typically a concept phase, requirements phase, design phase, implementation phase, test phase, and installation and checkout phase, are performed in that order, possibly with overlap but with little or no iteration. *Contrast with:* **incremental development; rapid prototyping; spiral model.**

wearout-failure period. The period in the life cycle of a system or component during which hardware failures occur at an increasing rate due to deterioration. *Contrast with:* **constant-failure period; early-failure period.** *See also:* **bathtub curve.**

WHILE. A single-entry, single-exit loop in which the loop control is executed before the loop body. *Syn:* **pre-tested iteration.** *Contrast with:* **closed loop; UNTIL.** *See also:* **leading decision.**

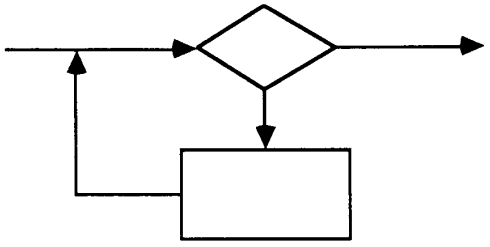

Fig 18
WHILE Construct

white box. *See:* **glass box.**

white-box testing. *See:* **structural testing.**

word. (1) A sequence of bits or characters that is stored, addressed, transmitted, and operated on as a unit within a given computer. *Syn:* **computer word.**
(2) An element of computer storage that can hold a sequence of bits or characters as in (1).
(3) A sequence of bits or characters that has meaning and is considered an entity in some language; for example, a reserved word in a computer language.
See also: **bit; byte.**

working area. *See:* **working space.**

working set. In the paging method of storage allocation, the set of pages that are most likely to be resident in main storage at any given point of a program's execution.

working space. That portion of main storage that is assigned to a computer program for temporary storage of data. *Syn:* **working area, working storage.**

working storage. *See:* **working space.**

workload. The mix of tasks typically run on a given computer system. Major characteristics include input/output requirements, amount and kinds of computation, and computer resources required. *See also:* **workload model.**

workload model. A model used in computer performance evaluation, depicting resource utilization and performance measures for anticipated or actual workloads in a computer system. *See also:* **system model.**

write. To record data in a storage device or on a data medium. *Contrast with:* **read.**

zero-address instruction. A computer instruction that contains no address fields. *Contrast with:* **one-address instruction; two-address instruction; three-address instruction; four-address instruction.**

4. Bibliography

[1] IEEE Std 610.1—see IEEE Std 1084-1986, IEEE Standard Glossary of Mathematics of Computing Terminology (ANSI) [11].[2]

[2] IEEE Std 610.5-1990, IEEE Standard Glossary of Data Management Terminology.

[3] IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology (ANSI).

[4] IEEE Std 828-1983, IEEE Standard for Software Configuration Management Plans (ANSI).

[5] IEEE Std 829-1983, IEEE Standard for Software Test Documentation (ANSI).

[6] IEEE Std 830-1984, IEEE Guide for Software Requirements Specifications (ANSI).

[7] IEEE Std 983-1986, IEEE Guide for Software Quality Assurance Planning (ANSI).

[8] IEEE Std 990-1987, IEEE Recommended Practice for Ada as a Program Design Language (ANSI).

[2]IEEE publications are available from the IEEE Service Center, Insitute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331.

[9] IEEE Std 1002-1987, IEEE Standard Taxonomy for Software Engineering Standards (ANSI).

[10] IEEE Std 1008-1987, IEEE Standard for Software Unit Testing (ANSI).

[11] IEEE Std 1084-1986, IEEE Standard Glossary of Mathematics of Computing Terminology (ANSI).

[12] IEEE Std 1012-1986, IEEE Standard for Software Verification and Validation Plans (ANSI).

[13] IEEE Std 1016-1987, IEEE Recommended Practice for Software Design Descriptions (ANSI).

When the following documents are completed, approved, and published by IEEE, they will become a part of the Bibliography of this standard:

[14] P610.7, Draft Standard Glossary of Computer Networking Terminology.

[15] P610.8, Draft Standard Glossary of Artificial Intelligence Terminology.

[16] P610.9, Draft Standard Glossary of Computer Security and Privacy Terminology.

[17] P610.13, Draft Standard Glossary of Computer Languages Terminology.

THIS PAGE WAS
BLANK IN THE ORIGINAL