Architecture-Driven Modernization Task Force Glossary of Definitions and Terms -- V3 24 JAN. 2006

Term	Definition	Source
4GL Statement	An informal (or popular) form of reference to a statement within a programming paradigm. See Programming Paradigm below.	NSD
Abstract Class	A class that has no direct instances, but is used as a base class from which subclasses are derived. These subclasses will add to its structure and behavior, typically by providing implementations for the methods described in the abstract class.	7
Abstract Data Type	A collection of data type and value definition and operations on those definitions which behaves as a primitive data type. The specifications of these types, values and operations are generally collected in one place, with the implementation hidden from the user. A data type whose nature is kept hidden, in such as way that only a predetermined collection of operations can operate on it.	5 \ 12
Abstract Semantic Graph	In *computer science* , an abstract semantic graph (ASG) is a *data structure* used in representing or deriving the *semantics* of an expression a formal language (for example, a *programming language*). An abstract semantic graph is a higher level abstraction than an abstract syntax tree(or AST), which is used to express the syntactic structure of an expression or program. An abstract semantic graph is typically constructed from an abstract syntax tree by a process of enrichment and abstraction. The enrichment can for example be the addition of back-pointers, *edges* from an identifier node (where a variable is being used) to a node representing the *declaration* of that variable. The abstraction can entail the removal of details which are relevant only in parsing, not for semantics.	28
Abstracted Action	Abstracted Action is an arbitrary collection of Primitive Actions. Dependencies between abstractedActions are “rolled up” dependencies between contained Primitive Actions. Abstracted Actions can be associated with more than one Callable Unit (?). Abstracted Actions are impontant to raise the level of abstraction in dealing with language-based elements below the procedure level.	16
Abstraction	Software that hides lower level details and provides a set of higher level functions.	4
Abstract Syntax Tree	In computer science, an abstract syntax tree (AST) is a *finite,* labeled, *directed tree,* where the *internal nodes* are labeled by operators, and the *leaf nodes* represent the operands of the node operators. Thus, the leaves have nullary operators, i.e., variables or constants. In computing, it is used in a *parser* as an intermediate between a *parse tree* and a *data structure,* the latter which is often used as a *compiler* or *interpreter* ’s internal *representation* of a *computer program* while it is being *optimized* and from which code generation is performed. The range of all possible such structures is described by the *abstract syntax* . An AST differs from a parse tree by omitting nodes and edges for syntax rules that do not affect the semantics of the program. The classic example of such an omission is grouping parentheses, since in an AST the grouping of operands is explicit in the tree structure.	28
Abstract Syntax Tree Meta-Model (ASTM)	ASTM is a meta-model and ADM package that allows for the interchange of system representations that have been abstracted into tree structures - a common format used by compilers and other tools. There are generic (GASTM) and specific (SASTM) versions of ASTM.	17

Action	Action is an abstract meta-class that represents language-independent executable statements within Callable Units. Actions are further subclassed in the model into: -Primitive Actions -Abstracted Actions	16
Action Semantics	A variation of denotational semantics where low-level details are hidden by use of modularized sets of operations and combinators.	13
Activation Record	A record containing all of the information associated with an activation or call or a procedure or function. This information includes: the return address of the caller, the procedures parameters and local variables, and the frame pointer or the caller.	11
Activation Stack	A stack of activation records, one for each active procedure call.	11
Actual Parameter	An actual parameter is the particular entity associated with the corresponding formal parameter in a subprogram call, entry call, or generic instantiation.	18
Actual Parameter or Argument	A parameter that appears in a call of the procedure or function.	11
Address	A location in physical memory.	4
AddressOf	A reference to an address.	NSD
ALC	See Assembler.	15
Alias	Data elements mapping to the same physical data with the same definition (same length, type, occurrences), but having a different data name. Aliases involve multiple definitions for the same record or element caused by poor Copy member utilization and by programmers creating hard coded record definitions. Other sources of alias creation and propagation include a lack of central or enforceable data administration standards.	15
Alter	A COBOL verb which changes the target of a Go To in a different paragraph This construct complicates the ease with which the program can be maintained because the destination of an Altered Go To is difficult to see. The Alter is considered a violation of structured coding principles.	15
API	Application Programming Interface (API) is a called (or optionally in-line) routine that expands or contracts data formats during application execution to reconcile expanded or un-expanded data formats between an application and a shared data store (i.e. an on-line file or shared data base).	15
Applicative Order Evaluation	An execution order in which the arguments in a function call are evaluated before the body of the function	6
Architecture	The way an application system is assembled that includes how data is stored and accessed (i.e. flat file, hierarchical, relational, etc.), how information is inputted and received by the user, how an application interfaces with other applications, the way functions are distributed or grouped across programs and a host of other issues. Three main aspects of architecture include business or functional architecture, data architecture and technical architecture.	15
Arcitecture-Driven Modernization (ADM)	ADM is the concept of modernizing existing systems with a focus on all aspects of the current systems architecture and the ability to transform current architectures to target architectures.	17
Argument	Parameter.	NSD
Array (1)	A composite type that consists of homogeneous components, indexed by discrete value.	18 \ 25 (p.39)

Array (2)	An array is a collection of elements of some fixed type, laid out in a k-dimensional rectangular structure.	18 \ 25 (p.39)
Array Reference	A reference to a composite type that consists of homogeneous components, indexed by discrete value.	NSD
Artifact	A well delineated portion of an existing system that can be indentified, extracted and represented in an external repository.	15
Ashcroft and Manna	E. Ashcroft and Z. Manna demonstrated a method of converting unstructured programs to structured ones. Note: In and of itself, it is not an elegant solution due to the fact that it is a switch driven solution.	15
Assembler	Assembler, also called assembler language code (ALC), is a low level computing language (second generation language) that is used to develop operating systems as well as application systems. IBM Assembler was widely used for these purposes.	15
Assigned To	"Assigned To" represents a way to show associations in KDM (e.g. Layer to Modules, Module to Run Time Entity)	16
Assignment	A special subclass of Primitive Action meta-class that represents assignment statements or system calls that are used to modify persistent data. Each Assignment is associated with one or more Data Unit. It “reads” from some Data Units and “writes” to other Data Units.	16
Asynchronous Message Passing	The message sending process allows messages to be buffered and the sending process may continue after the send is initiated; the receiving process will block if the message cue is empty.	9
Attribute (1)	A descriptive feature of an entity or relationship in the entity-relationship model.	1
Attribute (2)	The name of a column or a column header in a table, or, in relational-model terminology, the name of a domain used to define a relation.	1
Attribute Grammar	A set of equations associated to the grammar rules of a context-free grammar that define a collection of attributes associated to the terminal and nonterminals of the grammar. Attributes equations are written in purely functional form (i.e., without side effects) and may be solved fro the actual attribute values by different kinds of traversals of the parse or syntax tree. Alternatively, attribute values may be computed by replacing the attribute equations with equivalent side-effect-generating code using separate data structures as the symbol table.	10
Aux Build Component	A placeholder for any additional types of things that a particular platform may use during the KDM build. This can be subclassed further.	16
Axiomatic Semantics	The meaning of a program as a property or specification in logic.	13
Back End	The part of a compiler that depends only on the target language and is independent of the source language. The backend receives the intermediate code produced by the front end and translates it into the target language.	10
Backtracking	A manner to handle (nondeterministic situations by considering one choice at a time and storing information which is necessary to restore a given state of the computation. PROLOG interpreters often use backtracking to implement nondeterministic situations.	8
Backus-Naur Form (BNF)	A notation for context-free grammar rules first used in the Algol60 report to describe syntax. It comprises two metasymbols, usually written --> or ::= and |.	10
Base Type	(1) the C++ term for any type from which another type is derived via subtyping.	17

Behavior Usage	Behavior Usage is a subclass of Design Relation meta-class that represents usages in the executable statements within Callable Units: -Callable Unit uses zero or more Callable Units (for example, procedure call). -Callable Unit uses zero or more Data Units (for example read to a global variable, write to a local variable, read from a file, etc.) Usually, Behavior Usage corresponds to link-time and run-time relations.	16
Binary Component	Source Components "compile" into Binary Components. Binary Components represent any form of intermediate object files, byte-code, etc. Binary Components need to be further "linked" into Executables. Some software platforms may have only Binary Components, or only Executables. Another form of Binary Components is a Library.	16
Binary Expression	An expression that computes a relationship between two things.	NSD
Binary Relationship	Any relationship between two things.	18
Binding (1)	The act of associating attributes to a name is often referred to as binding. Most languages allow only static binding (compile time) Some languages, such as SNOBOL, allow dynamic binding, or binding of attributges while the program is running (at run time).	25 (p.37)
Binding (2)	A connection between an abstraction used in the languages and a data object as it exists in the computer hardware. The sage, establishment, and number of these bindings characterize the various imperative languages and affect their ease of use and performance.	5
Block	A block introduces a (possibly named) sequence of statements, optionally preceded by a declarative part.	18
Block Statement	A statement that introduces a (possibly named) sequence of statements, optionally preceded by a declarative part.	NSD
Bohm and Jacopini	C. Bohm and G. Jacopini defined the fundamental constructs of structured programs: sequence, selection and iteration.	15
Bottom Up (1)	The derivation of requirements, analysis and design models resulting from current systems and application area analysis through the use of automated recovery tools, techniques, interviews and humanistic interpretation of subsequent findings.	15
Bottom-Up (2)	A parsing algorithm that constructs the parse tree from the leaves to the root. Bottom-up algorithms include LR and LALR parsers, such as those produced by Yacc.	10
Branch Statement	A statement that defines a program point at which the control flow has two or more alternatives.	20
Breadth First	A method for traversing trees in which all of the children of a node are considered simultaneously. OR-Parallel PROLOG interpreters use breadth-first traversal.	8
Break Statement	The break statement causes program control to proceed with the first statement after the switch structure.	21
Bridge	A routine that expands or contracts batch data files to reconcile data format differences between expanded and un-expanded data stores in a Year 2000 project.	15
Build Description	Build Description is a set of platform-specific rules which describe the transformation from source files into executables, and then also the deployment of executables onto the run-time entities.	16

Build Package	A special subclass of Container meta-class that specifies various artifacts of existing software which are related with each other throughout the build process. By the build process we understand the series of transformations from source files to some form of executable code. Build Package consists of: -Source Components -Binary Components -Libraries -Executables -Resources -Configuration Descriptions -AuxBuild Components. Each Build Package describes a platform-specific way of repeatable mappings from source files to run-time entities.	16
Business Area	An organization specific, functionally bounded segment of an organization that, in IT terms, is supported by multiple application systems.	15
Business Rule	A combination of conditional and imperative logic that changes the state of an object or data element - depending on the methodology being used.	15
C	A programming language typically used for client/server system development.	15
C++	An object oriented programming language typically used for client/server system development.	15
Call (1)	To send a message	17
Call (2)	To evaluate a post fix expression identifying an object and associated function followed by parentheses containing a possibly empty, comma-separated list of expressions, which constitute the actual arguments to the function (C++),	17
Call (3)	To invoke the method function of a method object.	17
Call (4)	To apply a certain feature to a certain object, possibly with arguments. A call has three components: - the target of the call, an expression whose value is attached to the object; - the feature of the call, which must be a feature of the base class of the object's type; - an actual argument list.	17
Call (5)	This is a special subclass of the Primitive Action meta-class that represents procedure calls, etc. Call action is associated with zero or more Data Units. It also "reads" from some and "writes" to others. A Call action "writes" to Data Units through return or through "output" parameters. This class can be further subclassed to represent more language-specific forms.	16
Callable Unit	This is a subclass of Design Entity meta-class that represents a block of instruction statements. For example, A Callable Unit can represent a procedure in C or a paragraph in COBOL.	16
Call-by-Address	See Call-by-reference	NSD
Call-by-Address Parameter	A method of parameter transmission whereby the address of the actual parameter is copied to the formal parameter at the time of the call. The formal parameter is effectively a constant pointer variable. Any reference to the formal parameter is treated as reference to the actual parameter. Also known as a call-by-reference or call-by-location.	11
Call-by-Reference (1)	When an argument is passed by reference, the caller gives the called method the ability to access the caller's data directly and to modify that data if the called method so chooses. Pass-by-reference improves performance, because it eliminates the overhead of copying large amounts of data.	17 / 21
Call-by-Reference (2)	Any message passing in which a reference to (e.g., the address of) each argument is paased rather than its value.	17 / 21
Call-By-Result Parameter	A method of parameter transmission whereby the value of the formal parameter is copied back to the actual parameter at the time of the return. Prior to executing the return statement there is no correspondence between the actual and formal parameters.	11

Call-by-Value (1)	When an argument is passed by value, a copy of the argument's value is made and passed to the called method.	17 / 21
Call-by-Value (2)	Any message passing in which a copy of the value of each argument is passed rather than a reference (e.g., its address).	17 / 21
Call-By-Value Parameter	A method of parameter transmission whereby the value of the corresponding actual parameter is copied to the formal parameter at the time of the call. Thereafter, there is no correspondence between the actual and formal parameters. In particular, changes to the formal parameter have no effect on the actual parameter.	11
Call-By-Value-Result Parameter	A method of parameter transmission which combines call-by-value and call-by-result.	11
CASE	Acronym for "Computer Aided Software Engineering". This is the process where development and modification of software is accomplished with software designed for this purpose.	15
Case Statement	One of a series of case labels in a switch statement concluded by an optional default case.	21
Cast Expression	A special operation that handles type conversion.	21 (p. 168)
Catalog	A catalog is a named collection of schemas. The name of a catalog is used to qualify the names of the schemas in that catalog.	2
Century Midpoint	An application parameter representing the 2-digit year upon which the century date will toggle in a Year 2000 procedural workaround project. Any 2-digit year higher than the century midpoint would be assumed to have a "19" in the century field, and those lower than or equal to the century midpoint would be assumed to have a "20" in the century field. (This is common in many older systems).	15
Channel	The data structure, which may be realized in hardware, over which processes send messages.	9
CICS	Commonly used IBM telecommunications management product.	15
Class	A set of objects in the object-oriented model that contain the same types of values and the same methods; also, a type definition for objects. A description of the data and behavior common to a collection of objects. Objects are instances of classes.	1 \ 7
Class Unit	This is a subclass that represents classes in object-oriented languages. In this model, a class is a named collection of Callable Units (methods) and MemberUnits (data).	16
Clause	A general normal form for expressing predicate calculus formulas. It is a disjunction of literals (P1 or P2 or …) whose arguments are terms. The terms are usually introduced by eliminating existential quantifiers.	8
Client	A process that requests services by sending messages to server processes.	4
Client-Server	The software architecture in which clients are able to request services of processes executing on remote machines.	9
COBOL	Common Business Oriented Language. COBOL is the leading programming language used in the business world.	15
Code Flaws	See Defects	15
Code Replication	See Replication.	15
Code Splitting	The process of dividing a program into two or more smaller, separately compilable, more manageable modules. Also called code "slicing".	15
Code Stabilization	A refactoring concept that includes Code Flaw Analysis & Removal, Code Restructuring and Design Review and Improvement tasks Its goal is to improve the cosmetics, packaging and design aspects of individual source programs without changing their functionality.	15

Cohesion	Property of a modular software system which measures the logical coherence of a module.	14
Collection	The entire set of allocated objects of an access type.	18
Collection Type (1)	Any type of collection objects.	17
Collection Type (2)	Any instantiation of a collection type generator.	17
Column	A column is a multiset of values that may vary over time. All values of the same column are of the same data type and are values in the same table. A value of a column is the smallest unit of data that can be selected from a table or updated in a table.	2
Command Interpreter	A program (usually not in the kernel) that interprets user requests and starts computations to fulfill those requests.	4
Commands	Instructions in a job-control language.	4
Communicate Over	This is an association between one or many Run Time Entities	16
Communication Mechanism	This is a meta-class that specifies a particular way of inter-process communication for the given software platform, for example files, sockets, semaphores, etc. Communication Path is a particular (named) instance of a Communication Mechanism.	16
Communication Path	This is a subclass of relation between one or more Run Time Entities that represents a run-time communication between them. A Communication Path involves a certain Communication Mechanism which is provided by the software platform. Note: a regular Relation is usually one-to-one, but a Communication Path is many-to-many.	16
Compile	Association from one or more Source Compoennts into Binary Components or Libraries.	16
Compilation Unit	A program unit presented for compilation as an independent text. It is preceded by a context specification, naming the other compilation units on which it depends. A compilation unit may be the specification or body of a subprogram or package, including generic units or subunits.	18
Complexity	A category of measuring the effort required to maintain or enhance a program. This could involve the data representations or the way a program was constructed. Also implies difficulty levels in understanding and modifying applications at the system level.	15
Compliance Test	An operation that compares relevant attributes of a product with applicable standard requirements for determining the achievement of the level of quality required.	3
Component	A clearly delineated instance or occurrence of a physical, logical or externally defined item or work product within or associated with an application system. Components are delineated by functional commonality.	15
Compute Bound	A process that performs little input/output but needs significant execution time.	4
Concatenation	The process of combining two or more procedures which are executed sequentially.	15
Concept	A Concept is a set of Design Entities. A Concept should be a subclass of a Container. Concepts are very important for understanding existing software. They raise the level of abstraction by hiding implementation detailes.	16
Conceptual Model	This is a meta-class that represents the so-called Conceptual Views of software architecture. In our meta-model, Conceptual Model is defined as a set of: -Concepts -Conceptual Relations	16
Conceptual Relation	This is a subclass of Relation. Conceptual Relation is an “abstracted” relation, the rolled-up relations between entities in corresponding Concepts. Note: this should be made more explicit in the model.	16
Concurrency Control	Means to mediate conflicting needs of simultaneously executing threads.	4

Condition (1)	This is a special subclass of the Primitive Action meta-class that represents conditional logic, for example, if-then-else statements. Condition uses one or more Data Units. This class can be further subclassed to represent more language-specific forms.	16
Condition (2)	Any boolean (or enumeration-valued) expression involving the values of one or more properties. (b) any Boolean function of object values that is valid over an interval of time.	16
Condition (3)	Any logical statement about the current state of an object, the current state of the system environment, the existence or absence of an object, or the existence or absence of relationships among objects.	16
Condition Variables	A variable used within a monitor to delay an executing process.	9
Conditional Expression	A conditional expression is an operation or one or more operands that evaluates to true or false.	NSD
Configuration Description	Configuration Description is a set of platform-specific deployment rules which describe the mapping of executables onto the run-time entities, as well as the allocation of resources to the run-time entities. Note: the association between executable and Configuration Description should be probably renamed to “use during deployment”, rather than “use at runtime”. It probably should also include resources and run-time entities.	16
Constraint	A constraint is a rule that defines valid states data in an SQL database by constraining the values of data, or the relationships between values of data in tables.	2
Constraint Logic Programming Languages	PROLOG-like languages in which unification is replaced or complemented by constraint solving in various domains.	8
Constraints	Special predicates whose satisfiability can be established for various domains. Unification can be viewed as equality constraints in the domain of trees.	8
Construct	A source code instruction, or sequence of instructions, that is patterned in a known format and that executes in a predictable manner.	15
Constructor	An operation associated with a class that creates and/or initializes new instances of the class	7
Consume (1)	Platform Element consumes a Platform Resource. Note: this should be spelled “Consumes”	16
Consume (2)	An association between Communication Mechanism and zero or more Platform Elements.	16
Contain	This is an association between a Layer and zero or more other Layers, creating a hierarchy of Layers within an ADM Meta-model.	16
Container	This meta-class is defined at the Container Models section of an ADM Meta-model.	16
Context of Use	The users, goals, tasks, equipment (hardware, software, and materials), and the physical and social environment in which a product is used [ISO 9241-11 1996]	3
Context Switching	The action of directing the hardware to execute in a different context (kernel or process) from the current context.	4
Continue Statement	The continue statement, when executed in a while, for or do/while structure skips the remaining statements in the loop body and proceeds with the next iteration of the loop.	21 (p. 219)
Contravariant	A type that varies in the inverse direction from one of its parts with respect to subtyping. The main example is the contravariance of function types in their domain. For example, assume A<: B and vary X from A to B in X--> C; we obtain A--> C :> B--> C. Thus X--> C varies in the verse direction of X.	12

Control Logic	Instructions in a program or procedure that direct the sequence and conditions of execution of other instructions or programs.	15
Control Structures	Structures of statements that alter the strict sequential ordering in an imperative program, presenting alternatives to sequential control. Control structures can be conditional, iterative or unconstrained.	5
Control Transfer	In a computer source program, a change of control flow to a statement other than the next one, or from a calling program to an external program or subroutine.	15
Control Verb	A computer source program reserved word (verb), such as IF, Go To, Perform, Alter or Perform Thru, used in a program to transfer control explicitly.	15
Correctness	Software is (functionally) correct if it behaves according to the specification of the functions it should provide.	14
Count	A metric referring to the number of occurrences of a physical component, construct or combination of construct types within an application program, system, division, enterprise, etc.	15
Count Attribute (of an Entity Type)	An attribute of an entity that represents a quantitative value (such as an inventory item having an entity type called maximum value which is set, for example, 1000).	15
Coupling	Property of a modular software system which measures the amount of mutual dependence among modules.	14
Covariant	A type that varies in the same direction as one of its parts with respect to subtyping. For example, assume A<: B and vary X from A to B in D--> X; we obtain D--> A<: D--> B. Thus, D--> X varies in the same direction as X.	12
Criterion	A required level of quality measure against which attributes of an object (e.g., a product) or a process (e.g. the design cycle) are judged to evaluate the level of quality achieved.	3
Critical Regions	A section of code which must appear to be executed indivisibly	9
Cut	An annotation used in PROLOG programs to bypass certain nondeterministic computations.	8
Cyclomatic Complexity	A term used in graph theory to describe the number of basic paths through a graph. This is one of the McCabe Metrics.	15
Data Definition Standardization	Refactoring task that includes Data Name Rationalization, Field & Record Size Expansion, Literal Externalization, Data Definition Migration and Physical Data Upgrade. Data or data definition related tasks that fall into the Positioning stage will continue to be defined under this major task heading.	15
Data Flow Diagram	A diagram, typically automated using various tools, that depicts data flowing from one process to another. This model type is used within ADM to depict major data flows and related activities within a system or across the enterprise.	15
Data Name Rationalization	The process by which data element definitions are modified so that each element retains the same name and characteristics throughout an application system. This has a tendency to dramatically reduce the actual number of record groups and physical data names within a system by creating reusable Copy or Include code blocks for a given record, segment or table definition.	15
Database	A collection of files for storing related information.	4
Data Flow Dependency	Actions are related to each other through Data Flow Dependencies.	16
Data Group	A group of elements (such as a COBOL 01 level) and / or I/O records that have been grouped together based on common lengths, underlying record structure and / or verb transfer usage (i.e. Move, Read Into, etc.) Includes tables / records and segments.	15

Data Type Unit	This is a subclass that represents a data type. This can be further subclassed for a particular programming language. Note: We did not consider composite types. There should be an association between Data Type Units.	16
Data Unit	This is a subclass that represents data. The purpose of this meta-class it to define an “abstract memory location”, with read and write operations. This is further subclassed into the following: -Dynamic Data -Local Data -Global Data -Persistent Data -Member Data. Further subclasses can be added as an extension of the meta-model.	16
Dead Code	Portion of a program that is never executed because it cannot be reached by the static flow of control logic (syntactically dead). (Also see Unentered Procedure or Unexecutable Statement.) Logic is also dead based on certain data values that are never being true (semantically dead).	15
Dead Program	An executable program that is never executed due to system control objects (JCL, on-line control tables, Call verbs, etc.) never invoking that subroutine or load module within the production environment.	15
Deadlock	A state in which processes are waiting for events which can never occur, i.e., the processes cannot progress.	9
Declaration (1)	Any line of code that introduces one or more names into a program and specifies the types of the names	17, 18
Declaration (2)	Any language construct that associates a name with a view of an entity.	17, 18
Declaration (3)	Associates an identifier with a declared entity, including objects, types, subprograms, tasks, renamed entities, numbers, subtypes, packages, exceptions, and generic units.	17, 18
Declaration Unit	Declaration Unit represents any additional constructs (i.e. non-executable constructs, not related to data storage). The reason for including them into Design Entities, is that such constructs may use other Design Entities, in particular Data Type Units and Signatures.	16
Defect (1)	An unintended attribute that impairs the efficient usage of a product.	3
Defect (2)	Messages or warnings denoting a known condition, situation, or problem in a computer program. Defects include violations of structured source coding constructs that may, in fact, be acceptable to a given compiler, but which are unstable in nature. Defects make a program more difficult to maintain.	15
Default Statement	In a switch/case statement, the default case is the one that will be chosen whenever the value does not match any of the other cases. See Switch.	24
Definition	The specification of the implementation of something	3
Deliverables	The output results delivered from a given ADM step and / or task. Deliverables may include models, metrics, documentation, source code, narrative summary, forms and executable systems.	15
Denotational Semantics	The meaning of a program as a compositional definition of a mathematical function from the program's input data to its output data.	13
Density	Attibute of the KDM meta-class Relation, that specifies how many Design Relations (or other primitive relations) have been rolled-up (or abstracted) to produce this relation. A primitive relation has density one.	16
Depth First	A method for traversing trees in which the leftmost branches are considered first. Most sequential PROLOG interpreters use depth-first traversal.	8
Derivation	A tree of judgments obtained by applying the rules of a type system.	12
Derived Type	A new data type constructed by copying a type that already exists. The resulting new type is distinct and not identified as being copied from the existing type, though operations on the old type are automatically inherited in the new type.	5

Design Entity (1)	A special subclass of the meta-class Entity that specifies primitive elements, that are defined by a particular programming language. DesignEntities are further subclassed into several important subclasses in a separate diagram. Since Design Entities are located in source files, they have a location attribute.	16
Design Entity (2)	This meta-class was defined in the Conceptual Model. This diagram defines important language-independent subclasses of Design Entity.	16
Design Relation	Design Relation is a KDM subclass of Relation meta-class that represents primary language-based relations between Design Entites. Design Relations become the basis for the “abstracted” relations between various containers. There are the following three subclass of Design Relation: -Behavior Usage -Type Usage -Initialization	16
Design-Use Cycle	The course of developmental changes through which a product passes from its conception, during its usage, up to the redesign on the termination of its use.	3
Destruction	A special KDM subclass of the Primitive Action meta-class that represents destruction of data. This class can be further subclassed to represent more language-specific forms.	16
Development	The process of planning, analyzing, designing and constructing software systems to satisfy new or redefined applications requirements. This may be driven by iterative prototyping, spiral design, waterfall design, agile methods or other means.	15
Device Driver	An operating-system module (usually in the kernel) that deals directly with a device.	4
Device Interface	The means by which devices are controlled.	4
Diagnostics	See Defects	15
Dialogue	A process in the course of which the user, to perform a given task, inputs data in one or more dialogue steps and receives for each step feedback with regard to the processing of the data concerned.	3
Dimension	The number of independent interpreted input variables over which a domain is defined.	20
Direct Manipulation	A dialogue technique by which the user directly acts on objects on the screen, e.g. by pointing at them, moving them, and/or changing their physical characteristics (or values) via the use of an input devise [ISO 9241-16 1996].	3
Disambiguating Rule	A rule stated separately from the rules of a context fee grammar that specifies the correct choice of syntax tree structure when more than one structure is possible.	10
Distributed Processing	The use of multiple processors which communicate via a network.	9
Don't Care Nondeterminism	The arbitrary choice of one among multiple possible continuations for a computation.	8
Don't Know Nondeterminism	Situations in which there are equally valid choices in pursuing a computation.	8
Dual Purpose File	An object, such as a data element or record, that is used within the system in which it was created and passed on or received from a related system.	15
Dynamic Binding	Binding performed at run time. In OOP This typically refers to the associating of a particular class with a name, so that the method to be invoked in response to a message can be determined by the class to which it belongs at run time.	7
Dynamic Checking	A collection of run time tests aimed at detecting and preventing forbidden errors.	12

Dynamically Checked Language	A language where good behavior is enforced during execution.	12
Dynamic Data	This is a subclass of Data Unit that represents dynamically allocated heap data.	16
Effectiveness	The accuracy and completeness with which users achieve specified goals [ISO 9241-11 1996]	3
Embedded SQL	Embedded SQL is a technique of placing SQL statements in programs written in one or several more conventional programming languages, thus producing the effect of a conventional program "calling" procedures written in SQL alone.	2
Encyclopedia	A knowledge base of information possibly stored in a proprietary format, that is used by tools as a means of storing and linking various development models. This differs from a repository in terms of how open the meta-model may be to extensive modification and refinement.	15
Enterprise Redevelopment Planning	Collection of ADM tasks for assessing enterprise wide, architecture transition requirements across business area boundaries. Basic techniques also include developing a baseline for IT asset management, auditing current project activities and segmenting the enterprise for large-scale, cross-functional projects. Main deliverable includes a cross-functional transition strategy.	15
Entity (1)	A distinguishable item in the real-world enterprise being modeled by a database schema.	1
Entity (2)	Fundamental meta-class, describing any element of the existing system. Entities are further subclassed into logical entities, execution entities, build entities, design entities, data entities (tbd), conceptual entities and scenarios	16
Entrance Criteria	Recommended prerequisite external activities for ADM specific tasks.	15
Entry	Used for communication between tasks. Externally, an entry is called just as a subprogram is called. Its internal behavior is determined by one or more accept statements which specify the actions to be performed when the entry is called.	18
Enumeration	Any developer-defined type whose instances are named literal objects.	17
Enumeration Literal	Instances of an enumeration type expressed as named literal objects.	NSD
Enumeration Reference	A reference to an enumeration type or an enumeration literal.	NSD
Enumeration Type	A discrete type whose values are given explicitly in the type declaration. These values may be either identifiers or character literals, which are considered enumeration literals.	18
Environment	A special subclass of the meta-class Entity that is used to create hierarchies of containers. Environment is used as a source or sink of the relations that exist between containers and that are the rolled-up representations of cross-container relations between entities within the corresponding containers.	16
Essential Complexity	A term used in graph theory which represents the count of unstructured constructs in a control flow graph. This is one of the McCabe Metrics.	15
Evaluation	All activities to assess the quality of an object (e.g. a product) or a process (e.g. the design cycle) in relation to a criterion, which defines a required level of a quality measure.	3
Event	A happening in the business that triggers a business rule to be invoked. Also see Event Modeling.	15
Event Horizon	The amount of time that an application has left to begin an ADM task to prevent that system from reaching a certain failure point.	15

Event Modeling	A series of events, depicted in a diagrammatic format, that mirror real life activity cycles and can be interpreted by certain development tools in order to create operational systems.	15
Event Subtyping	A more detailed or specific decomposition of an event.	15
Evolvability	Ease of software evolution	14
Exception	An event that causes suspension of normal program execution. Bringing an exception to attention is called raising an exception. An exception handler is a piece of program txt specifying a response to the exception. Execution of such a program text is called handling the exception.	18
Executable	Binary Components are linked together into Executables.	16
Expansion	The removal of a COPY statement from a program during restructuring process and the placement of the statements from the COPY member into the program without reference to their origin. The increase in physical or logical size of a record or element definition or the corresponding physical data that is referenced by that element or record.	15
Explicitly Typed Language	A typed langauge where types are part of the syntax.	12
Expression	Part of a program that computes a value.	18
Expression Statement	A statement that computes a value.	NSD
Extended BNF (EBNF)	Adds bracketing metasymbols […] and {…} to BNF to indicate optional and repeated structures, respectively. (These can also be written as (…)? And (…)* to remain consistent with standard regular expression notation.)	10
Extensible	Adj. Describing software that is easy to modify to implement new or changed requirements, especialy without modification to existing moduels. Examples: Classes may eaily be extended via subclasses.	17
Extensible Expression	An class denoting an expression that may be extended via subclassing to denote specialized forms of expressions.	NSD
Extensible Statement	An class denoting an statement that may be extended via subclassing to denote specialized forms of statements.	NSD
External System File	A file or data store received from or passed to an interface system.	15
External System Objects	Physical or logical descriptions of an application system including models, documentation, job scheduling information, a data dictionary or repository. This information is initially gathered during the environmental analysis task.	15
Fairness	Processes will eventually be able to progress, I.e., enter their critical regions.	9
Fall Through	Implicit transfer of control from one paragraph to the succeeding paragraph or module in a source program.	15
Fault	A missing attribute that impairs the effective usage of a product.	3
File (1)	A named long-term repository for data - commonly called "persistent data.	4
File (2)	This is a special KDM subclass of the meta-class Entity that corresponds to source files, as well as other files (for example, binary files, resources, etc.). Can be further subclassed to express particular file types, that are relevant to the software platform.	16
First-Class Object	An object that can be stored in data structures, passed as arguments, and returned as the result of function calls. In functional languages, functions are first-class objects.	6
First-Order Type System	One that does not include quantification over type variables.	12
Fixed-Point Semantics	A denotational semantics where the meaning of a repetitive structure, such as a loop or recursive procedure, is expressed as the smallest mathematical function that satisfies a recursively defined equation.	13

Folding	The process of eliminating a separately named procedure by moving the code it contains into the procedures which invokes it.	15
Forbidden Error	The occurrence of one of a predetermined class of execution errors; typically the improper application of an operation to a value.	12
Foreign Key	A set of attributes in a relation schema that serves as a primary key for another relation schema.	1
Formal Declaration	See Formal parameter or argurments	NSD
Formal Parameter or Arguments	A parameter name that appears in the declaration or header of a procedure or function.	11
Forward Engineering	The traditional process of moving high-level abstractions and logical, implementation independent design to the physical implementation of a system. Source IEEE.	15
Frame Pointer	A register that normally points to the base or beginning of the activation record on the top of the activation stack.	11
Front End	The part of a compiler that depends only on the source language and is independent of the target language. The front end translates and analyzes the source program.	10
FTP	The file transfer protocol service	4
Function	A group of business activities, which together completely support one aspect of furthering the mission of the enterprise (source Information Engineering).	15
Functional Assessment	A major ADM task that evaluates current system backlog requirements, builds and compares bottom up data and functional models with target or top down models, defines the intersection of current data and functions, re-documents existing systems and determines the role of current systems under a strategic replacement initiative.	15
Functional Dependency	A rule stating that a given values for some set of attributes, the value for some other set of attributes is uniquely determined. X functionally determines Y if whenever two types in a relation have the same value on X, they must also have the same value on Y.	1
Generate	Association from one or more Source Components onto zero or more Source Components. This is useful to describe various source code generators, other than standard compilers.	16
Good Behavior	Same as being well behaved.	12
Gopher	A network service that connects information providers to their users.	4
Global	Data available to more than one unit.	25
Global Declaration	A declaration available to more than one unit. See Global.	NSD
Graph Theory	A way to represent the flow of control through computer programs. McCabe complexity metrics are computed by counting nodes and connectors within these graphs. Graphs can also be used as a basis to restructure programs through simplification of complex expressions of the control flow.	15
Graphical User Interfaces	Interactive programs that make use of a graphical display and a mouse.	4
Guard [condition]	Any condition that must be true (or have the proper enumeration value) for a trigger to cause the associated transition to fire.	17
Guarded Transition	Any statement transition that occurs only if the trigger fires while its associated guard condition evaluates to true (or to the enumeration value associated with the transition)	17

Guarded Statement	A statement in a language that introduces a guard condition. E.g. The 'Try' statement in java, C++ and C#. The On statement in Visual Basic.	NSD
Hardware Resource	This is a meta-class that describes the actual hardware resources to which the software resources, such as Platform Resource and Communication Mechanisms can be mapped. For example, stack spaces of several processes are mapped to the physical memory. Several processes share the same CPU, etc.	16
Heap	The portion of memory assigned to an executing program to use for dynamic memory allocation	11
Herbrand Universe	The set of all terms that can be constructed by combining the terms and constants which appear in a logic formula.	8
Hierarchical	A family of coding or design methodologies where program control blocks always proceed from a controlling procedure to an invoked procedure and back. Use of these methods results in systems that are easier to understand and maintain. Also refers to data structures that support a child / parent relationship between segments or record types. IMS is a hierarchical data structure.	15
Higher Order Function	A function that takes another function as a parameter or returns a function as its result.	6
Horn Clause	A clause containing (at most) one positive literal. The term definite clause is used to denote a clause with exactly one positive literal. PROLOG programs can be viewed as a set of definite clauses in which the positive literal is the head of the rule and the negative literals constitute the body or tail of the rule.	8
I/O Record Data Group	A data group that contains at least one definition used in an I/O (input / output) transaction.	15
Identifier (1)	The name bound to an abstraction.	5
Identifier (2)	One of the basic lexical elements of the langauge. An identifier is used as the name of an entity or as a reserved word.	18
Identifier Reference	A reference to an identifier.	NSD
Ill Typed	A program fragment that does not comply with the rules of a given type system.	12
Implicitly Typed Language	A typed language where types are not part of the syntax.	12
Import	Association from zero or more Source Components to zero or more other Source Components. This is useful to represent include relations between .C and .H files in C/C++, or import relations between packages in Java.	16
IMS	Commonly used, IBM hierarchical data base product.	15
Include Statement	A statement in a language (i.e.such as C or C++) which lexically introduces one program unit into the body of another program unit.	NSD
Include Element	The program unit referenced in an include statement for inclusion.	NSD
Incremental Development	A software process which proceeds by producing progressively larger increments of the desired product. At each stage, the complete increments form a subset of the final product. Each increment provides additional functionaltiy and brings the currently available subset closer to the desired one.	14
Infinite Trees	Trees that can be united by special unification algorithms which bypass the occur-check. These trees constitute a new domain, different from that of usual PROLOG trees.	8
Information Engineering (IE)	An interlocking set of formal techniques in which enterprise models, data models and process models are built up in a comprehensive knowledge base and are used to create and maintain software systems.	15

Inheritance	A relationship among classes, wherein one class shares the structure or behavior defined in an is-a hierarch. Subclasses are said to inherit both the data and methods from one or more generalized superclasses; the subclass typically specializes its superclasses by adding to its state data and by redefining its behavior.	7
Inherited Attribute	An attribute whose value depends on attribute values at syntax tree nodes that are not descendants. A nonsynthesized attribute.	10
Inherits	To obtain the declarations and definition of features via inheritance. Example: Child classes inherity features from their parent classes.	17
Initialization (1)	Initialization is a KDM subclass of Design Relation that represents instantiation of Data Units.	16
Initialization (2)	A special KDM subclass of the Primitive Action meta-class that represents instantiation of Data Type and Class Units and initialization of data.	16
Input/Output	A resource; ability to interact with peripheral devices.	4
Inputs/Output Bound	A process that spends most of its time waiting for input/output.	4
Instance	A specific example that conforms to a description of a class. An instance of a class is an object.	7
Instance Variable	The data items that are associated with (and are local to) each instance of a class	7
Instance Variables	Attributes within objects	1
Instantiable Data	This is an abstract meta-class that represents Data Units which can be instantiated: -Local Data -Dynamic Data -Class Data -Persistent Data Global Data is instantiated statically, when the Run Time entity is loaded.	16
Instantiate	Association between an Executable and one or more Run Time Entities.	16
Integrated Application	An application that agrees on data formats with other applications so they can use each other's outputs.	4
Interactive Multiprogramming	Multiprogramming in which each user deals interactively with the computer.	4
Interface	This meta-class corresponds to “abstracted” relations between containers and plays an important role in software architectures. An interface specifies a set of entities that has relations to/from other modules or a user environment.	16
Interface System	Any system that receives data from or sends data to systems involved in an assessment - but that is not involved in the assessment itself.	15
Interim Plan	A deliverable from a modernization assessment task that details how existing systems are to evolve under the current architecture. The term "interim" implies that various refactoring tasks and functional upgrades are to be applied under the current technical, functional and data architecture. Interim does not necessarily imply a brief time span.	15
Internal System File	A file used only within the system to which it is defined.	15
Interoperability	Ability of a software system to coexist and cooperate with other systems.	14
Inventory/Analysis	A series of modernization tasks for determining the state of current applications, identifying future information requirements and articulating a plan to meet interim user needs and long term strategic requirements. Also serves to re-document existing applications.	15
Item	A grammar rule choice with a distinguished position, indicating that a parse has reached that position in attempting to recognize the rule. Sets of items are used by a bottom-up parser to record a state reached during a parse. Items may have 0 or more tokens of lookahead attached to them. LR(0) items contain on lookahead, while LR(2) items contain one token of lookahead.	10

Iteration	Iteration is a repetition structure (such as for, while, or do/while); that terminates when the loop-continuation condition fails. Iteration modifies a counter until the counter assumes a value that makes the loop-continuation condition fail. An infinite loop occurs with iteration if the loop continuation step never becomes false.	21 (p. 286)
Job	A set of computational steps packaged to be run as a unit.	4
Job-Control language	A way of specifying the resource requirements of various steps in a job.	4
Judgment	A formal assertion relating entities such as terms, types, and environments. Type systems prescribe how to produce valid judgments from other valid judgments.	12
Jump Statement (1)	Jumps are changes in the flow of control typically invoked by goto commands of the form goto I.	26 (p. 53)
Jump Statement (2)	When a jump occurs the normal continuation is ignored and control passes to a continuation corresponding to the rest of the rpogram following the label jumped to.	26 (p. 53)
Junction	Interface point where a system with expanded fields passes data, receives data or otherwise interacts with another system whose corresponding fields have remained unchanged.	15
Kernel	The privileged core of an operating system, responding to service calls from processes and interrupts from devices.	4
Key (1)	A set of attributes in the entity--relationship model that serves as a unique identifier for entities.	16
Key (2)	A set of attributes in a relation schema that functionally determines the entire schema.	16
Key (3)	Candidate key--a minimal key.	16
Key (4)	Primary key--a candidate key chosen as the primary means of accessing the entire set or relation.	16
Knowledge Base	Information repository that contains object definitions comprising a system's design representation and the relationships among objects as well as syntactic and process rules that define a correct design within the development methodology in use.	15
Knowledge Discovery Meta-Model (KDM)	The foundational meta-model of ADM.	17
Label	A unique identifier assigned to a statement or program location.	NSD
Label Statement	A statement denoting a location to which flow of control can pass to.	NSD
Label Reference	A reference to a label.	NSD
Lambda Calculus	A simple syntactic model of computation equal in power to the Turing Machine.	6
Latent Type System	A type system where types are associated with values, not variables. This usually requires run time type checking, which is why latently typed languages such as SCHEME are often referred to as dynamically typed.	6
L-Attributed Grammar	An attribute grammar whose attributes may be computed by a left to right traversal of the source program. An attribute grammar must be L-attributed for the attributes to be computable during a parse that processes the input from left to right (as most parsers do). Synthesized attributes are always L-attributed.	10
Layer	A special KDM subclass of the meta-class Container that specifies layers of the so-called Logical View of the software architecture. Layers provide a different hierarchy of modules.	16
Lazy Evaluation	An evaluation technique for nonstrict functional languages.	6
Lazy Functional Languages	A function language adopting normal-order evaluation	6
Leaf Procedure	A procedure that does not call another procedure.	11

Lexeme	The actual character string read from the input when recognizing a token. The lexeme of an identifier token is the identifier name.	10
Library	This is a collection of Binary Components. For the sake of generalization we have also allowed “link” association from Binary Components into Libraries. Usually, Libraries (together with Binary Components) are “linked” into Executables.	16
Lightweight Process	A thread.	4
Link	Association between One or more Binary Components and an Executable, or a Library and Executables, or even Binary Components and Library. Note: need to clarify cardinality.	16
Literal	Denotes an explicit value of a given type, for example, a number, an enumeration value, a characer, or a string.	18
LL(k)	A top-down parsing algorithm that processes the input from left to right, producing a leftmost derivation using k tokens of lookahead. The term can also be applied to a language that can be unambiguously parsed using this algorithm.	10
Local	Available only to the procedure within which it is declared.	25 (p. 351)
Local Data Element	Data elements used to accomplish an intermediate task or calculation within a program and only within that program.	15
Local Data	Local Data belongs to a certain Callable Unit. Each instance of this particular Callable Unit defines its own instances of Local Data.	16
Local Declaration	A declaration available only to the procedure within which it is declared.	NSD
Location	Attribute of the meta-class Relation that specifies the linenumber and position of the relation in the corresponding File.	16
Logic Flaw	See Defect.	15
Logic Path	The sequence of coded instructions in a computer program that performs a function or task from one to many times. Logic paths lead to a termination or program exit point, back to the point of invocation (into another path) or into a loop.	15
Logical Data Element	An attribute describing a data item defined to one or more programs that has been defined one time within the system (i.e. no redundancy). There are typically many physical element definitions to one logical data element definition.	15
Logical Record	The full aggregation of attributes with no redundancy describing an I/O record in the context of the programming language being used. There may be many physical records used to describe a single logical record.	15
Lookahead LR(1) [LALR(1)]	The algorithm invented by DeRemer [1971]. The algorithm used in many bottom-up parser generators, including Yacc. A language is also called LALR (1) if it can be parsed unambiguously parsed by the LALR(1) algorithm.	10
Loop	This is a special subclass of the Primitive Action meta-class that represents repretition, for example, the for-loops in C. This class can be further subclassed to represent more language-specific forms.	16
Loop Invariant	In axiomatic semantics, a logical property of a while-loop that holds true no matter how many iterations the loop executes.	13
Loop Statement	See Iteration. Definite loop: a loop whose iteration count is known at entry. Indefinite loop: a loop whose iteration count is unknown at entry time: e.g. based on values calculated within the loop.	20 (p. 533-534)

LR(k)	A bottom-up parsing algorithm that processes the input from left to right, producing a rightmost derivation (inreverse) using k tokens of lookahead. The term can also be applied to a language that can be unambiguously parsed using this algorithm.	10
Macro	Many assembly (and programming) languages provide a "macro" facility whereby a macro statement will translate into a sequence of assembly (or high-level) language statements and perhaps other macro statements before being transated into machine code. There are two aspects to macros: definition and use.	25 (p. 3)
Macro Call	The use of a macro.	NSD
Macro Definition	The definition of a macro.	NSD
Macrokernel	A large operating system core that provides a wide range of services.	4
Maintainability	Ease of maintaining software. It can be further decomposed into evolvability and repairability.	14
Maintenance	Process of applying changes to existing systems that typically includes corrective, adaptive and perfective activities. Large scale architectural change is typically not considered maintenance. Corrective maintenance identifies and corrects implementation flaws or design errors. Adaptive or functional maintenance deals with changes in the data requirements or processing environment. Perfective maintenance supports performance improvement, cost reduction efforts or required technical improvements.	15
Maleability	Ease of modification of a software product. The product can be modified without modifying its design.	14
McCabe Metrics	Metrics described by Thomas McCabe. Basic metric types include cyclomatic complexity which measures the testability of a program and essential complexity which measures the structure of a program. (Also see Cyclomatic Complexity and Essential Complexity)	15
Measure	A method assigning comparable numerical or symbolic values to entities in order to characterize an attribute of the entities.	
Measurement	A numerical or symbolic value assigned to an entity by a measure.	
Measurement Accuracy	The measurement by which another measurement may be wrong.	
Measurement Scope	The domain (set of entities) to which a given measure may be applied.	
Measurement Range	The range (set of comparable values) assignable by a given measure.	
Member Definition	Any specification of the implementation of something defined as part of the definition of a class.	NSD
Member Data	Member Data belongs to a certain Class Unit. Each instance of this particular Calls Unit defines its own instances of Member Data.	16
Member function (or method):	A procedure or function that is defined as part of a class and is invoked in a message passing style. Every instance of a class exhibits the behavior described by a member function/method f the class.	19
Message Passing	A technique for providing mutual exclusion, communication, and synchronization between concurrent processes via sending massage between processes.	9
Metalevel Interpreter	An interpreter written in L for the language L.	8
Method	Procedures within an object that operate on the instance variables of the object and/or send message to other objects.	1
Methodology	A combination of methods and techniques promoting a disciplined approach to software development, maintenance or modernization.	14

Metrics	A numeric count or derived score that quantifies attributes of a system, external system descriptions, user or support areas or a relationship between two or more existing and / or planned systems.	15
Microkernel	A small privileged operating system core that provides process scheduling, memory management, and communication services.	4
Middleware	Technology that facilitates communication between workstation and host platforms by linking graphical front-end technology to a host environment. This interface includes direct links to legacy programs, legacy data and / or other server platforms.	15
Module (1)	A well delineated, executable program.	15
Module (2)	A module is a unit of SQL language, containing declarations of certain SQL entities as well as one or more procedures, each containing a single SQL statement.	2
Module (3)	This is a special subclass of the meta-class Container that specifies modules of the so-called Logical View of the software architecture. Modules are the basic building blocks of software. Modules are grouped into subsystems. Modules are also assigned to layers.	16
Monitor	An encapsulation of a resource and the operations on that resource which serves to ensure mutual exclusion	9
Multiprocessing	The use of multiple processors that share a common memory.	9
Multiprogramming (1)	Scheduling several competing processes to run at essentially the same time.	4
Multiprogramming (2)	Simulating concurrency on a single processor by interleaving instruction execution from multiple processes; time sharing or time slicing.	9
Mutual Exclusion	The property ensuring that a critical region is executed indivisibly.	9
Name	Attribute of the KDM meta-class “Entity”.	16
Nanokernel	A very small privileged operating system core that provides simple process scheduling and communication services.	4
Natural Semantics	A hybrid of operational and denotational semantics that shows computation steps performed in a compositional manner. Also known as a big-step semantics.	13
Nesting Level	The depth within a program logic path at which references to procedures reside in a program. The first level of nesting is the initial program entry point. Also the depth within a system logic path and generally measured by Call structure control transfers between modules. A driver program Call to a subroutine would transfer control from level one to level two.	15
Network Services	Services available through the network, such as mail and file transfer.	4
Networked Operating System	An operating system that uses a network for sharing files and other resources.	4
Nonprivileged State	An execution context that does not allow sensitive hardware instructions to be executed, such as the halt instruction and input/output instructions.	4
Non-Returning Call	Branch to a sub-routine that never returns control to calling program. Example: Call Abend.	15
Nonterminal	A name for a structure defined by a context-free grammar rule. Interior nodes of parse trees are labeled by nonterminals.	10
Normal-Order Evaluation	An execution order in which the arguments in a functional call are only evaluated if and when needed in the body of the function	6
Object (1)	Data and behavior (method) representing an entity.	1
Object (2)	An abstraction of certain business information, within a specific problem domain, that encompasses both data and processes (behavior) associated with that information.	15

Object Oriented Design	Design technique that incorporates concepts of object oriented programming such as information hiding, inheritance, classes, abstract data types, and messages. It supports object oriented programming techniques that involve a reusable building block approach to development.	15
Object-Oriented Programming	A method of implementation in which a program is described as a sequence of messages to cooperating collections of objects, each of which represents an instance of some class. Classes can be related through inheritance and objects can exhibit polymorphic behavior.	7
Occur-Check	A test performed during unification to ensure that a given variable is not defined in terms of itself [e.g. X = f(X) is detected by an occur-check and unification fails	8
Off Line	Handled on a different computer.	4
Operation	The execution of a business rule in event modeling.	15
Operational Semantics	The meaning of a program as calculation of a trace of its computation steps on input data.	13
Operator (1)	Any one of the special symbols (such as '*' or 'mod') that perform some logical or mathematical operation upon objects and literals.	NSD
Operator (2)	An employee who performs the repetitive tasks of loading and unloading jobs.	4
Override	The action that occurs when a method in a subclass with the same name as a method in a superclass takes precedence over the method in the superclass.	7
Parameter	One of the named entities associated with a subprogram, entry, or generic program unit.	18
Parameters	Data objects passed between the caller and the called procedural abstraction.	5
Parsing	The conversion of physical system components into data structures suited to code transformations within a compiler or software engineering tool. All tools used to document, analyze, import, improve, reverse or in any other way process existing source or executable object types of any nature require this facility.	15
Path	Attribute of the meta-class File that specifies the location of the File. For example, it can represent a URL, a full filename, etc.	16
Perform Range	The object of a Perform statement in COBOL. A Perform range can be either a Section (with a reference in a statement of the form "Perform Section name"), a single paragraph (with a reference in a statement of the form "Perform paragraph name") or a group of paragraphs and/or Sections.	15
Perform Range Violation	A construct in which a COBOL Go To statement within the range of the Perform branches outside the controlling Perform. A Perform range violation is considered a violation of structured coding principles.	15
Performance	In software engineering, performance is a synonym for efficiency. It refers to how economically the software utilizes the resources of the computer.	14
Persistence	The ability of information to survive (persist) despite failures of all kinds, including crashes of programs, operating systems, networks, and hardware.	1
Persistent Data	Persistent Data represents files, records, databases, etc.	16
Phase	A logical unit of a compiler. Typical phases include scanning, parsing, semantic analysis, and code generation. Phases are to be distinguished from passes, which comprise a complete sequential processing of the input program. Phases may or may not correspond to the physical code units within the compiler.	10
Physical	The material on which abstractions are built.	4

Physical Address	A location in physical memory.	4
Physical System Components	The actual pieces that comprise an executable production system such as JCL, load modules, source code, screen maps and related items.	15
Pipeline	A facility that allows one process to send a stream of information to another process.	4
PL/I	A commonly used third generation programming language found in many IBM mainframe environments	15
Platform Resource	This is a part of Software Platform, something that is "consumed by" Platform Elements. For example, stack space, heap space, code space, CPU cycles, bandwidth, etc. Platform Resources are assigned to hardware resources.	16
Pointer	An attribute of one object that contains an explicit reference to another object.	17
Pointer Expression	An expression that resolves to an attribute of one object that contains an explicit reference to another object.	NSD
Polymorphism (1)	A property of a languages type system in that an objects type may include type variables which can rang e over an infinite number of types. Most such polymorphic objects are functions which can be applied to arguments of many different types.	6
Polymorphism (2)	The ability of a program fragment to have multiple types (opposite of monomorphism)	12
Polymorphism (3)	That feature of a variable that can take on values of several different types or a feature of a function that can be executed using arguments that can be executed using arguments of a variety of types.	7
Precondition	In event modeling, a condition that must be met before an operation can occur.	15
Predicate Calculus	A calculus for expressing logic statements. Its formulas involve: atoms: P (Ta, Ts, …) where P is a predicate symbol and T1 are terms. -- Boolean connectives: conjunction(^), disjunction (v), implication (->), and negation (~) -- literals: atoms or their negations -- Quantifiers; for all (upside A), there exists (backwards E) -- terms (also called trees): constructed from constants, variables, and function symbols.	8
Premature Exit	See Unresolved Perform.	15
Primary Data Elements	Data elements necessary to send information to and receive information from a system or program. Excludes elements a program may define to accomplish some intermediate task or calculation. Primary elements are typically the basis for any redesign or integration effort.	15
Primitive Action	A Primitive Action is language-based. Primitive Actions are associated with Callable Units (each Primitive Action is hosted by exactly one Callable Unit). Primitive Actions are further subclassed into the following: -Assignment -Condition -Call -Loop -Return -Instantiation -Destruction	16
Primitive Type	See basic type	NSD
Privileged State	An execution context that allows all hardware instructions to be executed.	4
Procedural Abstraction	Separating out the details of an execution unit in such a way that it may be invoked in a program statement or expression.	5
Procedural Coupling	A logic construct in which three or more procedures transfer control to each other. Often these procedures are inter related in a complex manner, with numerous references to each other. The impact on its ability to be understood is typically negative.	15
Procedure (1)	A module, or comparable construct, in a procedural language.	15
Procedure (2)	A paragraph or Section in the Procedure Division of a program (especially COBOL).	15

Procedure (3)	A procedure is a subroutine written in SQL, contained in a module that is invoked by means of a call from a program written in a conventional programming language.	15
Procedure (4)	Any operation that does not return a significant value	17
Procedure (5)	Any operation that may perform multiple actions and modify the instance to which it is applied, but does not return a value	17
Procedure Call	A call to a procedure.	NSD
Process (1)	A program being executed; an execution context that is allocated resources such as memory, time, and files.	4
Process (2)	A low level activity that begins and ends.	15
Process Descriptor	A data structure in the kernel that represents a process.	4
Process Interface	The set of service calls available to processes.	4
Process Number	An identifier that represents a process by acting as an index into the array of process descriptors.	4
Process Switch	The action of directing the hardware to execute in a different context (kernel or process) from the current context.	4
Processor State	Privileged or nonprivileged state.	4
Production	Another term for context-free grammar rule or grammar rule choice.	10
Productivity	Efficiency of the software process	14
Program	Any static object-based application consisting of a set of types and classes interrelated specific to a particular (end-use) objective [OMG].	17
Programming Language	A language for programming, such as COBOL, C, C++, Java, etc. consisting of syntax and semantics for programming. The syntax of the language specifies those combinations of symbols which are in the language. The semantics specifies the "meaning" of syntactically correct constructs in the language.	23 (p. 1)
Programming Paradigm	The design space of programming languages is partitioned into paradigms, mechanisms for compiling and run-time management and language theory. Topics include object-oriented, functional, logic, and imperative programming paradigms.	27 (p.1981)
Protability	Software is portable if it is able to run on different machines.	14
Prototyping	An incremental development process where intermediate stages constitute executable prototypes of the end product; that is, they are just an approximation of it.	14
Provide	This association specifies interfaces of a given module that are used by other modules.	16
Pseudofile	An object that appears to be a file on the disk but is actually stored elsewhere.	4
Pure Functional Languages	Functional languages that provide absolutely no mechanism for performing side effects, and thus exhibit referential transparency.	6
Push and Factor	An approach used to restructure programs, in which control flow of a program is pushed through to termination and factored to force convergence and performability.	15
Qualified Expression	An expression qualified by the name of a type or a subtype. It can be used to state the type or subtype of an expression, such as an overloaded literal.	18
Qualified Identifier Reference	A reference to a qualified name.	NSD

Qualified Name	The name of any feature that has been qualified by the name of its enclosing class. Qualified names are used to prevent overriding in order to explicitly select the correct associated implementation.	17
Quality	The totality of features and characteristics of a product that bear on its ability to satisfy stated or implied needs [ISO 9241-11 1996]	3
Quality Assurance	The process of verifying whether a software product meets the required qualities. The critical evaluation of software systems and the application of standards of quality to that system. This ranges from simply monitoring code structure, to programmer training or through the implementation of a comprehensive quality program.	14 \ 15
Race	The state in which two or more processes are continuously competing for resources and neither progresses	9
Range	A contiguous set of values of a scalar type. A range is specified by giving the lower and upper bounds for the values. A range may be used in a membership test.	18
Range Expression	An expression that evaluates to a range.	NSD
Range Type	A type that characterizes a range.	NSD
Rapid Prototyping	Involves iterative, rapid refinement of software designs by quickly generating working prototypes and using feedback of prototype results to improve design specifications.	15
Real-Time System	A software system that interacts closely with an external physical environment and that must respond to external physical events in the time frame dictated by the characteristics of the external system.	15
Recursion	Construct where a procedure invokes itself, directly or indirectly. At a system level, recursion may be Call based, where a module can invoke itself through a chain of called subroutines. Recursion is a violation of structured design principles that results in functional and maintenance related problems.	15
Recursive-Descent	A top-down parsing algorithm that translates context-free grammar rules into a set f mutually recursive procedures, with each procedure corresponding to a nonterminal. Recursive descent parsing is usually the method of choice when writing a parser by hand.	10
Reduce-Reduce Conflict	In bottom up parsers, a property of a state in which a parser has a choice of two productions which can be used to reduce the parsing stack, and both are legal for the amount of lookahead allowed. Reduce-reduce conflicts have no natural disambiguating rule.	10
Refactor (1)	A collection of source code to source code improvement or front-end tasks that facilitate the understanding and maintenance of programs and/or systems, while preparing systems for redesign and reuse. Examples include restructuring, slicing, data name rationalization and middleware enabling.	15
Refactor (2)	Task of functionally evolving a program or system. See maintenance.	15
Reference	A reference to a value is just a location holding it.	26 (p. 118)
Reference Expression	An expression that evaluates to a reference.	NSD
Referential Integrity	Referential integrity places requirements on data so that values stored in one or more columns of rows of one table, called the referencing table, are identical to values stored in corresponding columns of rows of another table, called the referenced table.	2
Referential Transparency	The property of a language that states that equal expressions can be interchanged with each other	6
Register Window	A collection of registers assigned to an executing process in the SPARC architecture.	11
Relation (1)	A subset of a Cartesian product of domains.	1

Relation (2)	Informally, a table.	
Relation (3)	This meta-class is defined at the Conceptual Model diagram.	16
Relation Schema	A type definition for relations consisting of attribute names and a specification of the corresponding domains.	1
Relational Data Model	A diagram which describes data elements and relationships for part of a business, or for an entire enterprise. Involves "normalizing" data to optimize the design and implementation of that model.	15
Relationship	An association among several entities.	1
Reliability	Software is reliable if the user can depend on it. The mathematical theory of software reliability provides a definition based on statistics and probability principles.	14
Remodularization	Process of dividing a system or program into several smaller, more manageable modules and / or recombining those programs along functionally cohesive lines. This may include reconciliation of redundant logic and functional realignment of physical systems.	15
Rendezvous	The message-passing construct used in the Ada language.	9
Repairability	A software system is repairable if it allows the correction of its defects with a limited amount of work.	14
Replication	Technique used by legacy designers and programmers to rapidly deploy replacement systems or build new programs by copying and modifying existing programs. It differs from reusability in that functionality of the originating component is cloned and modified one or more times, forcing departure from the original baseline and replication of maintenance effort over the long term.	15
Repository	An information storage facility or central database that contains all meta-data relevant to the management, design, implementation and transition of one or more information systems and / or for the enterprise. A repository is typically the physical implementation vehicle for a meta-model of system or other meta-data.	15
Require	This association specifies interfaces of other modules that correspond to the cross-module relations.	16
Resident Monitor	A precursor to kernels; a program that remains in main store during the execution of a job to handle simple requests and to start the next job.	4
Resolution	A single inference step used to prove the validity or predicate calculus formulas expressed as clauses. In its simplest version: P or Q and ~P imply Q or R, which is called the resolvant.	8
Resource (1)	A commodity necessary to get work done.	4
Resource (2)	This is anything that is used by an Executable at run-time, for example, a bitmap image, an audio file, a data file, etc.	16
Restructure	To convert an unstructured source program to structured source program by reducing the essential complexity of all module sub-graphs to a one.	15
Retargeting	The process of changing a compiler to produce target code (assembly or machine code) for a different machine. This may involve rewriting the compiler back end or creating a machine definition file for the new machine.	10
Retention	The retention of a Copy statement in a program during the restructuring process. (Antonym: See Expansion.)	15
Return	This is a special subclass of the PrimitiveAction meta-class that represents return from a CallableUnit, for example, a return statement in C. This class is a terminator for the Action graph of a CallableUnit. Value passing out of a CallableUnit is represented using the Assignment actions. This class can be further subclassed to represent more language-specific forms.	16

Return [Statement]	A keyword that causes a method to return to its caller. This is normally done as the last statement in a method, but return can appear anywhere within the body of a method. If a method has been declared as void, there is no return value and the return statement will not accept an argument. If the method has been declared as rerunning any data type other than void, a return statement I required, and the return statement must be followed by an expression of the correct type.	24 (p. 1221)
Reusability	The act of analyzing the essence of existing systems, constructs or designs as input to the creation of new or target systems, constructs or designs. Implies either physically or logically referencing or replicating system artifacts in other systems.	15
Reverse Engineering	Process of analyzing a system to identify components and interrelationships, and to create representations in another form or at a higher level of abstraction. Source: IEEE.	15
Robustness	Software is robust if it behaves "reasonably" even in circumstances that were not anticipated in the requirements specification.	14
Row	A row is an instance of a row type. A row type is a sequence of {name, datatype} pairs. A row is the smallest unit that can be inserted into or deleted from a table.	2
Runaway Path	A logic path in which the flow of control falls past the last statement of a program. A runaway path is considered a violation of structured coding principles.	15
Run Time Entity	This is a special KDM subclass of the meta-class Container that specifies run-time “things”, such as processes, threads, etc. A RunTimeEntity is a Container for one or more Modules that are “assigned to” it. However, relations between RunTimeEntities are established through the so-called CommunicationMechanisms of the corresponding SoftwarePlatform (operating system, etc.).	16
Safe Language	A language where no untrapped errors can occur	12
Scenario (1)	A project oriented, assessment and implementation work plan template. Scenarios provide a means for organizations to rapidly deploy modernization projects.	15
Scenario (2)	A KDM meta-class represents an approximation of a use case as a behaviour graph of actions, associated through Data Flow Dependencies.	16
Scenario End	A special node, representing the termination of a particular scenario.	16
Scenario Model	Scenario-Model is a set of Scenarios.	16
Scenario Start	A special node, representing the start node of a particular scenario	16
Scheduler	An operating system module that manages the time resource.	4
Schema	A schema is a named conceptual entity that contains tables, constraints and other SQL objects. In SQL, data are considered to be associated with, but distinct from, schemas. The name of a schema is used to qualify the names of the objects in that	2
Scope	The region of program text over which a declration has an effect (is in existence). , also Program scope, procedure scope, block scope, type scope	18
Scope Rules	Rules in a language that define the area or section or a program in which a particular binding is effective.	5
Score	An algebraic combination of counts or other scores to produce a numeric descriptive value of an application or the external components supporting that organization.	15

Second-Order Type System	One that includes quantification over type variables, either universal or existential.	12
Self-Contained	The property of a Copy member which allows its associated Copy statement to be retained during the restructuring process.	15
Semaphore	A nonnegative integer-valued variable on which two operations are defined; P and V to signal intent to enter and exit, respectively, a critical region.	9
Sentence	One or more statements terminated by a period followed by a space in the Procedure Division of a program.	15
Server	A process that responds to requests from clients via messages.	4
Service Call	The means by which a process requests service from the kernel, usually implemented by a trap instruction.	4
Session	The period during which a user interacts with a computer	4
Shared File System	Files residing on one computer that can be accessed from other computers.	4
Shift-Reduce Conflict	In bottom up parsers, a property of a state in which a parser has a choice of two productions which can be used to reduce the parsing stack, and both are legal for the amount of lookahead allowed. A natural disambiguating rule is to prefer the shift, thus allowing the parser to match the longest possible input string at each point.	10
Side Effect	A change in the value of a variable as the result of evaluating an expression, e.g., if the expression contains an assignment operation.	6
Signature Unit	This is a subclass that represents a type signature of a CallableUnit. This is a list of DataTypeUnits. Each CallableUnit has zero or one SignatureUnit. SignatureUnits are very important architecturally, because they introduce physical dependencies on types. For example, when performing architecture understanding and refactoring of existing software assets, it is important to distinguish between the “usage in signature” and “usage in body”.	16
Sink	Attribute of the meta-class relation, that specifies the “entity” that is the sink of the relations.	16
Site	The set of computers, usually networked, under a single administrative control.	4
SLD Resolution	Selective linear resolution for definite clauses inference step used in proving the validity of Horn clauses.	8
SME	See Subject Matter Expert.	15
Software Process	Activities through which a software product is developed and maintained.	14
Software Product	All of the artifacts produced by a software process. This definition encompasses not only the executable code and user manuals that are delivered to the customer but also requirements and design documents, source code, test data, etc.	14
Software Reengineering	The use of tools and techniques to facilitate the analysis, improvement, redesign and reuse of existing software systems to support changing business and technical information requirements.	15
Software Platform	A Software Platform provides certain types of Platform Resources.	16
Source (1)	The thing operated upon or used as input to an operation or a complex process (i.e. a code generator, translator or transformer).	NSD
Source (2)	Attribute of the meta-class relation, that specifies the “entity” that is the source of the relation	16
Source (3)	See Source code.	NSD
Source Component	This is a special KDM subclass of the Container meta-class. It is also a subclass of File.	16
Source File (1)	A file in which code in a programming language is located.	NSD
Source File (2)	A file from which source is taken	NSD

Source Location	A location within a source file.	NSD
Source Program	A program defined in the programming	NSD
Spaghetti Code	A severely unstructured program or Section of code. It is called "spaghetti code" because a diagram of the logic paths in the program resembles the Italian pasta.	15
Specialization (1)	(a) The process of creating a specialization from one or more generalizations (b) the creation of a subclass via extension, refinement or restriction. (c) An extension of the behavior of ta type of object.	17
Specialization (2)	(a) the result of using the specialization process (bY any derived class,	17
Specialization (3)	(a) any relationship from a generalization to one or more of its specializations (b) the relationship between a class and its parents, (c) the relationship between a parent and a descendant that has been modified by refinement or deletion so that the descendant is no longer behaviorally compatible with its parent.	17
Spooling System	Stored newly arrived jobs on disk until they can be run, and storing output of old jobs on disk until they can be printed.	4
SQL Statement	An SQL statement is the smallest executable unit of SQL. SQL statements are used to create, alter, and remove metadata objects, as well as to insert, update and delete data in tables and to retrieve data from tables.	2
Stack Pointer	A register that points to the top of the activation stack.	11
Statement	A syntactically valid sequence of words or symbols beginning with a COBOL verb.	15
Static Analysis	A method of analyzing a program by tracking and identifying program logic paths without executing the program. Static analysis shows how the program is organized and how the control flow is directed by the use of various verbs and constructs.	15
Static Checking	A collection of compile-time tests, mostly consisting of type checking.	12
Statically Checked Language	A language where bad behavior is determined before execution.	12
Strict Functional Language	A functional language adopting applicative-order evaluation	6
Strongest Post Condition Semantics	A variant of axiomatic semantics where a program and an input property are mapped to the strongest proposition that holds true of the program's output.	13
Strongly Checked Language	A language where no forbidden errors can occur at run time (depending on the definition of forbidden error).	12
Structural Operational Semantics	A variant of operational semantics where computation steps are performed only within prespecified contexts. Also known as small-step semantics.	13
Structure	NSD	NSD
Structured Programming	A family of coding techniques in which a program's control logic proceeds from a controlling procedure (paragraph) to a subordinate procedure (paragraph) and back. Use of structured coding techniques results in code that is easier to trace, understand and maintain.	15
Subclass	A class that lies below some other class (a superclass) in a class inheritance hierarchy; a class that contains a subset of the objects in a superclass.	1
Subclass (or, Derived Class)	A class that inherits variables and methods from another class (called the superclass)	7

Subject Area	A high level classification of data that defines a group of entity types directly pertaining to a major function within the enterprise. Source: Information Engineering.	15
Subject Area Diagram	A high level diagram defining subject areas for an area of the business. Source: Information Engineering.	15
Subject Matter Expert	One who is knowledgeable in the functional or technical aspects of an application system or other area of study, such as redevelopment; also called a SME.	15
Subquery	A subquery is an expression that identifies data from tables. Subqueries are used as expressions in the context of SQL statements.	2
Subsumption	A fundamental rule of subtyping, asserting that if a term has a type A, which is a subtype of type B, then the term also has type B.	12
Subsystem	This is a special subclass of the meta-class Container that specifies subsystems of the so-called Logical View of the software architecture. Subsystems are the building blocks of software. Subsystems consist of modules. Subsystems are arranged into hierarchies. Modules are also assigned to layers.	16
Subtype	A new data type defined as a copy of another defined type, typically with a restricted subset of its values. It may generally be used in the same contexts as its parent type.	5
Subtyping	A reflexive and transitive binary relation over types that satisfies subsumption; it asserts the inclusion of collections of values.	12
Switch	A locally defined data element that is set and tested in a program and that controls the branching of logic flow through that program.	15
Switch Statement	A statement used to compare an expression to a series of possible cases. If the expression matches the case, the code below the case statement is executed. A break statement is used to exit the switch block once a case has been matched. If the expression matches none of the case statements, the code below the default statement is run. The break and default statements are not mandatory.	22
Synchronous Message Passing	The message sending process requires both sender and receiver to synchronize at the moment of message transmission.	9
Synonyms	Data elements mapping to the same physical data, but having a different definition and a different data name. These are usually redefining data elements. Synonyms are usually valid redefinitions of data elements. They are needed for various reasons such as for reporting and computations.	15
Syntactic Pattern	Specific program constructs or logical relationships among symbols independent of their function. Various static analysis products measure various types of Syntactic Patterns.	15
Synthesized Attribute	An attribute whose value depends only on the attribute values of descendants in the parse or the syntax tree. Synthesized attributes are the easiest to compute during a parse, requiring no special data structures or techniques. The syntax tree itself is the most important example of a synthesized attribute.	10
Systems Redevelopment	The process of significantly modifying or re-building application system (s) that essentially replaces portions or all of one or more existing systems through the use of software reengineering technology.	15
Table	A table is a collection of rows. Every row in a table has the same row type. The degree of a table is the number of columns that it has. The cardinality of a table is the number of rows in that table.	2
Target	The produced by or output by an operation.	NSD

Target File	A file (typically code) produced by an operation.	NSD
Task (1)	A specification including the intended result of an activity to be performed on an object (material) with a specified means (method).	3
Task (2)	A low level activity that begins and ends.	15
Task Performance	An activity carried out at a user interface and aimed at completing a task.	3
Template	The C++ term for any parameterized meta-class or any parameterized function. A class template specifies how a family of individual classes can be constructed much as a class declaration specifies how individual objects can be constructed.	17
Template Definition	The definition of a template.	NSD
Template Type	The definition of a template type.	NSD
Terminal	Another term for a token in a context-free grammar. Leaf nodes of parse trees are labeled by terminals.	10
Terminating Call	In a source program, a call to an external program or routine that does not return control to the calling program. Program termination will always occur. Also called non-returning Call.	15
Testing	The process of executing one or more programs in a system to verify that functional capability of that system meets the user requirements specified during a prior development or maintenance modification.	15
Thread	An execution context that is independently scheduled, but shares a single address space with other threads.	4
Throw Statement	A throw statement is executed to indicate that an exception has occourred (*i.e., a method could not complete successfully). This I called throwing an exception. A throw statement specifies an object to be thrown.	21 (p. 809)
Time	A resource; ability to execute instructions.	4
Timeliness	A process-related quality meaning the ability to deliver a product on time.	14
Timesharing	Interactive multiprogramming.	4
Top-Down (1)	A family of design and coding methodologies in which a system or program development effort proceeds from a controlling procedure or structure to a subordinate procedure or structure. Use of these methodologies results in systems that are easier to understand and maintain.	15
Top-Down (2)	A parsing algorithm that constructs the parse tree from the root to the leaves. Top-down algorithms include LL parsers and recursive-descent parsers, such as those produced by PCCTS.	10
Transfer Created Logic Loops	A loop caused by a Go To or a Fall Through; that is, a loop caused by a permanent transfer of control rather than a temporary transfer such as a Perform.	15
Transformation	A stage of modernization that is model driven and results in a change to the existing technical, data and / or functional architecture.	15
Trapped Error	An execution error that immediately results in a fault.	12
Trigger	An interpretation of an event that activates an operation in event modeling.	15
Type (1)	A collection of values with an associated collection of primitive operations on those values.	5
Type (2)	A collection of values. An estimate of the collection of values that a program fragment can assume during program execution	12

Type Definition	A type definition is a language construct introducing a new, unique type, whereas a subtype creates a compatible (possibly) constrained definition of the base type.	18
Type Declaration	A type declaration associates a name with a type introdueced by a type definition.	18
Type Equivalence	Rules that govern when variables or value from two different data types may be used together.	5
Type Inference (1)	A process in which the compiler determines the types of objects in a program without the programmer having to declare them explicitly.	6
Type Inference (2)	The process of finding a type for a program within a given type system.	12
Type Reconstruction	The process of finding a type for a program where type information has been omitted, within a given type system.	12
Type Rule	A component of a type system. A rule stating the conditions under which a particular program construct will not cause forbidden errors.	12
Type Safety	The property stating that programs do not cause untrapped errors.	12
Type Soundness	The property stating that programs do not cause forbidden errors.	12
Type System	A collection of type rules for a typed programming language. Same as static type system.	12
Typechecker	The part of a compiler or interpreter that performs typechecking.	12
Typechecking	The process of checking a program before execution to establish its compliance with a given type system and therefore to prevent the occurrence of forbidden errors.	12
Typed Language	A language with an associated (static) type system, whether or not types are part of the syntax.	12
Type Usage	TypeUsage is a subclass of DesignRelation that represents usages of types between DataTypeUnits, SignatureUnits and DeclarationUnits. Usually, TypeUsages are compile-time relations.	16
Typing Error	An error reported by a typechecker to warn against possible execution errors.	12
Unary Expression	An expression that computes a relation on one thing.	NSD
Unentered Procedure	A Section or paragraph which is never executed because it cannot be reached by the flow of control logic through the program regardless of data values.	15
Unexecutable Statement	A statement which is never executed because it cannot be reached by the flow of control logic through the program regardless of data values.	15
Unification	Matching of terns used in a resolution step. It basically consists of testing the satisfiability or the equality of trees whose leaves may contain variables. Unification can also be viewed as a general parameter matching mechanism.	8
Unit of Measure	A quantity in terms of which the magnitudes of other quantities within the same total order can be stated.	
Unresolved Perform	A coding construct where a procedure exit (end of Perform range) is not reached when the exit of an invoking procedure is reached. An Unresolved Perform is caused by a Perform range violation and is considered a major violation of structured coding principles; also called Premature Exit.	15
Untrapped Error	An execution error that does not immediately result in a fault.	12
Upgrade Unit	A system, or group of related systems, being treated as a single unit of work for purposes of a given redevelopment project scenario. One common application of upgrade unit segmentation is for enterprise wide century date change projects.	15
Usability	The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use [ISO 9241-11 1996].	3

Use	A KDM association between a Module and zero or more other Modules. This corresponds to an “abstracted” relation between the two Modules: any cross-module primitive relations between entities in both modules are rolled-up into the Uses relation and the corresponding association.	16
Use at Runtime	Association between Executable and Resource. Note: it is actually the RunTimeEntity that is capable of using Resource, not the Executable itself.	16
Use Mechanism	This is an association between zero or more CommunicationPaths and exactly one CommunicationMechanism (some CommunicaitonMechanisms may never become instanciated, and each CommunicationPath uses exactly one CommunicationMechanism).	16
User	A human being physically interacting with a computer.	4
User Identifier	A number or string that is associated with a particular user.	4
User Interface	An interface that enables information to be passed between a human user and hardware or software components of a computer system [ISO 12119 1194].	3
Valid Judgment	A judgment obtained from a derivation in a given type system.	12
Validation	The process of running selected input data through an original program and then running that same data through a program that has been modified through refactoring task to ensure that the modified program has retained functional equivalency. The output data of both validation tests should be identical, barring expected differences such as date / time stamps or field size variations.	15
Variable	An abstraction used in imperative languages for memory location or cell.	5
Verifiability	A software system is verifiable if its properties can be verified easily.	14
View	A view is a virtual table--a table that is not stored physically in the database, but whose contents are derived only when needed by an SQL statement.	2
Virtual	The result of abstraction; opposite of physical.	4
Virtual Address	An address in memory as seen by a process, mapped by hardware to some physical address.	4
Virtual Function (or, Deferred Method)	Most generally, a method of a class that must be defined by subclasses to the class. In languages in which dynamic binding is not the default, may also mean that a function is subject to dynamic binding.	7
Virtual Machine	An abstraction produced by a virtualizing kernel, similar in every respect but performance to the underlying hardware.	4
Virtualizing Kernel	A kernel that abstracts the hardware to multiple copies that have the same behavior (except for performance) as the underlying hardware.	4
Visibility	A process-related quality meaning that all steps and the current process status are documented clearly.	14
Warren Abstract Machine (WAM)	An intermediate (low-level) language that is often used as an object language for compiling PROLOG programs. Its objective is to allow the compilation of efficient PROLOG code.	8
Weakest Precondition Semantics	A variant of axiomatic semantics where a program and an output property are mapped to the weakest proposition that is necessary of the program’s input to make the output property hold true.	13
Weakly Checked Language	A language that is statically checked but provides no clear guarantee of absence of execution errors.	12

Well Behaved	A program fragment that will not produce forbidden errors at run time	12
Well-Formed	Properly constructed according to formal rules	12
Well-Typed Program	A program (fragment) that complies with the rules of a given type system.	12
World Wide Web	A network of service that allows users to share multimedia information.	4

Source #	Source Reference
1	Data Models (pg. 991, The Science and Engineering Handbook, CRC Press, 1997)
2	The SQL Language (pg. 1188, The Science and Engineering Handbook, CRC Press, 1997)
3	International User-Interface Standardization (pg. 1489-1490, The Science and Engineering Handbook, CRC Press, 1997)
4	What is an Operating System? (pgs. 1662-1663, The Science and Engineering Handbook, CRC Press, 1997)
5	Imperative Programming Paradigm (pg. 2004, The Science and Engineering Handbook, CRC Press, 1997)
6	Functional Programming Languages (pg. 2035, The Science and Engineering Handbook, CRC Press, 1997)
7	The Object-Oriented Language Paradigm (pg. 2063, The Science and Engineering Handbook, CRC Press, 1997)
8	Logic Programming and Constraint Logic Programming (pgs. 2091-2092, The Science and Engineering Handbook, CRC Press, 1997)
9	Concurrent/Distributed Computing Paradigm (pgs. 2177-2118, The Science and Engineering Handbook, CRC Press, 1997)
10	Compilers and Interpreters (pgs. 2144-2145, The Science and Engineering Handbook, CRC Press, 1997)
11	Run Time Environments and Memory Management (pgs. 2188, The Science and Engineering Handbook, CRC Press, 1997)
12	Type Systems (pgs. 2234-2235, The Science and Engineering Handbook, CRC Press, 1997)
13	Programming Language Semantics (pgs. 2252-2253, The Science and Engineering Handbook, CRC Press, 1997)
14	Software Qualities and Principles (pg. 2300, The Science and Engineering Handbook, CRC Press, 1997)
15	USRM Glossary of terms - (http://www.comsysprojects.com/SystemTransformation/tmglossary.htm)
16	KDM Submission
17	Dictionary of Object Technology, SIGS Books, 1995
18	Booch, G., Software Engineering with Ada, Benjamin Cummings Publishing Company, 1983
19	The Object-Oriented Language Paradigm (pg. 2063, The Science and Engineering Handbook, CRC Press, 1997)
20	Beizer, Boris, Software Testing Techniques, Van Nostrand Reinhold, 1990
21	Deitel, Harvey M, Java, How to Program, Prentice Hall, 4th Edition, 2001
22	Palmer, G, Java Programmer's Reference, Wrox Press, Ltd., 2000.
23	Backhouse, R.C., Syntax of Programming Languages Theory and Practice, Prentice-Hall, 1979.
24	Griffith, A, Java Master Reference The Definitive Java Langauge Reference!, IDG Books, 1998.
25	Aho, A, Ullman, J., Principles of Compiler Design, Addisen-Wesley, 1977.

26	Gordon, M.J.C., The Denotational Description of Programming Languages An Introduction, Springer-Verlag, 1970.
27	The Computer Science and Engineering Handbook, CRC Press, 1997
28	Wikipedia The Free Encyclopedia. (http://en.wikipedia.org/wiki/Main_Page)