

The AOIS Glossary

On Agent-Oriented Information Systems

Gerd Wagner

Institute of Informatics

Free University of Berlin, Germany

e-mail: gw@inf.fu-berlin.de

Edition 0.2 (April 2000)

This glossary has been written and compiled as a service to the AOIS community. It lists the most fundamental terms and concepts that are relevant to AOIS. Since in some cases, I could not avoid to express my personal views on certain matters, readers are invited to send me comments and criticisms. Also, proposals for additional entries are welcome. Though I cannot promise to include all suggestions made, I will carefully study each of them, and take it into consideration in one way or another.

Index

ACID Transaction, 1
Action, 1
Agent, 1
Agent Communication Language (ACL), 2
Agent-Oriented Information System (AOIS), 2

Business Transaction, 2

Contract Net Protocol, 3
CORBA, 3

Data Warehouse, 3
Database Management System (DBMS), 3
Derivation Rule, 4

EDI, 4
Enterprise Application Integration (EAI), 4
Enterprise Resource Planning (ERP), 5
Entity-Relationship (ER) Modeling, 5

Information System, 5
Integrity Constraint, 6
Interoperability, 6

Knowledge Interchange Format (KIF), 6
KQML, 6

Message Transport, 6
Message-Oriented Middleware (MOM), 6
Mobile Agent, 6
Multiagent Systems (MAS), 6

Object-Oriented Database (OODB), 7
Object-Relational Database (ORDB), 7
OLAP Application, 6
OLTP System, 6
Ontology, 7

Proposition, 8
Protocol, 8

Reaction Rule, 8
Relational Database (RDB), 8
Rule, 8

Speech Act Theory, 9
SQL, 9

TCP/IP, 9

Unified Modeling Language (UML), 9

ACID Transactions satisfy the properties of Atomicity, Consistency, Isolation, and Durability. As complex update operations on possibly distributed data, transactions are subject to various failure conditions. For instance, some step of a transaction may violate an integrity constraint, or the connection to a remote database may get lost, or the server running the transaction application may crash during the execution of a transaction. *Atomicity* denotes the requirement that a transaction needs to be all-or-nothing: either it executes completely or not at all. If all steps of a transaction have been performed successfully, the transaction *commits*, making all involved updates permanent. If some step fails, the transaction *aborts*, rolling back all involved updates. A transaction program should maintain the *consistency* of the databases it updates. Since the internal consistency of a database is defined by its integrity constraints, this means that all integrity constraints have to be satisfied when the transaction is completed. For performance reasons, transactions are executed concurrently by interleaving the execution of their single steps. Special techniques (such as locking mechanisms) are needed to avoid interference between transactions accessing the same database objects. Referring to concurrent transactions, one says that the *isolation* property is satisfied if their effects are the same as if they were run one at a time in some order, or, in other words, if they are *serializable*. Finally, a transaction is *durable* if all of its updates are stored on a permanent storage medium when it commits. This is usually achieved by writing a copy of all the updates of a transaction to a log file. If the system fails after the transaction commits and before the updates go to the database, then after the system recovers it rereads the log and checks that each update actually made it to the database; if not, it re-applies the update to the database.

Notice that unlike atomicity, isolation and durability which are guaranteed by the transaction processing system, maintaining consistency is the responsibility of the application programmer.

Action An action is something which can be done by

an agent. There are elementary and composite actions. In a formal language, an elementary action term may have the form $a(t_1, \dots, t_n)$ where a is the name of an elementary action type and t_1, \dots, t_n are argument terms that may denote actions, propositions or individuals. A composite action term may be composed of elementary action terms linked by means of, e.g., sequential and parallel composition.

Agent A computer program that can accept tasks from its human user, can figure out which actions to perform in order to solve these tasks and can actually perform these actions without user supervision, is an example of a *software agent*. More generally, any system that is capable of perceiving events in its environment, of representing information about the current state of affairs, and of acting in its environment guided by perceptions and stored information, is called an agent. If the environment is virtual, such as the Internet, we deal with software agents. If the environment is physical, we deal either with natural agents such as human beings and animals, or with artificial physical agents such as robots and embedded systems. The term agent denotes an abstraction that subsumes all these different cases.

Typical examples of software agents are web shopping assistants and life-like characters (artificial creatures) in computer games. Typical examples of artificial physical agents are the entertainment robot *Aibo* by Sony and the unmanned NASA space vehicle *Deep Space One*. It is expected that software agents capable to assist their users to cope with the increasing complexities caused by the accelerating and virtually uncontrolled growth of the World Wide Web will play a major role in the future.

The term *agent* is sometimes used as a synonym for *intelligent system*. But, in general, agents do not have to be 'intelligent'. In software engineering, for instance, the ability of an agent to communicate and cooperate with other systems in a flexible manner, and the ability of a mobile agent to migrate to another computer providing more resources via suitable network links, are considered more fundamental than any form of 'intelligence'.

The philosophical basis for the agent paradigm in computer science is the concept of *intentional systems* introduced by Daniel Dennett in [Den71] to characterize systems whose behavior can be best explained and forecasted by ascribing them beliefs, goals and intentions. Following Dennett, Yoav Shoham proposed in [Sho93] a mentalistic approach to model and program agents, called *Agent-Oriented Programming*. In this approach, the data structures of an agent program reflect basic mental components such as beliefs, commitments, and goals, while the agent's behavior is determined by reaction rules that refer to its mental state and are triggered by events, and possibly by its planning capabilities for pro-actively achieving its goals.

An important feature of agents is their ability to communicate and interact with each other. For artificial agents, communication is normally implemented by an asynchronous message passing mechanism. Agents created by different designers must speak the same agent communication language for expressing the type of communication act, and must refer to shared ontologies for being able to understand the contents of the messages of each other. Conversations between agents often follow a certain protocol that defines the admissible patterns of message sequences.

Similarly to the notion of *objects* in software engineering, the term *agent* denotes an abstraction that leads to more natural and more modular software concepts. While the state of an object is just a collection of attribute values without any generic structure, the state of an agent has a mentalistic structure comprising perceptions and beliefs. Messages in object-oriented programming are coded in an application-specific ad-hoc manner, whereas messages in agent-oriented programming are based on an application-independent agent communication language. An agent may exhibit pro-active behavior with some degree of autonomy, while the behavior of an object is purely reactive and under full control of those other objects that invoke its methods.

Agent Communication Language (ACL) An ACL defines a list of standard message types indicating the kind of communication act (such as

tell, ask, request, etc.) performed by sending a message of that type, and describes their semantics. An ACL message has the form $m(c)$ where m denotes the message type, and c denotes the message content. For expressing the message content, standardized vocabularies (often called ontologies) are used. The semantics of an ACL can be based on speech act theory, a philosophical theory of communication.

See also www.fipa.org where an industry standard ACL is defined.

Agent-Oriented Information Systems (AOIS)

represent a new information system paradigm where communication between different (software-controlled) systems and between systems and humans is understood as communication between agents whose state consists of mental components (such as beliefs, perceptions, memory, commitments, etc.). In enterprise information systems, for instance, the AOIS paradigm implies that *business agents* are treated as first class citizens along with business objects.

There is also an AOIS workshop series.

Business Rules are statements that express a *business policy*, defining or constraining some aspect of a business, in a declarative manner (not describing/prescribing every detail of their implementation). Business rules may be strict or defeasible (allowing exceptions). They can be formalized as integrity constraints, derivation rules, or reaction rules.

Business Transaction A sequence of actions performed by two or more agents, involving a flow of information and a flow of money, and normally also a flow of material or certain other physical effects. Usually, it requires some bookkeeping to record what happened. Today, this bookkeeping is done by the computer-based information systems of the involved business partners. Since an enterprise may participate in a great number of business transactions at the same time, this requires sophisticated information system technologies for guaranteeing high performance and consistency.

Contract Net Protocol In this coordination model, proposed in [Smi80], a given task to be per-

formed is broken into subtasks that can be performed concurrently. The various subtasks are announced to all other agents, and bids from agents capable to perform any of the announced subtasks are requested. The bids are collected, and a contract for a specific subtask is awarded to the best bidder.

CORBA The *Common Object Request Broker Architecture* is an established standard allowing object-oriented distributed systems to communicate through the remote invocation of object methods.

Database Management System (DBMS) The main purpose of a DBMS is to store and retrieve information given in an explicit linguistic format (using various symbols). As opposed to certain other types of information that are also processed in agents, this type of information is essentially propositional, that is, it can be expressed as a set of propositions in a formal language. In the sixties and seventies, pushed by the need to store and process large data sets, powerful database management systems extending the file system technology have been developed. These systems have been named *hierarchical* and *network* databases, referring to the respective type of file organization. Although they were able to process large amounts of data efficiently, their limitations in terms of flexibility and ease of use were severe. Those difficulties were caused by the unnatural character of the conceptual user-interface of hierarchical and network databases consisting of the rather low-level data access operations dictated by their way of implementing storage and retrieval. Thus, both database models have later on turned out to be cognitively inadequate. The formal conceptualization of relational databases by Codd in the early seventies rendered it possible to overcome the inadequacy of the first generation database technology. The logic-based formal concepts of the relational database model have led to more cognitive adequacy, and have thus constituted the conceptual basis for further progress (towards object-relational, temporal, deductive, etc. databases). Driven by the success of the object-oriented paradigm, and by the desire to improve the relational database

model, object-relational databases are now increasingly regarded the successor to relational databases. This development is being reflected in the progression of SQL, the established standard language for database manipulation, from SQL-89 via SQL-92 to SQL-99.

Data Warehouse A very large database that stores historical and up-to-date information from a variety of sources and is optimized for fast query answering. It is involved in three continuous processes: 1) at regular intervals, it extracts data from its information sources, loads it into auxiliary tables, and subsequently cleans and transforms the loaded data in order to make it suitable for the data warehouse schema; 2) it processes queries from users and from data analysis applications; and 3) it archives the data that is no longer needed by means of tertiary storage technology.

Most enterprises today employ computer-based information systems for financial accounting, purchase, sales and inventory management, production planning and control. In order to efficiently use the vast amount of information that these operational systems have been collecting over the years for planning and decision making purposes, the various kinds of information from all relevant sources have to be merged and consolidated in a data warehouse.

While an operational database is mainly accessed by OLTP applications that update its content, a data warehouse is mainly accessed by ad hoc user queries and by special data analysis programs, also called *Online Analytical Processing* (OLAP) applications. For instance, in a banking environment, there may be an OLTP application for controlling the banks's automated teller machines (ATMs). This application performs frequent updates to tables storing current account information in a detailed format. On the other hand, there may be an OLAP application for analyzing the behavior of bank customers. A typical query that could be answered by such a system would be to calculate the average amount that customers of a certain age withdraw from their account by using ATMs in a certain region. In order to attain quick response times for such complex

queries, the bank would maintain a data warehouse into which all the relevant information (including historical account data) from other databases is loaded and suitably aggregated.

Typically, queries in data warehouses refer to business events, such as sales transactions or online shop visits, that are recorded in event history tables (also called ‘fact tables’) with designated columns for storing the time point and the location at which the event occurred. Usually, an event record has certain numerical parameters such as an amount, a quantity, or a duration, and certain additional parameters such as references to the agents and objects involved in the event. While the numerical parameters are the basis for forming statistical queries, the time, the location and certain reference parameters are used as the *dimensions* of the requested statistics. There are special data management techniques, also called *multidimensional databases*, for representing and processing this type of multidimensional data. For Further Research, see [Cod94, AM97, IWK97].

Derivation Rules (or *deduction rules*) are used for defining intensional predicates and for representing heuristic knowledge, e.g. in *deductive databases* and in logic programs. Intensional predicates express properties of, and relationships between, entities on the basis of other (intensional and extensional) predicates. Heuristic knowledge is often represented in the form of *default rules* which may be naturally expressed using the weak and strong negation from partial logic (like in the formalism of ‘extended logic programs’). While relational databases allow to define non-recursive intensional predicates with the help of *views*, they do not support default rules or any other form of heuristic knowledge.

EDI Electronic Data Interchange, denotes the traditional computer-to-computer exchange of standard messages representing normal business transactions including payments, information exchange and purchase order requests. Besides the two main international standards for EDI messages, UN/EDIFACT and ANSI X.12, there are several vertical EDI standards. EDIFACT is administered by a working party (WP.4) of the United Nations Economic Commission

for Europe (UN/ECE). The EDIFACT syntax rules have been published by the ISO as ISO9735. In [Moo99], it is shown that current EDI standards have the message structure proposed by speech act theory. The current EDI standards are being criticized because of a number of problems such as underspecified meaning, idiosyncratic use and inflexibility. In 1999, a major initiative has been launched to replace the outdated EDI message syntax by a more flexible XML-based framework called *ebXML*.

Enterprise Application Integration (EAI) refers to the problem of how to integrate the increasing number of different application systems and islands of information an enterprise has built up over many years. The EAI problem also arises through the formation of a virtual enterprise or from merging two companies. While the integration of various islands of information including databases, sequential files, and spreadsheets, may be achieved through *data federation systems*, the interoperation between different application systems requires an asynchronous message exchange technology, also called *message-oriented middleware (MOM)*. In addition, a message translation service is needed to transform the messages sent by one application to the message language of another application. An application-independent EAI message language, called *Business Object Documents*, is proposed by the Open Application Group. The integration of applications across enterprise boundaries is also called ‘Enterprise Relationship Management’.

Enterprise Resource Planning (ERP) systems are generic and comprehensive business software systems based on a distributed computing platform including one or more database management systems. They combine a global enterprise information system covering large parts of the information needs of an enterprise with a large number of application programs implementing all kinds of business processes that are vital for the operation of an enterprise. These systems help organizations to deal with basic business functions such as purchase/sales/inventory (‘distribution’) management, financial accounting and controlling,

and human resources management, as well as with advanced business functions such as project management, production planning, supply chain management, and sales force automation. First generation ERP systems now run the complete back office functions of the worlds largest corporations. The ERP market rose at 50% per year to \$8.6 billion in 1998 with 22,000 installations of the market leader, SAP R/3. Typically, ERP systems run in a three-tier client/server architecture. They provide multi-instance database management as well as configuration and version (or 'customization') management for the underlying database schema, the user interface, and the numerous application programs associated with them. Since ERP systems are designed for multinational companies, they have to support multiple languages and currencies as well as country-specific business practices. The sheer size and the tremendous complexity of these systems make them difficult to deploy and maintain.

Entity-Relationship (ER) Modeling A conceptual modeling method and diagram language based on a small number of ontological principles: an information system has to represent information about *entities* that occur in the *universe of discourse* associated with its application domain, and that can be uniquely identified and distinguished from other entities; entities have *properties* and participate in *relationships* with other entities; in order to represent entities in an information system, they are classified by means of *entity types*; each entity type defines a list of (stored and virtual) *attributes* that are used to represent the relevant properties of the entities associated with it; together, the values of all attributes of an entity form the *state* of it; in order to represent ordinary domain relationships (or *associations*) between entities, they are classified by means of *relationship types*; there are two designated relationships between entity types that are independent of the application domain: *specialization* (subclass) and *composition* (component class).

ER modeling was introduced in [Che76]. In its original form, it included the *primary key* concept as its standard naming technique, but did not include specialization and composition.

The primary key standard naming technique proved to be inadequate since a standard name should be a unique identifier which is associated with an entity throughout its entire life cycle implying that it must be immutable. However, the basic idea of ER modeling does not depend on the primary key concept. It is also compatible with the *object identifier* concept of OO systems and ORDBs. This implies that ER modeling does not preclude the possibility of two distinct entities having the same state. It is therefore justified to view OO information modeling, such as UML class diagrams, as inessential extensions of ER modeling, and to regard ER modeling as the proper foundation of information modeling.

Information System (IS) An IS is an artifact (or technical arrangement) for efficiently managing, manipulating, and evaluating information-bearing items such as paper documents, ASCII text files, or physical objects. Today, especially in enterprises and other large organizations, there are more and more computerized ISs implemented by means of DBMS technology. One may distinguish between private, organizational and public ISs. Typical examples of a private IS are personal address databases and diaries. The major paradigms of an organizational IS are transaction-oriented database (OLTP) applications (such as ERP systems) and query-answering-oriented data warehouse (OLAP) applications. Typical examples of a public IS are libraries, museums, zoos, and web-based community ISs.

Integrity Constraints are sentences which have to be satisfied in all evolving states of a database (or knowledge base). They stipulate meaningful domain-specific restrictions on the class of admissible databases (or knowledge bases). Updates are only accepted if they respect all integrity constraints. The most fundamental integrity constraints are value restrictions, keys and foreign keys (or referential integrity constraints).

Interoperability denotes the ability of two or more systems to collaborate. At a lower level, this concerns the ability to exchange data and to allow for remote procedure calls from

one system to another. At a higher level, it requires the ability to participate in the asynchronous exchange of messages based on an application-independent language (such as KQML, or FIPA-ACL).

Knowledge Interchange Format (KIF) is a computer-oriented language for the interchange of knowledge among disparate programs developed by the ARPA-sponsored Knowledge Sharing Effort. It has declarative semantics (i.e. the meaning of expressions in the representation can be understood without appeal to an interpreter for manipulating those expressions); it is logically comprehensive (i.e. it provides for the expression of arbitrary sentences in the first-order predicate calculus); it provides for the representation of knowledge about the representation of knowledge; it provides for the representation of nonmonotonic reasoning rules; and it provides for the definition of objects, functions, and relations. [from UMBC Agent Web]

KQML The *Knowledge Query and Manipulation Language*, developed within the U.S. research project 'Knowledge Sharing Effort', is a language and protocol for exchanging messages. It focuses on an extensible set of message types, which defines the permissible communication acts.

Message-Oriented Middleware (MOM) denotes a type of software systems for managing transactional message queues as the basis of asynchronous message passing. Well-known products include IBM MQSeries and Sun JMQ. A standard MOM application programming interface for Java, called *Java Messaging Service (JMS)* has been proposed by Sun.

Message Transport An abstract service provided by a MOM system in the case of EAI, or by the agent management platform to which the agent is (currently) attached in the case of a FIPA-compliant interoperability solution. The message transport service provides for the reliable and timely delivery of messages to their destination agents, and also provides a mapping from logical names to physical transport addresses

Mobile Agent An agent that does not depend upon the agent platform where it began executing but is able to migrate to another agent platform.

Multiagent Systems (MAS) represent a new paradigm in distributed computing. Its characteristics are: a) there is no global system control; b) information is decentralized; and c) each participating agent has incomplete information and limited capabilities. One of the first models for dynamically allocating tasks in a MAS, is the Contract Net Protocol.

OLAP Application Online Analytical Processing applications allow to evaluate large data sets by means of sophisticated techniques, such as statistical methods and *data mining* techniques. They typically run on top of a data warehouse system.

OLTP System Online Transaction Processing systems are able to process a large number of concurrent database query and update requests in realtime. The information technology part of a business transaction is called an *online transaction*, or simply 'transaction'. It is performed through the execution of an application program that accesses one or more shared databases within the business information system. A transaction is a complex update operation consisting of a structured sequence of read and write operations. Ideally, a transaction satisfies the ACID properties. Business information systems are primarily OLTP systems. In almost every sector – manufacturing, education, health care, government, and large and small businesses - OLTP application systems are relied upon for everyday administrative work, communication, information gathering, and decision making. The first OLTP application in widespread use was the airline reservation system SABRE developed in the early 1960s as a joint venture between IBM and American Airlines. This system connects several hundred thousand nodes (user interface devices) and has to handle several thousand update request messages per second

Object-Relational Databases (ORDBs) have evolved from relational databases by adding several extensions derived from conceptual

modeling requirements and from object-oriented programming concepts. One can view the evolution of relational to object-relational databases in two steps. First, the addition of abstract data types (ADTs) allows *complex-valued tables*. ADTs include user-defined base types and complex types together with user-defined functions and type predicates, and the possibility to form a type hierarchy where a subtype of a tuple type inherits all attributes defined for it. Second, the addition of object identity, object references and the possibility to define subtables within an extensional subclass hierarchy allows *object tables*. There are two notable differences between object-relational databases and object-oriented programming. First, object IDs in ORDBs are logical pointers. They are not bound to a physical location (like C++ pointers). Second, in addition to the intensional subtype hierarchy of the type system, ORDBs have an extensional subclass (or subtable) hierarchy that respects the subtype relationships defined in their type system. ORDBs allow the seamless integration of multimedia data types and large application objects such as text documents, spreadsheets and maps, with the fundamental concept of database tables. Many object-relational extensions have been included in SQL-99.

Object-Oriented Database (OODB) Historically, the successful application of object-oriented programming languages such as Smalltalk, C++ and Java, has led to the development of a number of so-called ‘object-oriented database systems’ which support the storage and manipulation of persistent objects. These systems have been designed as programming tools to facilitate the development of object-oriented application programs. However, although they are called database systems, their emphasis is not on representing information by means of tables but rather on persistent object management. Any database concept which is intended as an implementation platform for information systems and knowledge representation must support tables as its basic representation concept on which query answering is based. Tables correspond to extensional predicates, and each table row corresponds to a proposition. This

correspondence is a fundamental requirement for true database systems. If it is violated, like in the case of OODBs, one deals with a new notion of database system, and it would be less confusing to use another term instead (e.g. persistent object management system) as proposed by [Kim95].

Ontology An ontology explicitly specifies the terms for expressing queries and assertions about a domain in a way that is formal, objective, and unambiguous. This includes the stipulation of terminological relationships and constraints in order to capture key aspects of the intended meaning of the specified terms. At least two sorts of expressions should be distinguished in an ontology: propositions and actions. An ontology is implicitly defined by a conceptual model (such as an ER or UML model). Communication between agents can only be successful if it is based on a common (or shared) ontology.

Proposition Something which can be believed (or asserted, asked, denied, etc.) by an agent. Also called *statement* or *sentence*. In a formal language, an atomic proposition may have the form $p(t_1, \dots, t_n)$ where p is a predicate and t_1, \dots, t_n are entity terms (in the simplest case constant symbols). A complex proposition may be expressed on the basis of atomic propositions by means of logical connectives (such as negation, conjunction, disjunction, etc.) and by means of (existential and universal) quantifiers in combination with entity variables.

Protocol A protocol defines the admissible patterns of a particular type of conversation or interaction between agents. Notice that an interaction protocol refers to the communication acts and high-level actions available to agents, whereas a networking protocol refers to message transport mechanisms such as TCP/IP.

Reaction Rule SQL databases support a restricted form of reaction rules, called *triggers*. Triggers are bound to update events. Depending on some condition on the database state, they may lead to an update action and to system-specific procedure calls. In [Wag98] a general form of reaction rules, subsuming production rules and database triggers (or ‘event-condition-action

rules’) as special cases, was proposed. Reaction rules can be used to specify the communication in multidatabases and, more generally, the interoperation between communication-enabled application systems.

Relational Database (RDB) Already in 1970, Edgar F. Codd published his pioneering article “A Relational Model of Data for Large Shared Data Banks” in the *Communications of the ACM*, where he defined the principles of the relational database model. This was the first convincing conceptualization of a general purpose database model, and it is not an accident that it relies on formal logic providing a clear separation of the conceptual user interface and the underlying implementation techniques. In the mid-eighties, IBM presented *DB2*, the first industrial-strength implementation of the relational model, which continues to be one of the most successful systems today. There are now numerous other relational DBMSs that are commercially available. The most popular ones include Informix, Oracle, Sybase and Microsoft SQL Server. To a great extent, the overwhelming success of these systems is due to the standardization of the database manipulation language SQL originally developed at IBM in the seventies.

While most well-established information processing systems and tools such as programming languages, operating systems or word processors have evolved from practical prototypes, the unprecedented success story of the relational database model is one of the rare examples where a well-established and widely used major software system is based on a formal model derived from a mathematical theory (in this case set theory and mathematical logic). Conceptually, a relational database is a finite set of finite set-theoretic relations (called ‘tables’) over elementary data types, corresponding to a finite set of atomic propositions. Such a collection of atomic sentences can also be viewed as a finite interpretation of the formal language associated with the database in the sense of first order predicate logic model theory.

The information represented in a relational database is updated by inserting or deleting atomic sentences corresponding to table rows

(or tuples of some set-theoretic relation). Since a relational database is assumed to have complete information about the domain represented in its tables, if-queries are answered either by yes or by no. There is no third type of answer such as unknown. Open queries (with free variables) are answered by returning the set of all answer substitutions satisfying the query formula.

Rule There are various types of rules: business rules, legal rules, calculation rules, derivation rules, production rules, rules of thumb, reaction rules, and many more.

Speech Act Theory A philosophical theory of communication proposed in [Aus62, Sea69]. Speech act theory is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements, but also performs actions. A speech act can be put in a stylised form that begins “I hereby request” or “I hereby declare”. In this form the verb is called the performative, since saying it makes it so. Verbs that cannot be put into this form are not speech acts, for example “I hereby solve this equation” does not actually solve the equation. In speech act theory, a communicative act is decomposed into a locutionary, an illocutionary and a perlocutionary act. A locutionary act refers to the formulation of an utterance, an illocutionary act refers to a classification of the utterance from the speaker’s perspective (e.g. query, command, promise, etc), and a perlocutionary act refers to other intended effects on the hearer, such as updating the hearer’s mental state. Speech act theory counts as the basis of ACLs.

SQL is a declarative language for defining, modifying and querying database tables. A table schema is defined with the command `CREATE TABLE...` and modified with `ALTER TABLE...` The content of a table can be modified by either adding, deleting, or changing rows using the commands `INSERT INTO...`, `DELETE FROM...` and `UPDATE...` Simple queries are formed with the expression `SELECT columns FROM tables WHERE condition`. Such a query combines the cross product of *tables* with the selection defined by *condition* and the final projection to the attributes oc-

curing in *columns*. More complex queries can be formed by nesting such SELECT statements (using subqueries in the WHERE clause), and by combining them with algebraic operators such as JOIN, UNION, EXCEPT. SQL queries correspond to relational algebra expressions and to predicate logic formulas: projection corresponds to existential quantification, join to conjunction, union to disjunction, and difference (EXCEPT) to negation. The most recent version of SQL, SQL-99, includes many object-relational extensions, such as user-defined types for attributes, object references, and subtable definitions by means of CREATE TABLE *subtable* UNDER *supertable*.

TCP/IP is a networking protocol used to establish connections and transmit data between hosts.

Unified Modeling Language (UML) is an established object-oriented modeling standard defined by an industry initiative organized and funded by Rational and led by three prominent figures of the OO modeling community: Booch, Jacobson, and Rumbaugh. UML recognizes five distinct modeling views: the *use-case* view for requirements analysis, the *logical view* for describing the static structure and the behavior of a system, and three implementation views regarding *components*, *concurrency*, and *deployment*. Each of these views is composed of several diagrams. A *use-case diagram* depicts a complete sequence of related transactions between an external actor and the system. The idea is that, by going through all of the actors associated with a system, and defining everything they are able to do with it, the complete functionality of the system can be defined. UML *class diagrams* are a straightforward extension of ER diagrams. In addition to conventional (stored) attributes, class diagrams also list the operations of a class which may be functions (derived attributes) or service procedures associated with the class. The behavior of a system is modeled by means of four types of diagram: *sequence diagrams* depict the message exchange between objects arranged in time sequence, where the direction of time is down the page; an alternative way of visualizing the message exchange between objects is offered

by *collaboration diagrams* emphasizing the associations among objects instead of the time sequence; *activity diagrams* are used for describing concurrent, asynchronous processing; finally, *state charts* allow to represent the state transitions of a system.

References

- [AM97] S. Anahory and D. Murray. *Data Warehousing in the Real World*. Addison Wesley, 1997.
- [Aus62] J.L Austin. *How to Do Things with Words*. Harvard University Press, Cambridge (MA), 1962.
- [Che76] P. Chen. The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.
- [Cod94] E.F. Codd. Adding value to relational and legacy dbms: The olap mandate. *Business Intelligence*, 1994.
- [Den71] D.C. Dennett. Intentional systems. *The Journal of Philosophy*, 68, 1971.
- [IWK97] W.H. Inmon, J.D. Welch, and G.L. Katherine. *Managing the Data Warehouse*. Wiley & Sons, New York, 1997.
- [Kim95] W. Kim. Introduction to part 1. In W. Kim, editor, *Modern Database Systems*, pages 5–17. ACM Press, New York, 1995.
- [Moo99] S.A. Moore. Categorizing automated messages. *Decision Support Systems*, 1999.
- [Sea69] J.R. Searle. *Speech Acts*. Cambridge University Press, Cambridge (UK), 1969.
- [Sho93] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
- [Smi80] R.G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

- [Wag98] G. Wagner. *Foundations of Knowledge Systems – with Applications to Databases and Agents*, volume 13 of *Advances in Database Systems*. Kluwer Academic Publishers, 1998. See <http://www.inf.fu-berlin.de/~wagnerg/ks.html>.