Tools for Medical Informatics

Robert Hindel

Informatics uses words, terms and expressions of various scientific disciplines. The proposed tools, hermeneutics and phenomenology, generate a basis for quality control by establishing the authenticity and validity of such expressions. Without such tools there is the danger that poorly defined expressions obscure true meaning and prevent progress. The method is demonstrated on "objects" as used in "object oriented programming" and on "open systems" as used in the International Standards Organization model for "open system interconnection."

KEY WORDS: Informatics tools, OSI model, object oriented programming, medical informatics.

MEDICAL INFORMATICS, although not fully defined in its scope, has recently attracted new attention because of "harmonizing" efforts between European and American developments. The term "harmonizing" tries to avoid the coercive connotation of "coordinating." The intended outcome of harmonizing is a substantial agreement between European and American methods and regulations with respect to medical equipment commerce.

Informatics could be described as a systematic approach to making relevant information available, to organize, and to clarify it. Informatics uses language as an information carrier. The language is often English and contains computer terms, medical terms, and special terms that may require explanation. Communication at professionals meetings, particularly at international meetings suffer, due to frequent misunderstanding of terms because linguistic tools of "quality control" are neglected. But even if the intention to communicate exists other group dynamics interfere. "Obviously we speak to communicate but also to conceal, to leave unspoken," says Steiner[1] and "natural selection favor the contriver." Personal goals motivate the contriver to choose certain terms for gaining political or economic advantages. Marketeers who aim to prevail and achieve supporting consensus use buzzwords that hide, rather than convey, meaning.

The following discussion describes and shows "informatics tools" that are suitable for improving the quality of communication. These tools are not new. As a matter of fact, these tools or practices are used in law, psychotherapy, historical research, and other endeavors where precision of language is important. Physicists and engineers, by contrast, have a low esteem for language. While the physicist invents expressions like "quarks with flavor" and indicates the arbitrary aspect of such word coinage, the engineer uses terms like "open systems," and "object oriented programming" (OOP) and offers little clarification for the uninitiated. The Informatics tools that I will discuss are based on philosophical disciplines. They have cumbersome names and are burdened by centuries of learned disputes, but they are applicable nevertheless to modern Informatics.

HERMENEUTICS

Hermeneutics is a discipline that interprets and evaluates what is written. It researches original texts and subsequent interpretations. It also tries to find a translation from the world of the originator into the present world of the user. For instance for a discussion of the "entity–relationship model" it is useful to read Chen's article.[2] Any new use and interpretation should be related to the meaning of the original term. Similarly, in any discussion about "relational databases" Codd's seminal article[3] is recommended and informative. Hermeneutics establishes an authority of meaning that is essential for clarity of communication. If various undefined meanings of the same term are used in a discussion, misunderstandings will develop and persist for a long time. Put simply, Hermeneutics asks the following questions: What is the authentic meaning of the term? Which authority will we agree to accept? and How does our understanding differ?

PHENOMENOLOGY

Phenomenology, the other discipline, tries to establish the essence of the entity in question.

From R. Hindel, PhD, Orange, CT.
No reprints available.

0897-1889/92/0502-0002$03.00/0

Like logic and computer programming it must be practiced in order to be fully understood, but some examples may be instructive. Applying the tools of phenomenology means discovering and describing "reality." An essential start is suspension of judgment. We all tend to categorize quickly so that we can proceed with more important activities, but phenomenology requires keen attention to the "given." Explaining presupposes understanding whereas describing admits some ignorance. Put differently, phenomenology evaluates whether proposed terms fit the situation. Experimental physics is based on such skill; so are preparation of a legal case, detective work, and other investigative activities. Phenomenological practice is irritating to those who want a situation resolved quickly. A very useful introduction to phenomenology is given by Ihde.[4]

The foregoing disciplines are complementary in the sense that hermeneutics is often an essential starting point for phenomenology. In an actual case the researcher collects articles about the subjects and the meaning of the terms in question. Next the seminal article or articles are found and read and finally, the researcher may or may not reach a new understanding and differentiate it form the "authority." This could require a great deal of work and for most purposes it would suffice to quote an established authority. A useful and constructive service would be a list of terms with quotations and glossaries.

In the following I should like to demonstrate the tools of hermeneutics and phenomenology on two terms: "objects" and "open systems," both very much in vogue and both frequently misunderstood and misquoted.

OBJECTS

Object oriented programming is an important and powerful new concept, implemented in such programs as C++ and SmallTalk. What used to be called "data" is now called "objects" and the relationship between the two terms is not easily understood. We read that objects are "self describing data structures" and that objects are "protected structures" but then we find such statements as, "Data are stored inside of an object and are accessible only through messages."[5] But another publication states that objects are analogous to pieces of data in other languages.[6] So it seems, that "objects" are both data and something containing data. An authority on the subject, Bjarn Stroustrup, who developed C++, which is based on C,[7] says simply "an object is a region of storage."[8]

This apparent contradiction, namely that "objects" are "data" and also a "region of storage," ie, an address of data, can be resolved by comparing the computer's handling of data with our human practice. If I use Basic—an easily understood, yet powerful language—and type "a = 3: b = 4: print a + b" and I get, correctly "7," what has happened inside the computer is allocation of the number "3" to an address "a" and allocation of the number "4" to an address "b." The operation "+" in the expression "print a + b" is understood by the computer as "take the content of address 'a' add it to the content of address 'b' and print the sum." For a computer no information or data exist unless they are located or stored somewhere. Those familiar with computer programming know that for any defined variable (such as "a" and "b") an address can be found where the value of the variable is stored. Casual language will not distinguish the content from the address and just say "a" or "variable a."

Our own way of calculating functions is different. If we say "take 3 and add 4" we could not possibly identify where in our brain "3" and "4" are located. Somehow our short-term memory retains this information and we also "know" what "add to" means and we—trained in algebra—find the answer.

Back to "objects." We now know, that they are equivalent to data (and data storage) but with some new aspects: object oriented programming lists these aspects as "instances," "abstraction," "encapsulation," "inheritance," and "polymorphism." Conventional understanding of these terms is of only limited use. Patient efforts are required to understand what the creators of SmallTalk (or other object oriented language) mean by these terms. In Pinson and Wiener[5] we read that "all objects are instances of a specific class" and "all the data and functional abstractions are defined in its class description protocol." Also "functional abstractions take the form of methods that give the details of how an object is to respond to mes-

sages." For instance, a command ("message") such as "convert to upper case" is applicable and correct for a string (text) but incorrect for an integer variable. It would not be included in the functional abstraction of the class "integer" (speaking the OOP language). "Inheritance" means that a member of a class (called "instance") will also have essential characteristics of the class, just as we would assume from our knowledge of biology. Specifically, "Subclasses inherit the properties of their superclasses," and "Objects are created by sending instance creation messages to the class name." (Here again "class name" is equivalent to "class" and to "address of class.")

These few citations suggest that a translation into conventional English would be much appreciated. The intentions are significant but the linguistic expression of these intentions leaves much to be desired. For purposes of "informatics tools" one could summarize that object-oriented programming is an orderly approach to defining data structures. But if data groups were defined a year ago in a document, and if the same data groups are now elevated to "objects," one is inclined to ask what has been added. If nothing was added, an understandable term was replaced by a buzz word.

OPEN SYSTEMS

Systems are like rigid machinery, insensitive and inflexible. "Open systems" on the other hand communicate, are accessible, and permit individuality. When we hear that a system does not work (eg, the health care system) we are reminded of a Kafkaesque situation in which the victim is at loss as to what is in store for him but is aware of pending doom. After the Challenger disaster a prestigious panel concluded that the "system did not work." Only one courageous physicist pointed out that gross negligence and carelessness of certain people was responsible, not a faceless system. Computer systems are normally not "open" and an open system is something worthy of closer attention. A major computer company sponsors the ad, "Freedom to choose, power to use, the open advantage." Twenty years ago the same company shocked IBM by explaining to uninitiated engineers how computers work. At that time it provided openness rather than buzzwords.

The seminal paper on open system interconnection (OSI) was written by Zimmerman in 1980[9] and is a tutorial on how heterogeneous computer systems could cooperate. The model for Zimmerman was ARPANET, a US computer network that permits computers of various origins to communicate and to use other computer's capabilities cooperatively. A computer at the California Institute of Technology could, under this scheme, request a computer at the Massachusetts Institute of Technology to process data according to a powerful algorithm and send the resulting files back to Cal Tech. The computer organization that permits such cooperation was named "open." Since then the "seven layers" suggested by Zimmerman[9] as necessary stages of moving a message from one computer to another have turned into rigid requirements of openness. Before analyzing the original document of system openness, let us review the conventional uses of the term "open."

An open meeting does not restrict attendance and permits expression of opposing opinions. An open house invites inspection of furnishings and layout without subjecting the visitors to high-pressure sales methods. An open society will permit free movements and questions by visitors. Even criticism of the officials, to some extent, is permitted. In all these examples of openness there is an implicit set of rules, a code of good conduct, that must not be violated. The visitor of an open house must not lie down on the display couch; the participant in an open town meeting must not injure others or damage property.[10] One of the unsettling experiences when being invited into an unfamiliar and apparently "open" situation results from an ignorance of such rules. Only very naive persons will assume that there are no rules. Openness exists as a tentative set of rules. The participants agree implicitly to a refined behavior and elevated maturity. There is no doubt, however, that the host has the power of terminating the game and removing the violator.

What does this mean for computer systems? Zimmerman[9] states that the term open was chosen to emphasize that by conforming to those international standards a system will be open to all other systems obeying the same

standard throughout the world. A later ISO publication[11] is more specific: "only external behavior of open systems is retained as the standard of behavior for real open systems." However, a real open system need not be implemented as described by the OSI model. The external behavior, though, presupposes an internal architecture that is functionally described by the seven layers.[11]

A brief description of the layers may be informative.

1. *Layer seven,* the top layer, is the application layer. For instance, if system A wants to have system B perform image compression, it will specify this application in layer seven.
2. *Layer six* is presentation—the form of the data, or the structure. This layer specifies how the supplied information is organized.
3. *Layers five to one* are different stages of the communication process. Layer five, the session layer, specifies how the connection with the other system is established and which protocol must be used to carry out the transfer. The lowest layer is the physical layer, ie, wire, optical fiber, airwaves, etc, the physical transmission medium.

Much has been written about these seven layers and a particularly detailed and lucid description was written by Tanenbaum,[12] but for our purposes these layers are stages in a process of communication. It is important to note, that the message, ie, the structured data that are sent, must match these stages.

I shall try to describe the process with an example. A company in New York prepares to ship a large piece of equipment to a company in Duesseldorf, Germany. Nobody in the German company reads English (a rare exception). The New York company finds a translator who prepares a description of the contents of the shipment. It turns out that the equipment must be disassembled into smaller packages. A document describes the identification of parts and the assembly. The New York Port Authority requires certain documents detailing the content, destination, etc. At the port of arrival, Duesseldorf, an inspection will take place and a document must be prepared. Custom officials in Germany will require certain documents for declaration of value and payment of import duty. The expediter in Germany needs papers in order to handle the transport from the airport to the premises of the receiving company. All these papers must accompany the individual packages that constitute the total shipment. As the packages pass one station after another, the corresponding documents are opened and processed. Finally, at the destination all packages are opened, the instructions read, and the equipment assembled. Like the seven layers of the OSI model we have stages of the overall process. A correspondence exists between the sender and the receiver in terms of understanding what should be done. Layer seven is rudimentary and says, "Here is the equipment you have ordered." Layer six is more elaborate; it is the translation from English into German and the overall explanation of the packages. Furthermore, in Germany it may be necessary for a bilingual expert to correct some of the wrong translations.

The overall transport is layer five, which contains the description of re-assembly, down to layer one, specifying the plane that flies the packages across the Atlantic.

Eleven years have passed since Zimmerman[9] wrote his paper. Today robust and well designed networks exist for computer to computer communication. Tanenbaum[12] described what is needed for a computer network in great details. Serial communication, which is available for all computers, can move data reliably between heterogeneous computers. An example is Compuserve, which permits uploading and downloading of all kinds of files and data structures. The "external behavior" is that of an open system, although I have never heard this term in connection with Compuserve.

Network communication (local area network [LAN] and wide area network [WAN]) can be considered a service performed for data transfer. The OSI model correctly identifies several conditions for successful transfer: The data structure at the sender must be organized into packets suitable for the transport mechanism, a physical transport medium must exist (wire, optical cables, air, etc); and reassembly at the receiver must guarantee data integrity. Once received correctly the "meaning" of the data must be understood, which may require a decoding process (layer 6). Finally, it must be evident

to the receiver what should be done with the data (layer 7). Control words, a message, or free text can be used to convey the application, but sender and receiver must speak the same language. If Marcus Hess, the hunter in Clifford Stool's computer spy story *The Cuckoo's Egg*[13] had not known how to talk to a VAX VMS 4.5, he could not have extracted information despite his luck with guessing passwords. In my opinion the OSI model consists of two parts: the "top layers" that present information (layer 6), and instructions for the use of such information (layer 7). This information is conveyed in the form of data that are transported from one location to another. The system where the information in question first resides is assumed to be different (heterogeneous) from the receiving system. The basic question that Zimmerman[9] addressed is, "How can heterogenous and physically distant computers cooperate?" Much of the complexity of the OSI model is associated with the transport of the information and the underlying model of a network for public telecommunication. Since 1980 when Zimmerman presented the OSI proposal, much progress has been made in telecommunication, and robust and reliable networks are available.

Turning to medical informatics the question is, "To what extent is the environment that is implicit in the OSI model applicable to a radiology department?" Granted much work has been done on the perfection of the OSI model. Nobody can object to "open" and everybody knows that we are dealing with "systems." Therefore, an "open system" seems to be the obvious choice. The danger is that terms like "open system," if not explained correctly, could obscure rather then illuminate the true requirements.

Up to this point we have applied hermeneutics. We have examined the definition of terms like "open," "heterogeneous systems," "application," etc, based on seminal documents. But we have no corresponding documents about the meaning of picture archiving communication systems (PACS). It is generally assumed that an essential function of PACS is reliable and fast access to images for diagnostic purposes. It is unclear whether PACS is a homogeneous system. On the one hand commercial systems have been delivered by system's integrators, and some purchase orders require even turn-key systems that include site preparation and training. On the other hand, integration of separate components into academic PACS has been successful. The system's integrator will prefer to use basic building blocks in form of computers by one vendor. Devices such as laser hard-copy units and laser film scanners can be connected without a complicated interface. Archives in form of magnetic or optical disk drives can also be readily connected. Although different interface protocols are used, the problems of interoperability have been worked out. It is not necessary to adhere to a general interface.

The situation is different with digital imaging modalities. These complex and expensive systems were developed specifically for the medical diagnostic market. All these modalities accomodate cathode-ray tube (CRT) displays, and all supply, in addition, removable storage for long-term archiving of images. The specific form in which data are written to such removable media is not standardized and not willingly disclosed.

Because the quality of the images used in a PACS depends on the quality of the supplied images, digital image data are important. Using the findings of the phenomenological exercise above, a radiology department lacks openness with respect to digital image data. It is conceivable that a number of input methods will be used depending on the modality manufacturer as long as the specific data format is disclosed.

The American College of Radiology–National Electrical Manufacturers Association (ACR-NEMA) standard addresses the problem of unified image data access and communication. The first version of the ACR-NEMA document was published in December 1985[14]; a second version in 1988 and this year version 3.0 will be ready for balloting.[15] Several excellent publications exist.[16,17] The latest version accepts ISO accredited networks as means of data transfer. This eliminates the complication of the first version, which required a special NIU (Network Interface Unit) as gateway between the parallel point-to-point ACR-NEMA physical interface and a commercially supported

network. Version 3.0[15] assumes that a robust network connection exists.

It is still unclear whether implementation has become simpler. Both command structure and data sets have grown in size and complexity and may not be vital for routine operations in a department. It should be remembered that it is the access to images and not to details of the examination that is expected from a PACS. If such information were important, it could be supplied with film images also. The resolution of films is certainly high enough to accomodate some additional thousands of bytes for text. If scientific studies require elaborate information about the examination, a radiology information system (RIS) can supply these.

For a further development of the ACR-NEMA standard as well as for standardization efforts in general, a careful analysis of what terms mean and whether they apply to the topic in question will be useful. The supplied brief examples of hermeneutics and phenomenology as formal disciplines are relevant in this respect.

Another area of potential application for the suggested tools is standardization in health-care informatics. Recently the American National Standards Institute, (ANSI), in an ambitious effort to coordinate standards, suggested formation of a planning panel comprised several organizations. A meeting on March 11, 1991, and a later meeting on August 29 disclosed widespread interest in this undertaking, but with vastly divergent expectations. Several organizations expected valuable data bases for their own purposes (Veterans Administration, Department of Defense, Social Security Administration, and others) while several standards writing organizations like Institute of Electric and Electronic Engineers (IEEE), American Society for Testing of Materials, High Level Seven, Computers and Business Equipment Manufacturing Association, Health Industry Manufacturing Association, and National Electrical Manufacturing Association hoped to become dominant authors of evolving standards. The European organization Comité Européen de Normalization, charged by the EEC with writing a European Medical Informatics standard, was also represented. It became obvious that vastly different interpretations of concepts and requirements exist, yet so far no effort has been made to develop a common knowledge base. Hermeneutics and phenomenology could generate such a base. The first step would be to generate a list of terms and concepts that are considered important or relevant for medical informatics. Next this list would be broken down into categories according to speciality and assigned to experts for hermeneutic evaluation. As explained above the relevant terms would now be interpreted according to "authorities" and the intended use reviewed. As a result, some terms would be redefined, others possibly replaced. Phenomenology would apply an analysis of the environments in which these terms and concepts will be applied.

REFERENCES

1. Steiner G: After Babel. New York, NY, Oxford, 1975
2. Chen PP: The Entity-Relationship Model. Toward a Unified View of Data. ACM 1:9-36, 1976
3. Codd EF: A Relational Model of Data for Large Shared Data Banks. ACM 13:377-387, 1970
4. Ihde D: Experimental Phenomenology. New York, NY, Paragon, 1979
5. Pinson LJ, Wiener RS: An Introduction to Object Oriented Programming & Smalltalk. Reading, MA, Addison-Wesley, 1988
6. Digitalk Inc, "Smalltalk/V," Los Angeles, CA, 1987
7. Kernighan BW, Ritchie DM: The C Language. Prentice-Hall Software Series 1978
8. Stroustrup B: The C++ Programming Language. Reading, MA, Addison-Wesley, 1986
9. Zimmermann H: OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection IEEE 28:425-432, 1980
10. Hindel R: Open System Characteristics. Proc CAR 1991. New York, NY, Springer-Verlag, 1991, pp 897-903
11. ISO-7498 Information Processing Systems—Open System Interconnection Basic Reference Model 1984
12. Tanenbaum A: Computer Networks. Englewood Cliffs, NJ, Prentice-Hall, 1981
13. Stoll C: The Cuckoo's Egg. New York, NY, Doubleday, 1989
14. ACR-NEMA: ACR-NEMA digital imaging and communications standard 300-1985. Washington, DC, National Electrical Manufacturers' Association, 1985
15. NEMA Draft version 3.0 DICOM 3.0 1991
16. Spilker C: ACR-NEMA digital imaging and communications standard; A non-technical description. J Digit Imag 2:127-131, 1989
17. Horii SC, et al: An Update on the ACR-NEMA standard activities. JDI 3:146-151, 1990