Transactions on Information and Communications Technologies vol 9, © 1994 WIT Press, www.witpress.com, ISSN 1743-3517

# Simplicity, uniformity and quality - the role of semantic analysis in systems development

K. Liu,[a] Y. Ades[b] & R. Stamper[c]
[a]*School of Computing, Staffordshire University, Stafford, ST18 0AD, UK*
[b]*Department of Computing and Information Technology, Greenwich University, UK*
[c]*Department of Information Management, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*

ABSTRACT

Industrial experience shows that heavy investment in IT does not always bring in a satisfactory economic return due to very often quality problems, which was partly caused by inadequacy of user requirements analysis and representation. Semantic Analysis, a method developed for analysing and specifying user requirements based on an 'agent-in-action' ontology, has demonstrated a way to improve software quality. The semantic structure is preserved through design and implementation in order to produce a satisfactory system. To this end, a set of methods and tools for systems engineering has be developed around Semantic Analysis and supported by Normbase. This approach has demonstrated several advantages in systems engineering such as concision of requirements representation, drastic reduction of documentation, and low cost of maintenance.

## INTRODUCTION: SEMIOTICS AND SOFTWARE ENGINEERING

There are many recognised paradigms of systems development: the waterfall approach, exploratory programming, prototyping, formal transformation and system assembly from reusable components (cf. [23]). The first three of these approaches are all currently used for practical systems development. The last two are less applied and are, especially the last one, in development stages. Very often one approach would be adopted supplemented by others in order to get benefit from each of them.

Semiotics [17,18,21] applied to computer systems development opens up another promising perspective. Andersen [1,2] presents an example of constructing computer systems based on such a paradigm. Patterns of human communications and use of languages are increasedly regarded as one of the foundations for computer systems design, which is witnessed for example in HCI (cf. [9]). The semiotic approach developed in the MEASUR research programme, which is now based on University of Twente, has offered a set of theories and methods for

information systems development [27]. A computer system in this paradigm is viewed as a part of human communication system. In studying the broad communication system and computerising a part of it, the important invariants are signs that are used by human agents with or without using computers. To construct a computer system involves understanding and representing communications among people. The engineering of a computer system is to produce a subsystem that can process signs as required by the agents. The functionalities of such a computer system, as a part of a larger communication system, is determined by the social and business needs of the agents in order to fulfil their objectives in the most effective manner.

We can study signs from six aspects, as Stamper pointed out in his semiotic framework (for a recent, detailed account see [26]). At a very first level *Physics*, signs have physical properties such as shape, density, legibility, and so on. *Empirics* is concerned with the variety, noise, entropy, redundancy, coding, etc. which have to be considered for the purposes of reliable data storage and signal transmission. The next level is *Syntactics* where the structure of a language, components of a data record and file structures will be studied. Designing of man-machine interfaces is closely related to an understanding of the empirics of sensory perception and the syntactic structure of a human language ([9], p252). The most essential aspect, the meanings of sign is studied in *Semantics*. At this level, people begin to think about the use of signs rather than the representation of signs where sentences, composed of signs according to certain syntactic rules, are used to make meaningful utterances. The links between actions and signs are established as meanings are assigned to the use of signs. The analysis at this level is to understand and represent the relationships between signs used and actions performed by people. The next aspect is the use of signs in context. Intentions are embedded in communications as conversations or negotiations are carried out. Speech acts are performed in a certain sequence following certain norms, for example cultural or business norms. These issues are studied in *Pragmatics*. The *Social level* is the last one in which signs can be studied where social effects of use of signs are the foci of the attention. Through the use of signs interpersonal relationships and social obligations can be created and altered.

In any business organisation, people use signs for communication to interact with each other purposefully. A communication process involves all the six aspects of signs. This large, complex organisation is an ontological system where the designing of a computer-related system is part of a "larger perspective of ontological design" [31]. Figure 1 schematically represents our point of view that has been applied in many of our systems designs.

## RELATIONSHIP AMONG SOFTWARE QUALITY, PRODUCTIVITY AND REQUIREMENTS ENGINEERING

Issues of software quality and engineering productivity are closely interrelated.

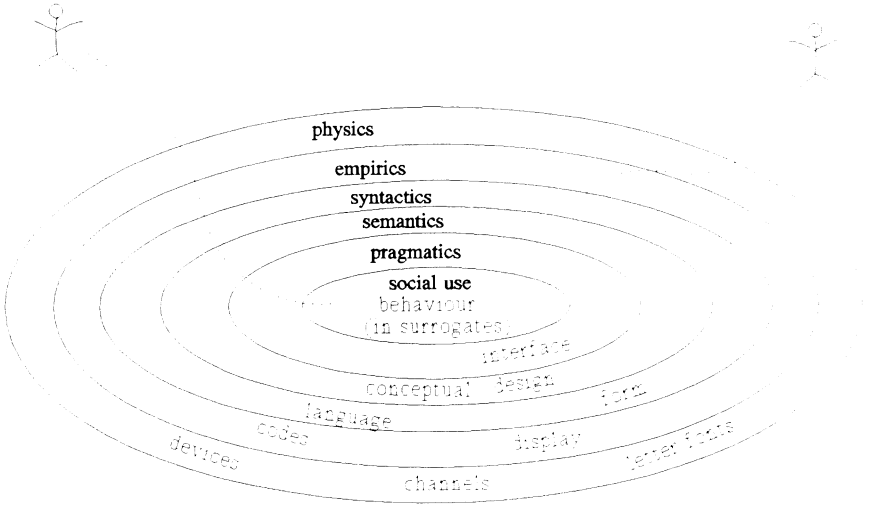



Figure 1: Information Systems Architecture.

Industrial practices suggest that the battle must simultaneously be fought on these two fronts: productivity and quality [29]. Productivity can be improved in many ways including: automation of software engineering processes, and re-use of engineering knowledge and software components. However, industrial experiences have shown that much more work must be done on both two areas to achieve satisfactory results. Much research effort has been put into producing CASE tools or environments for certain methodologies along the line of automation. In the direction of reusability, attention has been directed to the re-use of artifacts and knowledge on development of software and applications in order to reduce the effort of development and maintenance of new software and applications. Artifacts to be re-used include any products of the engineering process, such as systems, components and documentation; while the knowledge includes the facts and experience known by system engineers, such as the information on systems analysis and design [4,15].

A definition that Quality is "fitness for use" [11] suggests that one of the critical steps on the path to a quality product is the analysis of user requirements, followed by representing precisely and realising them exactly in the product. Requirements engineering should occupy a more important place in a whole process of software engineering than it is often the case today. Without a clear definition of requirements there is no standard against which to test quality. In software engineering, attention has been shifted from coding, design more to requirement analysis because the recognised problem as stated by Jalote ([13], p33) that software engineering is weakest in this critical phase.

Another important aspect we must take into account is the possible effect on the economics of software engineering. Requirements engineering has the dominant position in the whole process of software engineering. We are familiar with the escalating costs of rectifying errors which do not surface until a system has been put into operation. The typical software engineering lecturer presents a diagram like the following, giving detail about the design and construction phases but treating requirements definition a an undifferentiated task. In fact requirements definition carried out thoroughly should be assigned several levels, to which an order of magnitude in error rectification costs ought to be ascribed (figure 2).

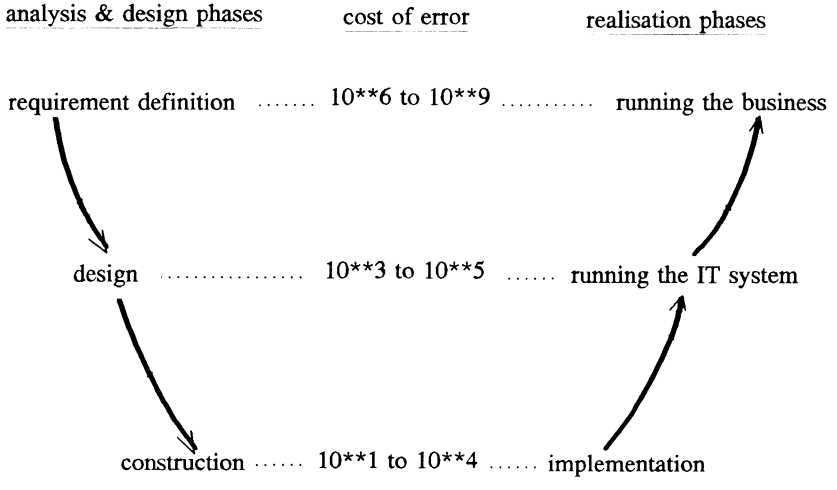



Figure 2: Cost of errors at different levels.

The size of the disaster caused by an error in requirements definition cannot be doubted having witnessed in the UK alone a few typical examples in recent years. One notorious hospital systems cost tens of millions of pounds on being abandoned, the Taurus project at the London Stock exchange cost hundreds of millions while the cost of the abortive Poll Tax system for the UK has been estimated at about 15 billion pounds sterling. A cynic would assert that these mistakes are good for software engineers if no responsibility can be pinned on them. We would assert that neglecting the requirements analysis and specification phase, as though it were purely the responsibility of politicians, administrative policy-makers and management, is inexcusable, from a professional point of view.

To minimise the chance of making those expensive mistakes, a task for our research has been to devise a method of requirements analysis and specification. However, expressing organisational requirements in a form that will meet the needs of software developers calls for a degree of rigour and precision that is otherwise

unnecessary in most other organisational tasks. The vehicle we have been devising provides a formalism for both the agents responsible for business and organisations and the software engineers who will construct the computerised systems to support the business functions.

## A FUNDAMENTAL DIFFERENCE BETWEEN MEASUR AND THE CONVENTIONAL METHODS

The method for requirements analysis and specification, Semantic Analysis, is based on a logic of norms and action developed in the research programme MEASUR.

MEASUR regards all information systems as systems of social norms. When behaviour is organised, it operates according to norms. The most important norms govern behaviour relating to the *substantive* tasks. In a small team we may need to be explicit only about these substantive norms because we can rely on people to keep one another informed and to behave in conformity with the norms that apply to them. In large systems, we have to be explicit about keeping records and passing messages: this generates the *semiological* or bureaucratic norms. We also need some *gubernatorial* norms which say what should be done to ensure conformity with other norms. MEASUR deals with **all** norms but classical methods confine our attention almost exclusively to the semiological norms or the bureaucratic aspects of the organisation. The present day, conventional methods are concerned with data elements and how to structure them for navigation through storage, with dataflows between nodes where data are created, processed, stored or used, and with the functionality of those nodes expressed in terms of transformations to syntactic structures. More recently these ideas have been expressed in terms of data-objects and their functions. All these techniques focus our attention, in fact, upon records, messages and the behaviour of bureaucratic organisation. They do not attempt to go behind the data and enquire what the data signify in the real world. Meaning cannot be ignored but the classical methods assume that it is sufficient to handle them informally, by virtue of our intuition about words. This is clear from the implied ontology of the vocabulary of the specification languages - they limit us to describing a world of data elements and what happens to them.

This point is illustrated by the observation that led to the research on which MEASUR is based. The research began in the early 1970's with the observation that a government department dealing with social security performs a massive information processing task on the basis of a relatively brief set of legal norms. A part of this legislation is then turned into huge number of volumes of clerical codes expressed in the classical 'dataflow' tradition. In the UK Department of Social Security, for example, the ratio is roughly 30-plus slim volumes of legal norms to 400 thick volumes of clerical codes. These clerical codes explain what data should be collected, how they should be stored, the forms which should be used and all the clerical tasks for processing them. Roughly speaking 20% of the legislation gives rise to these routine, clerical activities, the rest will only be applied infrequently by

specialist professional staff and by ministers, senior civil servants and the courts, and it never becomes part of the routine. The whole style of the clerical codes is based on the same concepts as those behind the classical requirements specification techniques. The initial aim of the research was to describe organisations in terms of their social norms in order to achieve the precise and concise form of specification which one finds in legislation.

Human behaviour presents regularities which are governed by norms. Human agents exercise their powers and responsibilities following some rationale within some limits. Their actions exhibit, construct and define the purposes, functions and hence the information requirements of the organisation. In contrast to many approaches to requirements analysis, the Semantic Analysis method treats the use of language as a raw material for analysis but does not stop at the surface. It focuses analysts' attention on the meanings of linguistic expressions by examining the relationships between the use of language and behaviour. In this way the analysis becomes an investigation of the patterns of business behaviour and the specification relates signs to the actions if substantive importance to the business, rather than dealing with actions of a bureaucratic nature.

Requirements specifications are to be completed, as far as business functions and performance are concerned, at three levels. There is, however, another level of specification for technical implementation (e.g. data flows and data formats). The three business requirements specifications are about *norms, communications*, and *semantics*. Analysis of norms aims at identifying the regularities in social behaviour relating responsibilities to certain types of agents (e.g. functional managers) and identifying conditions and constraints upon their actions. The purposes of the study is to understand how an organisation functions socially in relation to fulfilment of its business obligations and commitments within given legal contexts. Communication analysis deals with intentional effects that are achieved by sending messages from one agent to another. Intentions associated with their messages are dependent upon the context of the communication acts on the conversational protocols in which they are used. The most critical aspect, meaning, is the focus of attention of the Semantic Analysis which will be presented later. An objective of semantic analysis is to establish a requirement model in which basic patterns of behaviour are represented and semantics are expressed. The communications and norms are then described on the basis of the semantic information model.

## SEMANTIC ANALYSIS: A METHOD FOR REQUIREMENTS MODELLING

The method of Semantic Analysis is based on some radical philosophical logic and paradigm.

### A Logic of Norms and Affordances, NORMA

An approach to this problem of finding a logic that can deal with semantic

structures has been proposed by Stamper originally in 1985 and most recently discussed in 1993 where more detail can be obtained. In this paper we need only a sketch of the key concepts of NORMA.

The most fundamental assumption required is that, for all practical purposes, we construct our own reality by finding or defining the structures in the flux of behaviour in which we exist. James Gibson [10] introduced the concept of an *affordance* as a result of his work on the psychology of perception. Gibson's theory rejects the idea that an organism sees ready-made objects around it, such as an orange, but instead, the organism detects invariants in the flux of signals reaching its senses. As the observer moves towards, grasps and manipulates the orange, it discovers a cluster of invariants such as a closed visual field of colour, a spherical shape, an odour, and a softness, for example, these experiences and those others offered by hefting, throwing, biting and sucking constitute the orange. The orange may be conceived as a repertoire of behaviour available to the observer rather than as an entirely independent object. Perception depends upon the organism learning to recognise a repertoire of biologically useful invariants of this kind. Thus reality, as we know it, has its origin in the agent.

Human beings recognise not only the invariants that are valuable biologically but the socially valuable invariants. Society evolves patterns of behaviour that enable its members to live together effectively. These naturally evolving norms are the culture. Legal norms can be added to them to refine and extend the cultural structures. A person introduced to a social environment will detect invariants in the behaviour of the established membership of the group and will find her own behaviour subjected to sanctions and rewards that help to reinforce the boundaries of the social invariants. In this we are socialised naturally. Human society has the additional capacity of using a very rich repertoire of signs, including language, through which information reaching the group can be shared. These semiological experiences allow human groups to recognise invariants in a stream of information that would be beyond the ability of an individual to sense. Collectively, therefore, our perceptions are far beyond those of simple organisms. In this way we acquire sophisticated cognitive norms. Similarly, by being able to express norm semiologically, for example by writing legal rules, we can augment our own social reality by constructing more sophisticated invariants that allow us to live together more effectively. Some of these, a copyright for example, are not unlike the orange: they are invariants which correspond to a repertoire of behaviour in the social domain. Copyright may even be bought and sold just like an orange. Norms are invariants that are the social counterpart to the biological affordances that interested Gibson.

Ontological Dependency

An ontology is a philosophical position concerning our basic assumptions about existence. Method engineers are seldom explicit about their philosophical positions but these have far-reaching effects. For example, the classical methods, in their

formal structures, only recognise the existence of signs (messages and records). MEASUR is radically different from them because it adopts an agent-in-action ontology.

This very different philosophical position states that, for all practical purposes, nothing exists without a perceiving agent nor without the agent engaging in actions of some kinds. The words and expression we use names for invariant patterns in the flux of actions and events which agents experience. The classical distinction between entity, attribute and relationship disappears (a development that results in our being able to create re-useable structures) to be replaced by the concepts of affordance (for the behaviour patterns of physical or biological significance) and norms (for the socially defined patterns of behaviour). This accounts for the name of the MEASUR specification formalism, NORMA.

The agent-in-action ontology has other important implications for requirements engineering. In particular, it requires the relevant agent to be specified in every component of the requirements definition. The benefits of this constraint are that (a) we know the responsible originators are users of information, (b) we can handle differential meanings for the same form, (c) we know exactly whom to consult over details of design, (d) we can organise norms (system functionality) according to the various group agents involved. Clearly a philosophical position is not a side issue but of fundamental practical relevance.

Our philosophical arguments lead directly to the observation that an agent in a situation where it experiences certain invariants (affordances or norms) becomes a modified agent:

person upright

where we first name the agent and then the invariant. A modified agent can then behave in some ways that are not possible otherwise. So we may have

person upright walk

person upright jump

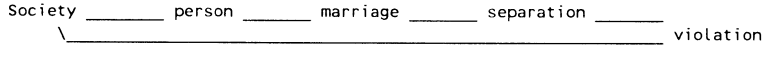
both of which are not affordances of a

person sitting.

Some invariants can only be experienced during the existence of others which we call their 'antecedents'. Similarly, in the social domain

Society (person, person) marriage separation

is a structure in the flux of experience where a separation can only exist while there is a related marriage which, in turn, can only exist during the coexistence of two people and the society which provides the appropriate norms. 'Separation' has many meanings but its meaning is substantially governed in this example by the context provided by the lattice of dependencies. These we can express in what we call an 'ontology chart'. Here, for example, is the chart for separation with another extension to include a violation of the separation by someone:



For our present purposes, the essential idea is that we must know the ontological structure behind a word to know its meaning. If we are not sure when an instance of any one of the above concepts can exist, then we are not sure of its meaning. A brief illustration is in the CONTEST example.

## PROJECT ONE: CONTEST

In this section we address some features of Semantic Analysis in requirements engineering and systems design.

### The project

CONTEST, a project nearing completion, aims at developing a knowledge-based, large scale information system. It stores items (i.e. questions of which a examination can be composed), and produces tests (or, in another word, examinations) according to test specifications given by teachers or educational specialists who want to have a test produced by the system to make discriminations of defined characteristics over specific kind of population. The major system components have been articulated as follows from which the complexity of the system can be seen [6,7]:

1. The databases: There are five databases: item banks, test banks, graphics banks, auxiliary material banks, and population banks. The contents of the five banks are represented in different data types: numerical data, textual data, graphs, etc.
2. The knowledge-base: It contains knowledge needed for eliciting test specification and test generation.
3. The mathematical model-base: The mathematical models are needed for producing test drafts.
4. The general engine and man-machine interface: Through use of these two mechanisms, users can interact with the system for all kinds of applications.

Due to the complexity of the requirements and the foreseeable high cost of the project, the institute felt that it is worth putting great effort in producing a correct requirement specification which can be only guaranteed by involvement of the professionals who verify the requirements.

### Requirements analysis

The requirement model was actually produced interactively by both the analysts and professionals. At the first step, the main task is to obtain an understanding of the total problem to be tackled from various written materials and interviews. The understanding of the problem was then indicated by key terms and concepts which were listed as candidate items to be included in the requirement model. Examples of those terms and concepts are: *institute, school, person, test, student*, etc. All these were then classified into different types. For example, *person, school* and *institute* are agents; *writes* an item and *writes* an test specification are actions; *writers* are role names; *ability scale* and *administration time*, as they describe some aspects of an item, are determiners for item. Relationships of generic-

specific structure are also identified. For example, school is a specific kind of institute. In the requirement model all these elements are linked together according to ontological dependencies. Figure 3 presents a major part of the model. (The major omission from this model is the root agent form which the whole structure originates.)

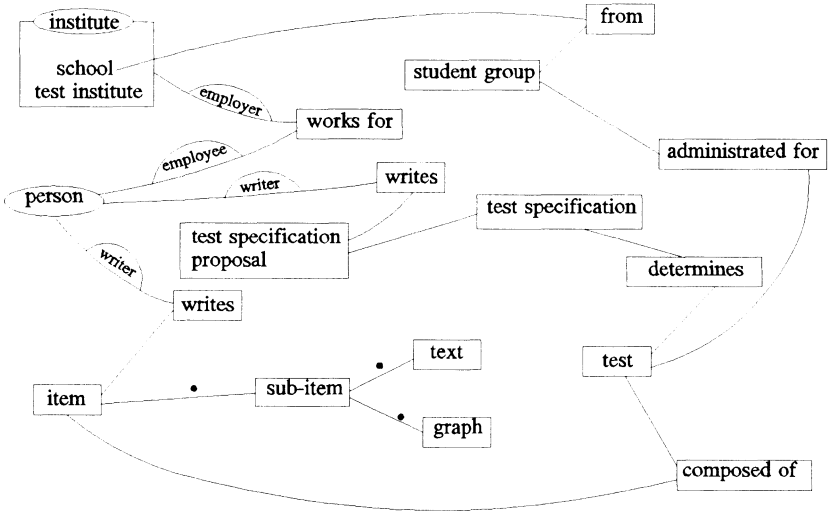



Figure 3: Part of the CONTEST semantic model.

The items enclosed in ovals are agents, e.g., *institute* and *person*. The labels in 'semi-circles' are role-names, such as *writer, employer* and *employee*. *School* and *test institute* are specific kinds of institute, therefore they are placed in the box below the institute. All determiners have been omitted in this example, though they belong to the model. Examples of determiners are: names of particular institute or person; attributes of an item.

Documentation and design

Believe it or not, the total amount of requirement documentation of CONTEST is three A4 pages, which consist of a one page of the semantic model (part of it is shown in figure 3), two pages of determiners of some of the elements of the model and some norms to specify logical and procedural constraints. This was one of the two most striking features experienced by the project team. Before Semantic Analysis was used in the project, the team had conducted requirements analysis using two methods. After that, however, the decision could not be made on whether either or both of the models reflected what was desired by the professionals. One of reasons was the huge amount of documents created an obstacle to the professionals' understanding. The notations in the documents were just syntactic puzzles to the professionals who could not, therefore, even think

about the semantics represented in the model. The documents produced by one of methods amounted to a little more than 30 pages, in contrast to 3 pages produced by the Semantic Analysis.

The other striking feature experienced was the simplicity of the documentation. The elements in the model are the terms and concepts familiar to the users because they are derived from the users' vocabulary. Those concepts are linked by one single relationship, ontological dependency. Once this is taught to a user, he or she will be then able to read and check the model. In fact, after a first draft of the semantic model was made by the analysts, the professional specialists, became an active part of the design team. Very soon they could comment, criticise and suggest improvements to the model. In this way the control of the quality of the requirement modelling has remained in the hands of the responsible professionals.

The CONTEST system has been designed and implemented in object-oriented approach. The conversion from the semantic requirement model to the object-oriented design was relatively simple because of some conversion rules that have been worked out, such as correspondence between agents and objects, behaviour in the model and actions in the design, determiners in terms of the Semantic Analysis and properties in object-oriented terms. Because the requirement model was semantically verified by the users, confidence in the design, therefore and the quality of the whole system has be dramatically enhanced despite some errors incurred during programming.

## PROJECT TWO: NAMAT

NAMAT is an operational software system developed with the use of the Semantic Analysis method. It supports the student information system for a university using the credit-hours academic system. NAMAT 1.0 was deployed in 1987 and was extended several times. The last major extension resulted in NAMAT 4.2 in 1990.

The effect of Semantic Analysis on NAMAT's development and initial user acceptance was reported in [8]. The user management acceptance was and remains particularly encouraging due to the robustness of the temporal student database.

In this section we compare NAMAT and a system, we refer to here as, System-200. System-200 is a popular software package used by over 200 universities. System-200 was built without the aid of semantic analysis to also serve universities with credit-hour academic systems. Both support all academic transactions between the student and the university, altogether about fifty academic transactions. Both systems also record the academic history of the student's previous studies including individual subjects and marks. As well as other personal data. And both have facilities for extraction of ad hoc reports.

Effectiveness in supporting student administration

1. NAMAT's temporal Student Database allowed the extraction of some historical reports that are not possible with System-200.
2. System-200 required one officer to follow up special student cases: *student cases officer* to follow up the loose ends of System-200. No such a role existed in the University using NAMAT, because NAMAT followed up student cases automatically.
3. NAMAT's maintenance system is more responsive to its host university. Every adaptive and corrective maintenance request is dealt with. System-200, on the other hand, could not be changed at all for some requests because that would have caused compatibility problems with the next version of the package.

Cost to support software

NAMAT is supported by just one in-house technical staff member performing adaptive and corrective maintenance, database administration, user training and writing all management reports programs requested.

System-200 on the other hand requires:

- Two in-house technical staff for support
- US$25,000 annual maintenance fee
- Several part time end-users who writing 4GL programs for management reports
- Substantial training investment.

The cost of System-200 training is an order of magnitude more than that required for NAMAT.

Handling Suspension Transactions

Usability: The most immediate is the relative simplicity of NAMAT.

1. **One** NAMAT transaction replaces **two** separate SYSTEM-200 transactions - one to create the prohibition and another to create the anecdotal entry.
2. SYSTEM-200 leaves the user total discretion on whether to make the anecdotal entry and if he does - total arbitrariness on what information to record in this entry.
3. SYSTEM-200 imposes on the user a foreign vocabulary - "HOLD", "FLAG", "SCREEN", "ANECDOTAL ENTRY".
4. SYSTEM-200 provides no information about the student status. NAMAT constructs from student history a concise natural language sentence of the students relevant status and prompts the user with it. This saves the user having to look at many screens.
5. System-200 users must have an understanding not only of what suspension means administratively but also of high degree of interpretation required to map this into System-200 transaction(s).
6. SYSTEM-200 requires the user to memorise SCREEN numbers.
7. The user has to make two System-200 transactions to retract and record a clerical error. Reset the Hold screen flag and another anecdotal entry. NAMAT 4.2

has one transaction to support each retraction and clerical error.

Temporal Functionality

NAMAT is based on a temporal database [22], with non-destructive update technique. This means that users do not loose any information they have previously entered. System-200 support for historical information is inconsistent therefore it is very difficulty to produce historical reports, and monitor student population automatically.

Maintainability

1. A maintenance problem arises when an attempt is made to add another authority for suspension. System-200 requires a technical staff to add another flag that has the effect of prohibiting a student from registration in screen #004 together with the appropriate data and program changes. In NAMAT to add another authority requires no data structure or program changes at all.
2. System-200 is not restrict with the authority. For example when an academic department suspends a student. System-200 is not actually capable of telling us which department. To get this information the user would have to look up the free format anecdotal entry (if the user bothered to put one in) or presume it is the same department that the student is majoring in. It is not even feasible to be more specific about departments because that would require creating flags for each department in the university on the screen! In NAMAT the authorities for all academic transactions are not hard-coded in the screens.

Summary

Whereas it is possible to achieve these same quality characteristics with conventional methods, it is necessary with semantic analysis to embody these characteristics in the requirements specification. NAMAT is a verification of this claim to necessity.

Secondly, Semantic Analysis made it possible to develop and maintain a tailor-made system cheaper than System-200 package solution! The University using System-200 dumped a predecessor tailor-made system and recognises System-200 as a better solution. The few tailor made systems that we have come across so far are too flimsy to consider seriously for comparison. But the invitation is open to make a comparison with another tailor made system.

## NORMBASE: A TOOL TO SUPPORT SYSTEMS DEVELOPMENT

Normbase [28] is primarily developed as an information system shell, and expanded with functions that are required for supporting systems development processes. Therefore it becomes a self-contained, integrated environment for systems development as well as an information system shell for all kinds of applications.

An important aim of the Normbase research has been to evolve automated tools to support the whole process of information systems engineering. Normbase can be used in several stages of a system development: analysis, design and implementation, but it is especially designed to help system developers during analysis and design to concentrate on the core subject matter (or business problems) rather than technological issues. Normbase facilities will allow analysts to conduct elaborated requirements engineering. A strong emphasis has been put in requirement elicitation, articulation and analysis during which knowledge about the business organisation will be obtained. The business knowledge is expressed as the norms of teams, firms, social groups or nation states. All these norms form a conceptual foundation for the design and construction of the system that meets the users' requirements at the policy, organisation strategy and business operation levels.

Design of a database is automatically generated by a Normbase from the Semantic Model. The Semantic Analysis option on a Normbase allows the analyst to establish the conceptual model of the problem. The Semantic Analysis facility guides the analyst to enter specifications of each conceptual entity. With such a facility every entity entered into the system will be added onto the semantic model immediately. In fact the modelling is an interactive process with users. Because the semantic modelling employs users' vocabulary rather than creating artificial labels as most of the analysis methods would do, the users can understand the model easily with a modest amount of assist from the analyst. The concision of the representation also encourages the users' attempt to understand what has been written in the documents for quality control purposes (very often the key requirements documents in the form of semantic models for a project are no more than two or three pages). Once the analysis is approved, the semantic model will serve as the conceptual design of a Semantic Temporal Database (STDB).

An STDB is generated from the semantic mode. Adding a rich semantic structure to the data has marked advantages: meaningless structures or data are eliminated; temporal constraints are applied automatically and complex temporal queries are simple. To illustrate this past point, for example, a simple query about which guests at a hotel are attending various functions would be expressed as

*guest(person)* <u>*while*</u> *attending(person, function)*

in the form "a <u>while</u> b" requires 2 operands and 1 operator, 3 expressions, instead of in the 99 expressions needed in SQL! Norms are constructed using formulas like the one above and they are associated with the *start* and *finish* of the existence of each element in the semantic model. The system, therefore, also guide the analyst when eliciting norms from the user community. An interpreter applies the norms to the data and norms can be held as data within the STDB. Data, semantic model and norms (data, schema and functions) are never overwritten in the STDB, their existence attribute has its finish time added but the history is persevered. These features of the STDB all contribute to system quality.

## CONCLUSIONS: HOW SEMANTIC ANALYSIS INCREASES SIMPLICITY, UNIFORMITY AND QUALITY

A fundamental change in the philosophy underlying the concept of an information system has far-reaching effects on quality of requirements specified using MEASUR. In summary they are as follows

- E-R-A distinction is removed, the semantic schema is simpler as a result;
- all concepts are patterns of behaviour, so that the schema incorporates time;
- Strict semantic principles ensure that the schema is stable and its parts are re-usable;
- the agent-in-action ontology allows a rich structure of agent, times, authorities etc to be implicit in the schema thus reducing documentation by a factor of about 10;
- rigorous specification plus simple, understandable documentation puts the user back in control over the definition of requirements;
- requirements analysis entirely focused on the business or substantive problems without technical complexities also improves quality;
- uniform, nor-destructive STDB for data, schema and norms vastly reduces complexity of implementation.

MEASUR is still rather new but we and many other collaborators have now tested it sufficiently in practice for us to feel confident that it greatly enhance system quality.

## NOTE

The MEASUR methods are commercially available through Semiotica at:
Hengelosestraat 705, Postbus 545, 7500 AM Enschede, The Netherlands.
Telephone +31.53.836580, fax +31.53.337415.

## REFERENCES

1. Andersen, Peter B. (1990) *A Theory of Computer Semiotics: semiotic approaches to construction and assessment of computer systems.* Cambridge University Press, Cambridge.
2. Andersen, Peter B. (1991) A Semiotic Approach to Construction and Assessment of Computer Systems, in Nissen, *et al.*
3. Andersen, Peter B. and Berit Holmqrist (eds) (1994) *Semiotics in the Workplace*, De Gruyter.
4. Biggerstaff, T.J. & Richter, C. (1987) Reusability Framework, Assessment, and Directions, *IEEE Software*, 4(2) March pp.41-49.
5. Boehm, B. (1981) *Software Engineering Economics.* Prentice-Hall, Englewood Cliffs, New York.
6. Boekkooi-Timminga, E. (1989) *Models for Computerised Test Construction,* Academisch Boeken Centrum, De Lier.
7. Boekkooi-Timminga, E. and Sun, L. (1991) Contest: A Computerised Test

Construction System, in Hoogstraten, *et al.*

8. Badr, E., Y. Ades, A. Eid (1992) The Experience of Qatar University with NAMAT, 13th Annual Conference of ACRAO, University of Qatar, Doha.
9. Dix, A, J. Finlay, G. Abowd and R. Beale (1993) *Human-Computer Interaction*, Prentice Hall, NY.
10. Gibson, J.J. (1979) *The Ecological Approach to Visual Perception*, Houghton Mifflin Company, Boston.
11. Hall, Terence J. (1992) *The Quality Manual*. John Wiley, Chichester.
12. *Hoogstraten, J. and Linden, W.J. van, (eds) (1991) Proceedings of Dutch National Conference on Educational Research*, Stichting Centrum voor Onderwijsonder-zoek van de Universiteit van Amsterdam.
13. Jalote, P. (1991) *An Integrated Approach to Software Engineering*. Springer-Verlag, New York.
14. Jirotka, M., J. Goguen and M. Bicherton (eds) (1994) *Social Technical Aspects of Requirements Engineering*, Academic Press, New York.
15. Li, Haikuan (1993) *RITL: An Information System for Application Reuse-in-the-Large*. Delft University Press, Delft.
16. Liu, K. (1993) *Semiotics Applied to Information Systems Development*. Doctoral Thesis, University of Twente, Enschede.
17. Morris, C.W. (1938) Foundations of the Theory of Signs, *International Encyclopedia of Unified Science*, 1,2. Univ. of Chicago Press, Chicago.
18. Morris, C.W. (1946) *Signs, Language and Behaviour*. Braziller, New York.
19. Navath, S. (ed) (1985) *Proceedings of ACM-SIGMOD's International Conference on Management of Data*, Austin, TX, SIGMON Record 14 (4).
20. *Nissen, H.-E., Klein, H.K. & Hirschheim, R. (eds) (1991) Information Systems Research: Contemporary Approaches and Emergent Traditions*, Elsevier Science Publishers, North-Holland.
21. Peirce, Ch.S., (1931/35) *Collected Papers of Ch.S. Peirce*, edited by Hartshorne, C. & Weiss, P. Cambridge, Mass. 1960.
22. Snodrgass, R. (1985) A Taxonomy of Time in Databases, in Navath.
23. Sommerville, Ian (1992) *Software Engineering (fourth edition)*. Addison-Wesley Publishing Company, Workingham, England.
24. Stamper, Ronald K. (1991) The Semiotic Framework for Information Systems Research, in Nissen, *et al.*
25. Stamper, Ronald K. (1992) Signs, Organisations, Norms and Information Systems. *Keynote paper for the 3rd Australian National Systems Conference*, Wollongang.
26. Stamper, Ronald K. (1992) Signs, Norms and Information Systems. ICL/University of Newcastle Seminar on "Information".
27. Stamper, Ronald K. (1994) Social Norms in Requirements Analysis, in *Jirotika, et al.*
28. Stamper, R.K., K. Liu, M. Kolkman, P. Klarenberg, F. van Slooten, Y. Ades, and C. van Slooten (1991) From Database to Normbase. *International Journal of Information Management*, 11 62-79.

29. Troy, Robert (1991) Impact of Methods on Productivity and Quality. *ESEC'91 Proceedings*, Lamsweerde, A.van & Fugetta, A. (eds), Springer-Verlag, Berlin.
30. Weigand, H. (1990) *Linguistically Motivated Principles of Knowledge Base Systems*. Foris Publications, Dordrecht.
31. Winograd, T. & Flores, C.F. (1986) *Understanding Computers and Cognition*. Addison-Wesley Publishing Company.