1991

AN ONTOLOGICAL EVALUATION OF NIAM'S GRAMMAR FOR CONCEPTUAL SCHEMA DIAGRAMS

Ron Weber
The University of Queensland

Yanchun Zhang
The University of Queensland

AN ONTOLOGICAL EVALUATION OF NIAM'S GRAMMAR FOR CONCEPTUAL SCHEMA DIAGRAMS

Ron Weber
Department of Commerce
The University of Queensland

Yanchun Zhang
Department of Computer Science
The University of Queensland

ABSTRACT

This paper describes an evaluation of the grammar employed in Nijssen's Information Analysis Method (NIAM) to generate conceptual schema diagrams. The evaluation is undertaken using the ontological model proposed by Bunge and Wand and Weber. It shows that the grammar used to generate NIAM conceptual schema diagrams is primarily deficient in two respects. First, semantic overload occurs in NIAM. Multiple ontological constructs are represented by a single grammatical construct in NIAM. Second, NIAM has semantic deficit. Some ontological constructs have no corresponding grammatical constructs in NIAM. On the basis of the evaluation undertaken, some propositions are made about how NIAM will be perceived and used in practice.

1. INTRODUCTION

An important part of many information system development methodologies is the grammar(s) they use to generate scripts describing some aspects of the real world. These grammars may be formal (LOTOS), semi-formal (entity-relationship modeling), or informal (unrestricted English). Similarly, the scripts they produce may be formal (predicate logic statements), semi-formal (entity-relationship diagrams), or informal (English sentences).

Many such representational grammars now exist (Olle et al. 1988). Attempts have been made to better understand their nature and their relative strengths and weaknesses in generating scripts that form the basis for information systems analysis, design, and implementation. For example, they have been evaluated via feature analyses, case studies, laboratory experiments, and field studies (e.g., Floyd 1986; Shoval and Even-Chaime 1987). In the absence of better theory, however, researchers now seem to agree that further empirical research to evaluate these grammars is problematical (e.g., Bubenko 1986, Lindgreen 1990).

In an attempt to provide an improved theoretical basis for evaluating information systems representational grammars, Wand and Weber (1989a, 1990a) have adapted and extended an ontological formalism developed by Bunge (1977, 1979). Their formalism focuses on three aspects of a grammar:

1. Does the grammar contain constructs that allow it to generate scripts which represent all pertinent aspects of the real world to be modeled?
2. Does the grammar contain constructs that allow it to generate scripts which enable information systems designed and implemented on the basis of the scripts to faithfully track state changes in the real-world systems they are intended to model?
3. Does the grammar contain constructs that allow it to generate scripts which facilitate "good" decompositions during information systems analysis and design?

To evaluate the merits of their model, Wand and Weber (1989b, 1990b) have already undertaken an evaluation of the entity-relationship model (ERM) and data flow diagrams (DFDs). Their initial results are tentative but encouraging. Their model permitted a systematic, theoretically-based evaluation of the ERM and DFD grammars. Nevertheless, more work still needs to be done on their model in a number of areas. For example, the pragmatics of applying the model to evaluate grammars are not straightforward. Moreover, the implications of some types of grammatical strengths and limitations identified via the model are still unclear.

To extend the work of Wand and Weber, in this paper we employ their model to evaluate the grammar used by Nijssen's Information Analysis Method (NIAM) to generate conceptual schema diagrams (CSDs). NIAM is currently employed by many organizations in both Australia and Europe to undertake conceptual schema and relational database design. Its proponents argue it has excellent semantic modeling capabilities and that it leads to elegant relational database designs (e.g., Zhang and Orlowska 1990a, 1990b, 1991). Thus an evaluation of NIAM should be useful to both its existing and prospective

users. In our view, also, NIAM's grammar for conceptual schema diagrams (GCSD) provides a richer basis for testing the Wand and Weber model than either the ERM or DFD grammars. In this light, we believe our paper provides an additional, important test of the analytical powers of their model.

The paper proceeds as follows. Section 2 provides a brief recapitulation of the Bunge-Wand-Weber model (henceforth the BWW model). Section 3 describes our use of the model to analyze the ontological foundations of NIAM's GCSD. Section 4 summarizes some ontological deficiencies in NIAM's GCSD. Section 5 provides propositions about the consequences of these deficiencies and presents some conclusions.

2. OVERVIEW OF THE BUNGE-WAND-WEBER MODEL

Wand and Weber (1990b) argue their model has two major purposes. First, it can be used to better understand the characteristics of good information systems. "Good" is defined in a special and restricted way to indicate whether the "deep structure" of an information system manifests the *meaning* of the real-world system it is intended to represent. Second, it can be used to understand and predict the characteristics of good information systems grammars. Again, "good" refers to how well the scripts generated using the grammar manifest the meaning of the real-world system to be represented.

In this paper, we focus on the second purpose of the BWW model. In terms of this purpose, three elements of the model are important. First, the model prescribes a set of ontological constructs that supposedly are necessary and sufficient to describe the structure and behavior of real-world things (see Table 1 in Wand and Weber 1990b). Using these constructs, information system grammars can be evaluated to determine whether they are *ontologically complete*. A one-to-one mapping should exist between each ontological construct and each grammatical construct. Otherwise, Wand and Weber predict the following problems might arise.

1. *Semantic overload:* More than one ontological construct may map into a single grammatical construct.

2. *Syntactic overload:* An ontological construct may map into more than one grammatical construct.

3. *Semantic deficit:* An ontological construct may have no corresponding grammatical construct.

4. *Syntactic excess:* A grammatical construct may have no corresponding ontological construct.

The second element specifies four requirements that information systems must satisfy if they are to faithfully track the real-world systems they are intended to model.

1. *Mapping requirement:* Each real-world system state must be represented by at least one information-system state.

2. *Tracking requirement:* Changes of state in the information system must correspond to changes of state in the real-world system.

3. *Reporting requirement:* External (input) events that occur in the real-world system must be reported to the information system.

4. *Sequencing requirement:* The order in which external (input) events occur in the real-world system must be the same as the order in which external (input) events occur in the information system.

Wand and Weber argue that information systems grammars must contain constructs that allow these requirements to be satisfied in the scripts they generate. Otherwise, the grammar will not produce "good" scripts.

The third element specifies the set of conditions that must be satisfied if "good" decompositions are to be undertaken based on scripts generated using a grammar. A central notion in the model is the concept of a well-defined event. All events can be conceived as a change from one state to another state. Given a system is in one particular state, events are *well defined* if the subsequent state can always be predicted. According to Wand and Weber, the following condition is necessary for a decomposition of an information system to be good: *For a given set of external (input) events at the system level, all induced events in every subsystem (which includes the system) must be either specified external events or well-defined internal events.* In other words, only external events in a subsystem can be poorly defined in the sense that the subsequent system state cannot be predicted given its current state.

Again, Wand and Weber argue that information system grammars can be evaluated in terms of whether they generate scripts where, for a given set of external events, all events in all subsystems are either specified external events or well-defined internal events. If the grammar does not lead users to produce scripts satisfying this condition, good decompositions will not be forthcoming.

3. ONTOLOGICAL ANALYSIS OF NIAM'S GCSD

In the following subsections, we use the BWW model to analyze NIAM's GCSD.[1] We evaluate the grammar as reported in Nijssen and Halpin (1989) (henceforth NH). To illustrate GCSD constructs, Figure 1 shows a NIAM diagram (script) produced using the grammar for a particular domain of discourse. The diagram does not illustrate all GCSD constructs. For brevity, we have omitted some features. We believe these omissions have not biassed our evaluation.

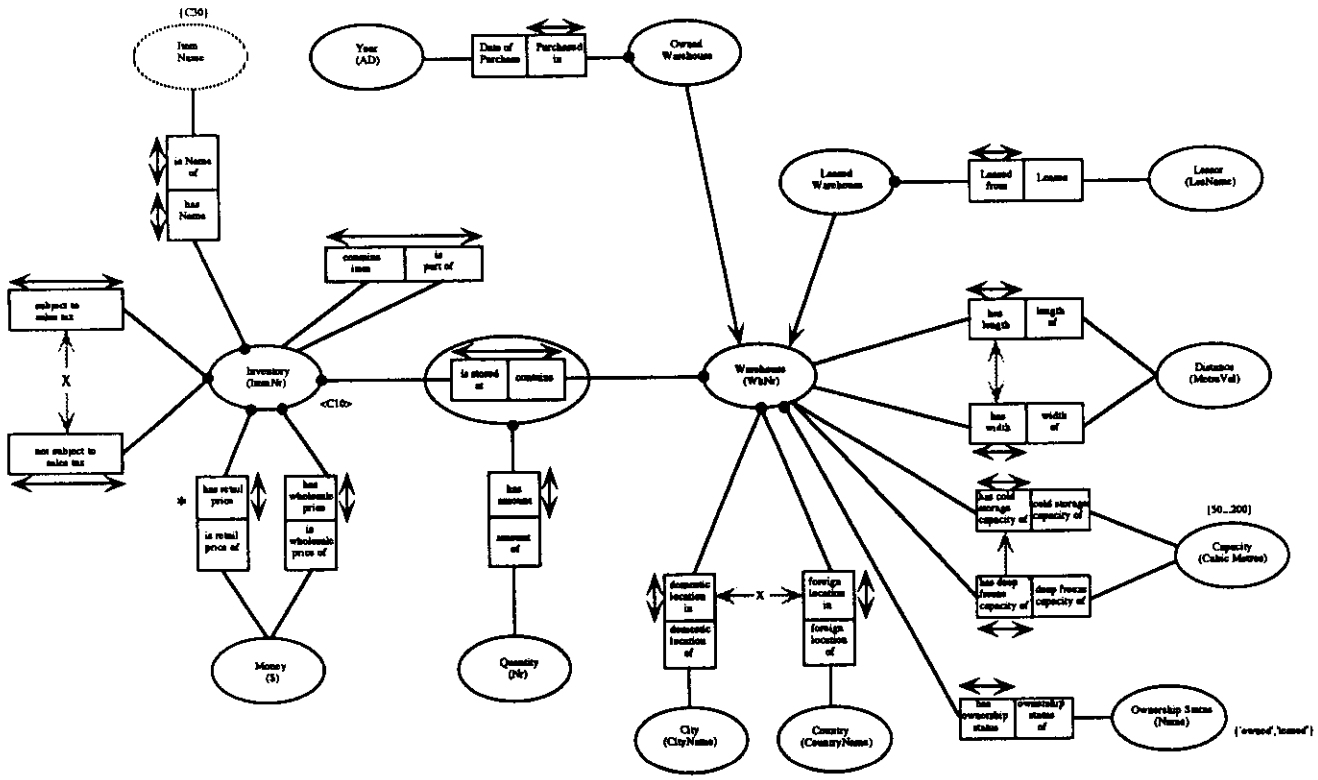

Figure 1. NIAM Conceptual Schema Diagram for an Inventory/ Warehousing Domain of Discourse

3.1 Fact

The most fundamental construct in NIAM is a fact. NH (p. 36) define an elementary fact as an assertion "that a particular object has a property, or that one or more particular objects participate together in a relationship." An elementary fact cannot be broken up into sub-facts without loss of information. Compound facts are composed of elementary facts.

Facts in the BWW model have a slightly broader meaning. Bunge (1977, p. 169) lists the following "definite descriptions and propositions" as "factual items":

Thing *b*	Thing *b* exists.
Property *P*	Thing *b* possesses property *P*.
Kind *K*	Thing *b* is of kind *K*.
State *s*	Thing *b* is in state *s*.
Event *e*	Thing *b* undergoes change *e*.

All the above statements can be cast in the form: a particular thing has a particular property. The property might be existence, membership of a kind, etc. Thus, NIAM facts closely corresponded to facts in the BWW model.

3.2 Entities

NH (p. 37) define entities as "basic objects or things that we want to talk about....An entity may be a tangible object (e.g., the person Felix) or an abstract object (e.g., the subject CS112)." An entity *type* (a set of entities of the same kind) is shown as a solid ellipse in a NIAM diagram. In Figure 1, inventory is a NIAM entity type.

At first glance, a one-to-one mapping appears to exist between NIAM entities (instances of an entity type) and things in the BWW model. However, NIAM entity types are also used to represent the *codomains* of the state functions that represent properties in the BWW model. In Figure 1, the capacity entity type shows a warehouse's cold-storage and deep-freeze capacities in cubic meters. In the context of the BWW model, the cubic-meter, cold-storage and deep-freeze capacities are properties of a warehouse thing. State functions that represent these properties map the warehouse to positive integers. In Figure 1, the entity type called "capacity" has a one-to-one mapping to a set of integers that specify its measure in cubic meters.

In summary, NIAM entities may correspond to a thing or an element in the codomain of a property in the BWW

model. As a result, semantic overload exists because more than one ontological construct maps into a single grammatical construct. In this light, the BWW model predicts that NIAM users may be confused because they have to employ knowledge not embedded in the diagram to interpret the meaning of a construct.

3.3 Role

NH (p. 42) define a role as "a part played by an entity in some relationship." The relationship may be unary. In Figure 1, an inventory item is or is not subject to sales tax. Alternatively, the relationship may be n-ary (n > 1). In Figure 1, the is-stored-at role for an inventory item participates in a binary relationship between inventory item and warehouse. In both cases, roles are represented as a box on a NIAM diagram. Roles in an n-ary relationship (n > 1) are represented by concatenated boxes (see the is-stored-at and contains role boxes in Figure 1).

In the context of the BWW model, NIAM's role construct constitutes a property of a thing. A role may participate in a unary relationship (e.g., the subject-to-sales-tax role in Figure 1). A role may link a type of thing (represented by a solid ellipse in NIAM) with a codomain (represented by either a solid ellipse or dotted ellipse in NIAM; e.g., the has-retail-price and has-name roles in Figure 1). A role may participate in an n-ary relationship (n > 1) where the role "links" two or more types of things (e.g., the is-stored-at role links an inventory type of thing with a warehouse type of thing). In each case, the role is a property of some thing.

Roles also may be contained in an ellipse. In Figure 1, an ellipse has been drawn around the is-stored-at role and the contains role. The ellipse stands for a complex entity type, which is equivalent to a complex type of thing in the BWW model. In this example, the complex thing is conceived as a particular inventory item stored at a particular warehouse. The is-stored-at and contains roles are properties of the complex thing as well as properties of the simple inventory and warehouse things. Note, the has-amount role is also a property of the complex thing.

3.4 Label Types/Reference Modes

A *label*, by itself or in conjunction with another label, may identify a particular object. Labels are instances of a label type, which is represented in a NIAM diagram via a broken ellipse. In Figure 1, item name is a label type for the inventory entity.

Without further information, sometimes the meaning of a label type value is unclear. For example, some inventory items may have two names: the name given by their supplier and the name given by their wholesaler. In Figure 1, the item-name label type does not indicate which name is being used to name the inventory item. In NIAM, this ambiguity can be resolved using a *reference mode*. Reference modes are types of roles that spell out more completely the nature of the label type. In Figure 1 the is-name-of role might be changed to an is-supplier-name-of role to make clear that item name refers to the supplier's name.

In the context of the BWW model, a label type is the name of a codomain that may be used by one or more property functions, and a reference mode is a property of a thing. Neither the reference mode nor label type constructs are needed if the relevant (codomain) entity type name and role name fully specify the nature of the property of a thing. In this light, the BWW model predicts NIAM users may be confused because syntactic overload exists. Two grammatical constructs, roles and reference modes, can be used to represent the property ontological construct. Moreover, two grammatical constructs, entity types and label types, can be used to represent property function codomains. Note, however, that a reference mode is a *type* of role and a label type is a *type* of object (entity) type. Users perhaps benefit from a finer classification of properties (roles) and property codomains (entity types). This possibility must be tested empirically.

3.5 Derived Facts

In NIAM, derived facts are facts that are a function of one or more other facts (NH, pp. 58-59). On a NIAM diagram, the role associated with an entity type in a derived fact has an asterisk placed beside its box. The derivation rule is specified below the diagram. In Figure 1, the asterisk beside the has-retail-price role box indicates an inventory item's retail price depends upon its wholesale price (e.g., a constant markup applies).

In the context of the BWW ontological model, derived facts assert a thing possesses an *emergent* property – a property whose values depend upon the values of other properties. Consistent with the BWW model, therefore, NIAM distinguishes emergent properties (manifested in derived fact types) from hereditary or resultant properties (manifested in stored fact types).

3.6 Nested Fact Types

Sometimes complex things are represented explicitly in NIAM via nested fact type constructs (NH, pp. 54-55). A nested fact type is depicted as an ellipse around two or more concatenated role boxes. In Figure 1, two roles pertaining to inventory items and warehouses are nested.

In the context of the BWW model, nested fact types are used when complex things have an emergent property of interest. In Figure 1, quantity on hand is an emergent property of the complex thing comprising an inventory item-warehouse pair.

3.7 Uniqueness Constraints

NIAM uniqueness constraints show that the values of roles in a relationship are constrained. Specifically, they indicate which *instances* of a role or a concatenation of roles must be unique. They reflect 1-to-1, 1-to-N, and M-to-N relationships that are well known within database theory.[2]

The double-headed arrow in Figure 1 over the leased-from role indicates only one instance of a particular warehouse number will appear in the set of tuples (warehouse number, lessor name) manifesting the relationship between leased warehouses and the names of their lessors. Thus, this uniqueness constraint reflects a 1-to-N relationship from lessors to warehouses. Similarly, the double-headed arrow spanning both the stored-at and contains roles indicates each instance of an item-number, warehouse-number pair must be unique. Thus, this uniqueness constraint manifests an M-to-N relationship between inventory items and warehouses.

In the context of the BWW model, the nature of uniqueness constraints is straightforward: they are a type of state law – laws that restrict the possible state space of a thing to its lawful state space.

3.8 Entity Type Constraints

In NIAM, entity type constraints have two purposes. First, they specify the set of values that an entity type can be assigned. Second, they specify abstract data types, which NH (p. 160) define as "basically a set of values with a well defined set of operations."

Entity type constraints are represented on a NIAM diagram by annotations showing set extensions or character strings beside the entity types to which they apply. In Figure 1, the entity type constraint beside the capacity entity type shows the cold-storage or deep-freeze capacities of warehouses is a minimum of 50 cubic meters and a maximum of 200 cubic meters. In addition, the square brackets indicate that numbers within the set are subject to the usual arithmetic operations.[3]

In the context of the BWW model, entity type constraints have two meanings. First, they can be a type of state law. They specify allowed values that a property of a thing may have, given that the property is considered *in isolation* from other properties of the thing. For example, if a warehouse's storage capacity depends upon the warehouse's location, an entity type constraint would *not* show this inter-property constraint. Second, they can specify lawful transformations. If entity type constraints represent abstract data types, the operations defined via the abstract data type constitute lawful transformations in the BWW model. Thus, a situation of semantic overload exists: one grammatical construct represents two ontological constructs. Note, however, the syntax of entity type constraints appears sufficient to distinguish between the state-law and lawful-transformation constructs embedded in them. In the above example, the operations (lawful transformations) are implied by brackets.

Both uniqueness (discussed in 3.7 above) and entity type constraints are state laws in the BWW model. At first glance, therefore, two grammatical constructs appear applicable to a single ontological construct, and syntactic overload seems to exist. However, each grammatical construct represents a different *type* of state law. Thus, the grammatical and ontological constructs are not equivalent. The BWW model predicts syntactic overload occurs only when multiple grammatical constructs are *equivalent* to a single ontological construct.

3.9 Mandatory/Optional Roles

NH (p. 114) define mandatory and optional roles as follows: "A role is **mandatory** if and only if, for all states of the database, it must be recorded for every member of the population of the attached entity type; otherwise the role is **optional**." Mandatory roles are designated by a dot attached to the entity type ellipse where the ellipse is intersected by the arc from the role box that represents the mandatory role. In Figure 1, the has-name role is mandatory; each inventory item must have an item name. However, the domestic-location-in role is optional; not all warehouses are sited at domestic locations. Nonetheless, note that the disjunction of the domestic-location-in and foreign-location-in roles is mandatory.

In the context of the BWW model, mandatory and optional roles indicate whether a thing has a particular property. A mandatory role indicates a thing within a particular class or kind always has the property designated by the role. An optional role indicates a thing within a particular class or kind may or may not have the property. Mandatory and optional roles partially define the state space of things represented in NIAM.

3.10 Subtypes

NIAM subtypes are entity types that have all roles possessed by their supertype and at least one role possessed only by themselves. Thus, subtypes show the classes and "is-a" relationships that have been extensively described within the database management literature (e.g., Korson and McGregor 1990). Subtypes are shown in NIAM via directed arcs from the subtype to their supertype. In Figure 1, owned warehouses and leased warehouses are subtypes of warehouses.

Subtypes are *kinds* within the BWW model because things in subtypes must be defined via at least two properties: at least one property from their supertype and at least one property that is a property only of the subtype. In this

light, *classes* in the BWW model (set of things that can be defined via one property) must be represented by entity types in NIAM that play only one role. If the entity type plays more than one role, the things represented by the entity type will form a kind.

Thus, even though an entity type is used to represent both classes and kinds, semantic overload is avoided because the number of roles attached to the entity type determines whether it constitutes a class or a kind. Moreover, property relationships among different classes and kinds are shown in NIAM via directed arcs.

3.11 Equality Constraints

In NIAM, equality constraints indicate a value for one role (or sequence of roles) of an entity type is recorded if and only if the value of another role (or sequence of roles) of the entity type is recorded. In Figure 1, the length and width of a warehouse are optional roles; they may or may not be recorded. The dotted, double-headed arrow between the "has length" and "has width" roles, however, indicates the length of a warehouse is recorded if and only if the width of a warehouse is recorded.

In terms of the BWW model, equality constraints are a type of state law. Equality constraints indicate one property of a thing has a non-null value if and only if another property of the thing has a non-null value.

3.12 Exclusion Constraints

NIAM exclusion constraints indicate an entity can play only one of two or more optional roles. In Figure 1, the domestic-location-in role and foreign-location-in role are optional. However, the dotted line with an "X" between the role boxes indicates a warehouse can play only one of these roles at any time. Thus, if warehouses have a domestic location, they cannot also have a foreign location.

In terms of the BWW model, exclusion constraints mean things have either one property or another property. Unlike most other NIAM constraints, exclusion constraints are not state laws in the BWW model. If a thing in the BWW model never has a particular property throughout its life, the property is not included in the functional schema used to describe the thing.

3.13 Subset Constraints

NIAM subset constraints indicate instances of a role (or sequence of roles) are a subset of the instances of another role (or sequence of roles). In Figure 1, consider the two roles relating to the cold-storage and deep-freeze capacity of warehouses. Note, warehouses may or may not have cold-storage or deep-freeze facilities (the roles are optional). Furthermore, assume warehouses have deep-freeze facilities only if they have cold-storage facilities. However, warehouses might not have deep-freeze facilities even though they have cold-storage facilities. Thus, the role instances showing which warehouses have deep-freeze facilities will be a subset of the role instances showing which warehouses have cold-storage facilities. Diagrammatically, the subset constraint is shown by a dotted arrow from the role whose instances are the subset to the role whose instances are the superset.

In terms of the BWW model, subset constraints mean things may have a particular property only if they already possesses another property. If things never have the former property throughout their life, the property is never modeled via the thing's functional schema. If, however, things may have the former property at various times in their life, the property will sometimes have a null value and sometimes have a non-null value. Whenever the former property has a non-null value, however, a state law requires that the latter property has a non-null value (the essence of the subset constraint). In short, subset constraints are a type of state law defined over two or more properties of a thing.

4. ONTOLOGICAL DEFICIENCIES IN NIAM'S CONCEPTUAL SCHEMA DESIGN GRAMMAR

Table 1 shows a summary comparison of constructs in NIAM's GCSD and constructs in the BWW model. As a basis for representing and designing the deep structure of information systems, the table indicates NIAM's grammar has both strengths and weaknesses. On the one hand, it provides a rich array of constructs that allow many aspects of real-world states and state laws to be represented. These constructs are more powerful than those provided by the ERM or DFDs (Wand and Weber 1989b). On the other hand, it suffers from semantic overload and semantic deficit. The following two subsections summarize the nature and implications of these deficiencies.

4.1 Semantic Overload

Our analysis in section 3 indicates semantic overload in NIAM occurs only in one area: the NIAM entity construct is used to represent both things and elements in the codomain of a property in the BWW model.

Unfortunately, the BWW model does not indicate the predicted consequences of this type of semantic overload. Indeed, NIAM advocates would argue *advantages* accrue by having entity types represent both types of things and codomains of property functions. In brief, the overload allows users to avoid having to distinguish between entities (things) and attributes (properties), which is a well-known dilemma in the ERM (NH, pp. 313-314). Thus, while the BWW model is vague in its predictions about this aspect of NIAM's GCSD, it nevertheless highlights an important and contentious issue that warrants further theoretical and empirical research.

Table 1. Evaluation of NIAM'S GCSD for Ontological Completeness

OntologicalConstruct	Representation in NIAM's CGSD
Thing	Things via entities. Composite entities in nested fact types
Property	Roles, reference modes. Emergents sometimes via derived fact types.
Property Codomain	Entity types, label types.
State	May not be fully represented.
Conceivable State Space	May not be fully represented.
State Law	May not be fully represented.
Lawful State Space	May not be fully represented.
Event	May not be fully represented.
Conceivable Event Space	May not be fully represented.
Transformation	May not be fully represented. Some implied in entity type constraints.
Lawful Transformation	May not be fully represented. Some implied in entity type constraints.
Lawful Event Space	May not be fully represented.
History	None
Coupling	May not be fully represented.
System	May not be fully represented.
System Composition	May not be fully represented.
System Environment	May not be fully represented.
System Structure	May not be fully represented.
Subsystem	May not be fully represented.
System Decomposition	None.
Level Structure	None.
External/Internal Event	None.
Stable/Unstable State	None.
Well-Defined Event	None.
Poorly-Defined Event	None.
Class	Via entity types that have only one role.
Kind	Via entity types that have more than one role and via subtypes.

4.2 Semantic Deficit

NIAM's semantic deficit problems can be grouped into three types: deficiencies in the representation of states; deficiencies in the representation of dynamics; and deficiencies in the representation of systems and their associated constructs.

State-representation deficiencies exist because NIAM's GCSD cannot fully represent conceivable state spaces, state laws, lawful state spaces, and stable and unstable states. On the basis of the BWW model, we predict these deficiencies lead to three types of problems with NIAM scripts: (a) the mapping requirement in the BWW model will be violated; (b) the scripts will not form the basis for controls design; and (c) the reporting and sequencing requirements in the BWW model will be violated.

Dynamics-representation deficiencies exist because NIAM's GCSD cannot fully represent events, conceivable event spaces, lawful event spaces, transformations, lawful transformations, and external and internal events. On the basis of the BWW model, we predict these deficiencies lead to three types of problems with NIAM scripts: (a) designers may fail to identify important events and transformations in the real-world system; (b) the tracking, reporting, and sequencing requirements in the BWW model may not be satisfied; and (c) the scripts will not form the basis for controls design.

System-representation deficiencies exist because NIAM's GCSD cannot fully represent couplings between things and, consequently, subsystems, level structures, and the composition, environment, and structure of systems. On the basis of the BWW model, we predict these deficiencies undermine NIAM's usefulness for undertaking decompositions during analysis and design.

5. SOME PROPOSITIONS AND CONCLUSIONS

In light of our analysis, we make the following propositions about NIAM's GCSD.

1. Because of the semantic overload surrounding things and property codomains, users will lack conceptual clarity about the nature of NIAM entities.

2. Because NIAM provides a better representation of real-world statics relative to real-world dynamics, users will tend to employ NIAM to analyze "data-oriented" rather than "processing-oriented" systems.

3. Because NIAM does not fully represent systems, subsystems, stable states, and well-defined events, users are unlikely to employ it to undertake decomposition during analysis and design.

4. Because NIAM does not fully represent state laws and lawful transformations, users are unlikely to employ it to design controls.

5. NIAM will tend to be used with other methodologies that compensate for its semantic deficiencies.

To conclude, we have found the BWW model useful in generating predictions about strengths and weaknesses in NIAM's GCSD. In this regard, we have a candidate theory about the characteristics that information systems grammars must possess if they are to generate scripts providing good representations of the meaning embedded in real-world systems. The quality of these predictions is now an empirical issue.

6. ACKNOWLEDGEMENTS

We are indebted to Maria Orlowska and the reviewers for comments on an earlier version of this paper. We are especially grateful to Terry Halpin for detailed comments and helpful discussions. Naturally they are absolved from the obligation of having to agree with our arguments or responsibility for any errors or omissions. The research described in this paper was supported in part by grants from the Australian Research Council and GWA Pty. Ltd.

7. REFERENCES

Bubenko, J. A. Jr. "Information System Methodologies – A Research Review." In T. W. Olle, H. G. Sol, and A. A. Verrijn-Stuart (Eds.), *Information Systems Design Methodologies: Improving the Practice*, Amsterdam: North-Holland, 1986, pp. 289-318.

Bunge, M. *Treatise on Basic Philosophy: Volume 3: Ontology I: The Furniture of the World*. Boston: Reidel, 1977.

Bunge, M. *Treatise on Basic Philosophy: Volume 4: Ontology II: A World of Systems*. Boston: Reidel, 1979.

Floyd, C. "A Comparative Evaluation of System Development Methods." In T. W. Olle, H. G. Sol, and A. A. Verrijn-Stuart (Eds.), *Information System Design Methodologies: Improving the Practice*, Amsterdam: Elsevier Science Publishers, B.V., 1986, pp. 19-54.

Korson, T., and McGregor, J. D. "Understanding Object-Oriented: A Unifying Paradigm." *Communications of the ACM*, Volume 33, Number 9, September 1990, pp. 40-60.

Lindgreen, P., Editor. *A Framework of Information Systems Concepts: Interim Report*. The Netherlands: The IFIP WG 8.1 Task Group FRISCO, 1990.

Nijssen, G. M., and Halpin, T. A. *Conceptual Schema and Relational Database Design: A Fact Oriented Approach*. Sydney: Prentice-Hall, 1989.

Olle, T. W.; Hagelstein, J.; Macdonald, I. G.; Rolland, C.; Sol, H. G.; Van Assche, F. J. M.; and Verrijn-Stuart, A. A. *Information Systems Methodologies: A Framework for Understanding*. Reading, Massachusetts: Addison-Wesley, 1988.

Shoval, P., and Even-Chaime, M. "Data Base Schema Design: An Experimental Comparison between Normalization and Information Analysis." *Data Base*, Volume 18, Number 3, Spring 1987, pp. 30-39.

Wand, Y., and Weber, R. "A Model of Control and Audit Procedure Change in Evolving Data Processing Systems." *The Accounting Review*, Volume 64, Number 1, January 1989a, pp. 87-107.

Wand, Y., and Weber, R. "An Ontological Analysis of Systems Analysis and Design Methods." In E. Falkenberg and P. Lindgreen (Eds.), *Information Systems Concepts – An In-Depth Analysis*, Amsterdam: North-Holland, 1989b, pp. 79-107.

Wand, Y., and Weber, R. "An Ontological Model of an Information System." *IEEE Transactions on Software Engineering*, Volume 16, Number 11, November 1990a, pp. 1282-1292.

Wand, Y., and Weber, R. "Towards a Theory of the Deep Structure of Information Systems." In J. DeGross, M. Alavi, and H. Oppelland (Eds.), *Proceedings of the Eleventh International Conference on Information Systems*, Copenhagen, 1990b, pp. 61-71.

Zhang, Y., and Orlowska, M. "Designing Relational Databases from NIAM Conceptual Schemas." In *Proceedings of the International Conference on Systems Management '90*, Hong Kong, 1990b, pp. 261-265.

Zhang, Y., and Orlowska, M. "A New Polynomial Time Algorithm for BCNF Relational Database Design." *Information Systems*, 1991, publication forthcoming.

Zhang, Y., and Orlowska, M. "Transforming a NIAM Conceptual Schema into an EKNF Relational Database Schema." In *Proceedings of the International Conference on Databases, Parallel Architectures, and Their Applications*, Miami Beach, Florida, 1990a, p. 563.

8. ENDNOTES

1. NIAM comprises more than its GCSD. It also provides comprehensive conceptual schema design *procedures*. We focus only on the GCSD to test the BWW model.

2. NIAM permits flexible, inter-predicate uniqueness constraints which, for brevity, are not discussed here.

3. A second constraint beside the inventory entity item indicates item numbers are character strings that may be up to ten characters long. Also, use of the "< >" symbols indicates character strings have predefined operations (e.g., an ordering operation). The third constraint beside the item name label type indicates item names are made of character strings that can be up to thirty characters long. The braces indicate no predefined operations exist for these character strings.