APPLICATION OF MODELLING TECHNIQUES

Schmidt: The main point of this session is to report experiences with the development and application of modelling tools and techniques. As Diane Smith puts it, "Have you used it yet?" Despite the amount of work in this area there seems to be a substantial lack of application experience.

The general topic has been broken into four subtopics as follows:

1. Experience with Modelling Methodologies and Tools

   1.1 Modelling Experiences

   1.2 Model to Model Transformations

2. Software Manifestations

   2.1 Database and Language Integration

   2.2 Database and System Integration

[Each subtopic was treated in a separate parallel discussion which was not recorded. Instead, discussion summaries were recorded.]

MODELLING EXPERIENCES

Schmidt: A number of people here have had experience in applying modelling techniques such as with the relationships aggregation, generalization and association. These people include Brodie, McLeod, Mylopoulos, Borgida, Carbonell, John and Diane Smith and Ted Codd. Also, Mayr, Paolini, Shaw, Thatcher, Weber and others have experience with specification techniques.

Smith, D.: Given that you have a knowledge representation scheme, a data model, or a specification language, what is your response to the following questions:

- What are your intended applications?
- What has your modelling methodology actually been applied to, from beginning to end?
- Have you modelled in the small or in the large?
- What was the impact of your intended applications on your methodology? Does it bias any parts of it, e.g., does your application make you emphasize structures as opposed to operations?
- What primitive techniques have survived the test of time? Which did not?
- What techniques do you use to acquire your modelling inputs? I have the feeling that in AI an expert is hired; in databases, users are interviewed and designers generate pages of specifications; in programming languages I think there are some interviews and a lot of introspection.

Summary of Responses to Questions

Brodie: Concerning the intended applications, we tried to distinguish the nature of applications in the three areas. Unfortunately, the programming language point of view was not represented in the discussion. It appears that AI systems attempt to model the way people think. These systems are seldom intended for large numbers of end-users or production environments. They are more for experimenting with AI theories and knowledge representation. Database designers are concerned primarily with large amounts of formatted data and transactions for large numbers of end users. There seems to be a trend now to be more concerned with modelling actions and transactions.

What about actual experience? Few tools and techniques are being used other than by the designers themselves. Almost all experience related to toy and small systems. Data abstraction, the Smiths' semantic hierarchy model and McLeod's semantic data model have been used on small, real systems. Of the experience related, only the relational model is widely used and is being used for small and large scale applications.

It appears that many modelling concepts have been around for some time. Now developers are taking an empirical approach to evaluating and applying them. Methodologies and tools are evolving and applications are being constructed. This seems to be taking some time.

What are the open modelling problems in each area? In databases there are problems with modelling concurrency, a lack of emphasis on and concepts for modelling behavior, and generally improving knowledge representation to increase semantic integrity, and coping with evolution of a model. AI people indicated problems with acquiring knowledge and putting it in an existing knowledge base. Pat Hayes mentioned difficulties in reasoning about the way people reason, and in modelling shape.

The following chart summarizes the remainder of the discussion. Ted Codd said that in his view there is a strong distinction between the relational data model and the modelling methodologies that use it. In the other approaches discussed, the model and the methodology were intimately related.

Modelling Tool or Technique	Features	Successes and Failures
relational model (Codd)	database model, facilitates data independence, uniform definition and manipulation, etc.	basis for many languages, systems, design techniques
relational model/Tasmania (Codd)	database design aid includes data dictionary features	similar to Smiths' work
formal specification	formal, precise definition	aid understanding of application, only for mathematically oriented designers
NANO-KLAUS (Hendrix)	semantic networks, English interface, First order logic, Supports deduction, random facts, type information	too early to tell
TAXIS (Mylopoulos)	aggregation and generalization has been applied to data and procedures, design by specialization/generalization, for design of interactive information systems	too early to tell
Semantic Data Model (McLeod)	for specification, user interface, database design, system integration, supports abstraction	original model too complex, being simplified
Semantic Hierarchy Model (John and Diane Smith)	specifies behaviour as well as structure, ability to express predicates over the schema as well as the database	success of relativism
Active and Passive Component Modelling (Brodie)	based on data abstraction, the semantic hierarchy model supports conceptual modelling and specification of behaviour and structure	successful application to several complex cases, e.g., over 10,000 lines of PASCAL/R code
data dictionary	database design tool, a database for information about the application database	large scale successes and failures reported
Knowledge engineering-expert systems	well understood AI techniques, strong reliance on hierarchic organization and inference engines	both successes and failures
Cognitive Modelling (Carbonell)	to formulate and test new theories, to model human memory and inference	success with some toys

MODEL TO MODEL TRANSFORMATIONS

Schmidt: The purposes of model to model transformations are: comparison, analysis, verification, communication (especially in heterogeneous systems) and finally mapping logical models onto physical models. People with experience here are Diane Smith, Hardgrave, Shaw, Katz, Reiter and others.

Questions on Transformations

Smith, D.: I have divided the questions into four groups related to the purposes of transformations.

One reason for transforming models is for comparison and analysis. For example, Hardgrave is interested in comparing data models and Reiter is interested in comparing schemas to analyze inferencing potential. The questions here are:

- what is the nature or purpose of the comparisons?
- what is to be learned?
- what mechanisms support analysis?
- what is the basis of the mechanisms?

A more pragmatic reason for transformations is for conversion, migration and evolution. Some questions here are:

- What are the conversion applications in AI, DB, and PL? I think mostly of database examples, such as are due to changes in design of DBMS.
- How are transformations specified? How many languages are needed, one for the source and one for the target model?
- What are the approaches for specifying transformations? What level of abstraction? manual or automated?

A third application of model to model transformations is implementation.

- What are the mechanisms for representation?
- How is the correctness of mappings guaranteed?

· When a representation changes how do you cope with existing data?

A fourth application for transformation is communication. This concerns heterogeneous, distributed systems in databases. Are there similar problems in AI and programming languages?

Summary of Responses to Questions

Smith, D.: We spent most of our time determining how transformations manifested themselves in AI, databases and programming languages. The main applications were for comparing and analyzing systems, converting between systems, specifying low level representations for high level specifications and communicating between systems. The following table compares these four applications across the three areas.

	PL	DB	AI
Comparison and Analysis	Informal mapping between specification languages to show equivalency	Semi-formal comparisons hampered by different ideas of what a data model is; set theory used	Metric for measuring inference is first order logic; rough metric for heuristic power is # useful inferences versus total # inferences.
System Conversion/ Migration	Program evolution tools: version control and system routines	Database design evolution; changing DBMS and/or data model	No evidence of work or of interest, probably due to small databases
Implementation	Program design tools, Program verification, multiple representations used in some application	Technique 1: Menu of representations; Technique 2: Automated design of physical database structure (better at files than databases); some dynamic and static reorganization	Simulators for first order logic; inverted files for semantic nets, somewhat ad hoc; reorganization on quiesed system
Communication	parameter passing protocols	Multiple views over logical structure; query translation; data extraction for distributed heterogeneous databases	Multiple views over knowledge representations; query generation to communicate with different databases, e.g., natural language work; message interpretation data extraction

DATABASE AND LANGUAGE INTEGRATION

Schmidt: Most projects in this area are represented here. The list includes:

APN	Relational model + LISP
PASCAL/R	Relational model + PASCAL
ADAPLEX	Daplex + ADA
CEDAR	Entity-relationship + MESA
ASTRAL	Relational model + a new high level language
RIGEL	Relational model + a new high level language
PLAIN	Relational model + a new high level language
FQL	functional model + APL-LISP-like language
XPL/S	several data models + CLU-like language

The following list of questions was prepared by Rick Cattell. The responses were collected by Lawrence Rowe who led the discussion.

0. What is the form of query constructs?

   They are primarily based on predicate calculus (functional and relational calculus) and support both tuple-at-a-time and relation-oriented operations.

1. What ADT mechanism and data model are provided, and how are they related?

   ADT mechanisms are embedded in existing languages. If one didn't exist it wasn't added. Leavenworth attempted to build a query facility using ADT's. New languages such as RIGEL and TAXIS have ADTs.

2. Is a tuple in a relation a record? How is PL data access extended/limited for DB access?

   Some yes, some no.

3. How are aggregate operations on DB incorporated? Are control constructs extended? Are higher-level operations added?

   Some languages do not support aggregate operations (e.g., count all managers that make more than $50,000), others do so as built-in and user-defined functions.

4. How is a type hierarchy integrated? How are overloaded operation names disambiguated?

   With the exception of TAXIS, the languages discussed did not support type hierarchies.

5. How are PL concurrency and DB transaction mechanisms integrated?

   They aren't. The ADAPLEX and PASCAL/R groups are looking at mechanisms for doing this.

6. How is the PL type system integrated into a stored data schema? Can new PL types be added at run time?

   All languages support PL types in the database with some application-specific restrictions.

7. Are views supported? What is the view definition language? How are updates handled?

   Most languages do not support views. Those that do (FQL, RIGEL, TAXIS) support either procedurally defined updates or automatic update translation.

8. Do DB integrity violations invoke PL exceptions?

   Some people are working on this difficult problem. Typically, they would like control before and after database recovery.

9. How do user variables refer to DB objects, e.g., on object deletion? (garbage collection)

   Buneman asks why not persistent programs? Balzer treats objects as normal LISP objects. Others do not allow references or handle the dangling reference problem.

DATABASE AND SYSTEM INTEGRATION

Schmidt: This topic concerns integrated environments for data design, definition, access, exchange, conversion, etc. This requires systems with a database in the middle with additional subsystems programming tools such as compilers, interpreters and report writers, documentation tools such as message systems, intelligent editors, plus a number of different user interface types, e.g., for experts and novices.

A number of projects investigating such systems are represented here. They are CEDAR (Cattell), CONSUL (Mark), PIE (Goldstein), 'PISA' (Albano) and KLAUS (Hendrix).

Questions Concerning Integration

0. What tools are integrated?

1. How is the integrated system made easily comprehensible? How is it made uniform?

2. Is the user a novice or expert? Can both be accommodated? How?

3. What are the important features in a data model for dealing with programs, documents, . . .?

4. Is there a mechanism to specify screen display for DB objects? On what is it based?

5. How can existing tools, e.g., a compiler, be converted to read and write from the DB?

6. Do users trust the DB to provide the protection and reliability of "old" file systems?

7. How are programs (and programmers) changed to deal with a large, long-term, concurrent DB?

8. Is the software self-describing? Is there a user help facility? How?

9. How is history maintained? How is it used?

10. What additional structuring facilities are user-visual?

11. Can DB objects of mixed-type sub-parts be displayed or manipulated?

12. Can data retrieval and manipulation languages above the DB access level be tailored to applications?

13. What DB features cause performance problem in innermost tool loops? How can they be dynamically disabled when not required?

14. How are versions of programs, documents, etc., (and parts) represented, accessed, maintained?

15. Programming Language/Database Issues: a) How are constraints expressed; caught at compile or runtime? b) How can programs deal with large, concurrently accessed, incomplete DBs? c) How are dynamic types of objects handled? d) User points to DB objects? e) Types, sets, are both views, how disambiguated?

Summary of Responses to Questions

Cattell: Two points were quite clear. First, none of the systems discussed had been used by more than a few users. In fact, with the exception of PIE, only parts of the other systems are implemented so the experience is very limited. All systems have similar goals and components which can be summarized as follows:

- Comprehensibility important. Techniques:
  - Fewer primitives, more uniformity.
  - Abstraction: filter details from user.
  - Meta-Knowledge to control display.
  - Spatial and entity-based rather than query language.
  - Natural language, explanations.

- User's view of data. Principles:
    - Uniform presentation and representation scheme.
    - Decomposed structure instead of "files."
- Procedural representation. Alternatives:
    - Use PL for everything.
    - Specialized languages.
- Version handling. Existing mechanisms:
    - Layers.
    - Data dictionary versions.
    - Incremental files and merge.

Balzer: Am I correct that using integrated systems to support cooperative projects among several people was not discussed? (Cattell) Yes.