Data Abstraction for Designing Database-Intensive Applications*

Michael L. Brodie
Department of Computer Science
University of Maryland

Introduction

There is a growing exchange of ideas amongst Artificial Intelligence (AI), Database (DB) and Programming Language (PL) researchers concerning conceptual modelling of complex, object-oriented applications. The complexity of these applications arises from complicated structural and behavioral properties which change through time; concurrent, interactive access by users with different processing needs over a shared database; information locality (i.e., DB views, PL data abstractions, AI perspectives); and primarily update-oriented transactions. Two main problems raised by these applications are:

- o managing the intellectual complexity of their design, development and evolution, and
- o defining and ensuring semantic integrity.

A number of research projects are addressing these problems by developing conceptual modelling tools and techniques (i.e., representation schemes, languages and methodologies) which take advantage of recent AI, DB and PL results. This paper outlines the approach and results of one such project, entitled Active and Passive Component Modelling (ACM/PCM), in which a comprehensive design and development methodology is being formulated. ACM/PCM is based on a semantic database model [1], from the DB area, which is being extended using semantic network, predicate calculus and actor concepts from AI and, more importantly, data, procedure and control abstractions from PL. The extended model supports hierarchies of multiply typed data abstractions needed for the intended application class.

Two features distinguish ACM/PCM from techniques with similar goals [2,3,4,5]: modelling exclusively in terms of data abstractions (components) and modelling behavior (active or procedural properties) as well as structure (passive or data properties). Both features aid in addressing the problems of complexity and semantic integrity.

Conceptual and Database Modelling

The complexity of the modelling process is substantially reduced when application properties are considered independently of their implementation. The separation is supported by two modelling phases. Conceptual Modelling is the process of producing an abstract model of an application (e.g., a representation-free formal specification). Reasoning in this phase is exclusively in terms of the application using such tools as semantic database models. A conceptual model of a criminal court application specifies the essential information content and the meaningful operations, e.g., judges, prosecutors, crimes, defendants and lawyers, together with operations appoint_judge, indite_defendant, etc. Database Modelling involves producing an implementation of a given conceptual model using tools such as the relational database model. Reasoning here concerns implementation detail and correctness with respect to the conceptual model, e.g., relations to represent judges and procedures to implement appoint_judge.

ACM/PCM was conceived as a methodology for the effective application of conceptual and database modelling tools. It uses a semantic database model to deal with database issues not addressed by methodologies such as structured programming and provides a behavioral component absent from most semantic database models. At the database modelling level, it provides a methodology for the use of high level languages that fully integrate a PL type system with a database model, e.g., PASCAL/R[6], RIGEL, PLAIN, and THESIUS. ACM/PCM also provides a straightforward mapping from the conceptual to the database level.

Behavior and Structure

Applications involving dynamic systems of complex data are perceived and should be modelled

*This work is supported in part by the National Bureau of Standards under contract NB795BCA0216 and by the National Science Foundation under grant MCS 77-22509.

in terms of their structural and behavioral properties. Structural properties characterize states of a model, e.g., data and static aspects, while behavioral properties characterize state transitions, e.g., procedures and dynamic aspects. For semantic integrity both structural and behavioral properties must be defined and maintained. In the criminal court example, a defendant is completely characterized by both its information content and the effect of the operations available on defendants.

DB and PL are beginning to place equal importance on structure and behavior. DB approaches to modelling have addressed structure with less concern for behavior. This is evident in the primitive nature of operations in database models and in the absence of behavioral design in database design techniques. In an attempt to be abstract (i.e., representation-free), the PL approach to data abstraction deals explicitly with behavior but only implicitly with structure. Clearly structure and behavior are intimately related, but neither is entirely satisfactory on its own. Structural concepts are appropriate for modelling states while behavioral concepts are appropriate for modelling state transitions.

ACM/PCM provides facilities for modelling and integrating both structure and behavior. Distinguishing conceptual and database levels aids in separating an abstract specification of structure and behavior from their implementation in data structures and associated procedures. At the conceptual level, an object's structure is specified in terms of objects from which it is composed and with which it has static relationships. In the criminal court example, a defendant is composed of name, age and address objects and must be associated with a criminal charge object; additionally, a defendant may be associated with a set of prior convictions. In object-oriented applications it is natural to view objects in terms of structural properties rather than as the result of a sequence of operations. An object's behavior is specified by the effect of the only operations available on the object. Typically, an object has one creation, one destroy and zero or more modify operations, e.g., the operations available on defendant are indite-defendant, dismiss-defendant and convict-defendant. Using the relational model at the database level an object is implemented using one or more relations while operations are implemented using procedures.

Structural properties are modelled using object schemes and are specified using a data type algebra [7]. An object scheme is a network diagram in which nodes represent objects and edges represent structural relationships. Object schemes are similar to data structure diagrams and have been used to design aggregation and generalization hierarchies in the semantic hierarchy model [1]. The data type algebra is a language for constructing complex types from simpler types. The algebra extends traditional data structuring disciplines to include predicate calculus and operations such as type generalization, union, intersection, projection, selection and join (i.e., the relational algebra raised to the type level). The algebra operations take types as arguments and produce new types. The type system supports the data abstractions of the semantic hierarchy model including type hierarchies, subtypes and multiply typed objects. In addition, it provides strong typing for improved semantic integrity.

Behavioral properties are modelled using action schemes and are specified using the data type algebra, a procedural formalism and pre- and post-conditions for each operation. In database models, behavioral properties are constrained implicitly by structural properties, e.g., triggered delete in hierarchies, storage and removal classes in the CODASYL model and relational invariants of the semantic hierarchy model. For improved semantic integrity behavior should be modelled explicitly. Action schemes are graphical aids for expressing behavioral properties of applications. An action scheme is a network in which a node represents an object and a directed edge represents an insert, delete or update action invoked by one object on another object. Action schemes are used to model actions and action relationships such as invocation dependencies and independencies.

Data Abstraction

The principle of abstraction is to suppress irrelevant details of an object under consideration and to emphasize details appropriate to the current context. Data abstraction is the application of abstraction to data objects. An abstract data type is a user defined type composed of a specification and an implementation in which the only usable information is that given in the specification. These concepts are the heart of ACM/PCM. Conceptual modelling produces specifications while database modelling produces implementations. The encapsulation of operations and structure motivated the concern for behavior and provided the basis for integrating structural and behavioral properties.

The data abstraction approach to design differs from other approaches to database design [2,3,4]. Typically, structure and behavior are considered separately. Structural properties are designed, based on query and transaction needs, and are expressed in a schema. Procedures are designed subsequently, as needed, considering only a subset of the schema and using low level insert, delete and update operations. Constraints are added to ensure that procedures do not violate the application semantics. In the data abstraction approach, structure and behavior are fully integrated. An application is decomposed into objects of interest. Each object is defined in terms of structural properties and operations or actions available on it. Insert, update and delete actions are defined for each object to ensure the semantic integrity of each object. These actions provide the only means of accessing objects. Object actions provide a complete base for constructing transactions. A transaction is a sequence of object actions designed to fulfill a particular user need.

Five Forms of Abstraction

ACM/PCM employs a semantic database model for the design and specification of conceptual models. The model's semantic primitives and organizational principles guide designers and users in designing and understanding the application. The organizational principles are forms of abstraction used to express structural and behavioral relationships. Five forms of abstraction have been recognized: aggregation (part_of), generalization (is_a), classification (instance_of) and realization (implementation_of), all from the semantic hierarchy model, as well as a new abstraction called association (member_of). Association is a form of abstraction in which a set of objects of one type is considered as a higher level set object. A set object type is a powerset of the member object type. Association is used to associate or partition a variable number of objects of a given type, e.g., a set of players forms a soccer team, a defendant may be associated with a set of zero or more prior convictions. The principle forms of abstraction (and their inverses) used in ACM/PCM are aggregation (decomposition), generalization (specialization), and association (membership). Each applies equally well to organizing structure and behavior, e.g., judge is a specialization of court_employee and hire_judge is a specialization of hire_court_employee.

ACM/PCM Experience

ACM/PCM has been used for conceptual and database modelling of several complex applications. Conceptual models have been designed for criminal court and real estate information systems. The relational database model and PASCAL/R have been used for database modelling of the criminal court application (the implementation consists of approximately 10,000 lines of PASCAL/R code). This experience has aided in refining ACM/PCM and has confirmed the proposed benefits of the approach.

Conclusion

Despite large differences in terminology and ultimate objectives, AI, PL and DB overlap significantly in the area of modelling dynamic systems of complex data. ACM/PCM has resulted from an attempt to take advantage of the distinct perspectives of each area and to integrate tools to address the issues of modelling complexity and semantic integrity.

Acknowledgement

The author is grateful to Dzenan Ridjanovic who has contributed to ACM/PCM and has implemented the criminal court system.

References

1. Smith, J. M. and Smith, D. C. P. Database abstraction: Aggregation and generalization. ACM TODS 2, 2 (June 1977).

2. Proc. NYU Symposium on Database Design, Graduate School of Business, New York University, May 1978.

3. Lum V. et al. 1978 New Orleans data design workshop report. Proc. 5th Int'l. Conf. Very Large Data Bases, Rio de Janeiro, October 1979.

4. Teory, T. J. and Fry, J. P. The logical record access approach to database design. ACM Comp. Surv. 2, 12 (June 1980).

5. Mylopoulos, J., Bernstein, P. A. and Wong, H. K. T. A language facility for designing database-intensive applications. ACM TODS 5, (June 1980).

6. Schmidt, J. W. Some high level language constructs for data type relation. ACM TODS 2, 3 (Sept. 1977).

7. Brodie, M. L. The application of data types to database semantic integrity. Information Systems 5, 4, 1980.