Computer Science in Economics and Management **2**: 3–15, 1989.

Integrated Modeling Systems

ARTHUR M. GEOFFRION
University of California, Los Angeles, CA 90024, U.S.A.

(Received: 15 December 1988)

Abstract. This paper examines in some detail the concept of an integrated modeling system. It distinguishes three main types of integration: model integration, solver integration, and integration of various utilities. Model integration is further divided into four subtypes based on a four-level model abstraction hierarchy: specific models, model classes, modeling paradigms, and modeling traditions. The paper then goes on to consider how structured modeling supports the various types of integration. The overall conclusion is that structured modeling offers an attractive approach to the design of integrated modeling systems.

Key words. Modeling system, integration, structured modeling.

1. Introduction

One of the more common phrases heard in recent years is 'integrated modeling system'. Just what does it mean? The informal answer to this question offered below determines the general plan of this paper.

The phrase has two parts. I take 'integrated' to mean *uniting things that can stand alone usefully but that are even more useful when put together*; and 'modeling system' to mean *computer software that supports (part of) the modeling life-cycle.*

The most important part of the second definition is 'modeling life-cycle'. One rendering of the life-cycle stages for a typical computer-based Management Science/Operations Research (MS/OR) application is as follows [cf. Agin (1978), Gass (1987), Hammond (1974), Chapter 10 of McFadden and Hoffer (1985), and Ramamoorthy *et al.* (1984)]:

Recognize modeling need or opportunity
Analyze feasibility and requirements
Obtain charter and make project plan
Design in detail
Build (implement, develop data)
Test (verify, validate) and revise
Prepare for use (install, train, educate)
Use (solve, operate, study model properties)
Analyze results
Report and explain findings and conclusions
Document
Maintain and update

Evaluate and review
Growth and evolution of model/system
Terminate or replace.

Of course, these stages do not necessarily occur sequentially; often several stages co-exist and earlier stages are revisited. Also, life-cycles can be quite different for one-time applications as distinguished from applications designed for repeated use over time, and for situations where prototyping is used versus where it is not used.

What is particularly evident from a life-cycle point of view is the great variety of activities requiring modeling system support. A little reflection shows that nearly everything that is required falls in three main categories: support for models, support for solvers, and utilities (tools) of various sorts. (These notions are explained, respectively, in Sections 2.1, 2.2, and 2.3.)

A modeling system worthy of the name therefore must support models and solvers and provide needed utilities. Integration across these three categories is essential.

Beyond this essential between-category integration, there remain fertile opportunities for within-category integration. That is the focus of this paper: uniting different models with one another, uniting different solvers with one another, and uniting different utilities with one another. The next section examines each of the three in turn. Then Section 3 discusses how structured modeling does or does not support these types of integration. Finally, Section 4 presents a brief summary.

2. Three Kinds of Integration

2.1. MODEL INTEGRATION

To think fruitfully about model integration, it is useful to think in terms of a natural four-level hierarchy of model abstraction:

- 'specific models' within a single model class
- 'model classes' within a single modeling paradigm
- 'modeling paradigms' within a single discipline's modeling tradition
- discipline-specific 'modeling traditions'.

More than four levels can sometimes be recognized, but a four-level hierarchy is convenient for present purposes. A few words are in order to explain each of these levels.

A 'specific model' is a completely definite instance of a model, including all data values (e.g., a particular Hitchcock–Koopmans transportation model).

A 'model class' is a collection of conceivable, similar, specific models; it is definite neither as to data values nor as to the identity or even number of items of various types, but otherwise is quite specific as to mathematical form (e.g., the class of all Hitchcock–Koopmans transportation models).

A 'modeling paradigm' is a collection of similar model classes that has established its conceptual value and influence (e.g., the class of all network flow models).

A discipline-specific 'modeling tradition' is a collection of modeling paradigms that tend to be associated with one another in the academic and practitioner communities owing to similarities of the technical apparatus they commonly involve. Of particular interest are the distinct modeling traditions of MS/OR, database management, computer programming languages, and artificial intelligence.

For future reference, here are some of the modeling paradigms of the four fields just mentioned:

Management Science/Operatons Research
- Decision Tree
- Discrete Event System
- Linear Program
- Markov Chain
- Network Flow
- Queueing System

Database Management (Tsichritzis and Lochovsky, 1982)
- Entity-Relationship Data Model
- Hierarchical Data Model
- Network Data Model
- Relational Data Model

Programming Languages (Hailpern, 1986)
- Functional
- Object-Oriented
- Procedural (Imperative)

Artificial Intelligence (Brachman and Levesque, 1985)
- Frames
- Logic
- Production Rules
- Semantic Network

Four kinds of model integration follow from the four-level hierarchy. *In each case, all higher levels are held fixed.* We now discuss each kind individually.

2.1.1. *Different Specific Models Within One Model Class*

Integration of different specific models within a single model class is one of the simplest kinds of integration.

This kind of integration often goes by the name 'consolidation' in the context of spreadsheet modeling. For example, imagine 12 activity reports for a given department that are identical in design but specific to different months of the year; now consolidate them into a single year-end report of essentially the same design. Or imagine several financial statements that are identical in format but specific to different divisions of a company, consolidated into a single company-wide financial statement of the same general format (e.g., Spiegelman, 1986).

Another example of this kind of integration occurs when two echelons of a distribution system are first modeled separately but in a similar way, and then the two models are combined into one overall multi-echelon model.

Whereas the first pair of examples result in a new specific model within the original model class (perhaps with minor revisions of the model class in some cases), the second example results in a new specific model within a new model class that is essentially a cross of the original model class with itself. Examples of the first general type can capture much of what typically is meant by the term 'aggregation'.

This kind of integration is important when summaries must be produced from similar but distinct reports or models; when it is expedient for different people or groups to build different submodels using a standardized model template; and, sometimes, when suboptimization needs to be avoided.

2.1.2. *Different Model Classes Within One Modeling Paradigm*

Integration across different model classes within a single modeling paradigm is somewhat more challenging than the first type of integration because of the greater degree of possible dissimilarity among the models being integrated.

Integration of this type occurs when two or more linear programming models are combined into one comprehensive model, as happens quite commonly.

This type of integration is also quite common among network flow models, another modeling paradigm that lends itself readily to integration. A recent example is the award-winning work at Citgo (Klingman *et al.*, 1986). The complete model "integrates the supply, distribution, pricing, financing, and sales functions of the short-term, 'downstream' petroleum products operations."

Large-scale economic models are often built using this kind of integration. See, for example, Phillips (1981), which discusses a technique for constructing a large generalized equilibrium model as a network of simpler process submodels. All process submodels fall within the general process model paradigm, and it is this similarity that makes a general integration methodology possible. Another good paper in this general vein, but with more of an optimization orientation, is Hogan and Weyant (1983).

This kind of integration is important because real problems often transcend the boundaries of individual model classes. This is particularly likely to be the case when it is necessary to deal simultaneously with different business functions. Avoiding piecemeal treatment and suboptimization may require model integration in this sense.

In the neighboring field of programming languages, the analog of a 'model class' is a computer program exclusive of any data or other detail on which to operate or with which to instantiate itself. The analog of this kind of integration is the linking of different computer programs written within, say, the procedural paradigm.

This kind of integration also arises in the neighboring field of database management in the guise of 'view integration' (e.g., Batini, Lenzerini and Navathe, 1986). Each view is expressed as a schema (model class) within a particular data model. Part of the enterprise-level database design problem is to integrate these schemata into a

single one that preserves most or all of the functionality associated with each individual schema.

2.1.3. *Different Modeling Paradigms Within One Modeling Tradition*

Integration across different modeling paradigms within a single discipline-specific modeling tradition has two distinct subtypes. The first is integration of model classes from different paradigms. The second is integration of different paradigms themselves. Both are important for reasons similar to those that apply to the previous type of integration, and are still more difficult owing to still greater dissimilarity of what must be integrated. Paradigm integration is especially valuable to the extent that it enables integration of model classes from different paradigms to be reclassified as integration of model classes from a single (more general) paradigm.

This kind of integration occurs within the MS/OR tradition when, for example, a queueing model is combined with a network flow model in order to describe a computer communication network, or when a resource allocation model is combined with a CPM-style project management model.

Damon and Schramm (1972) can be viewed in terms of this type of integration. It combines a quadratic programming model for the 'production sector', a simple nonlinear demand model for the 'marketing sector', and a deterministic financial planning model for the 'financial sector'. Integration is achieved by recasting all three submodels within the nonlinear programming paradigm. This can be thought of as integration by 'subordination' of multiple paradigms to a more general paradigm.

Another example is Federgruen and Zipkin (1984), which integrates the basic vehicle routing and resource allocation paradigms.

Examples of this kind of integration in the related field of programming languages can be found in the special issue introduced by Hailpern (1986). Conscious integration of programming language paradigms is a major contemporary research theme.

An example of this kind of integration in the related field of artificial intelligence can be found in Brachman, Fikes, and Levesque (1985), which integrates the frame and logic paradigms.

An example of this kind of integration in the related field of economics can be found in Weyant (1985), which shows how the general economic equilibrium paradigm subsumes four other modeling paradigms in the context of energy-economic modeling: variable-coefficient input-output theory, process networks, linear programming, and general nonlinear optimization.

2.1.4. *Different Modeling Traditions*

Integration across different discipline-specific modeling traditions is the fourth and final type suggested by the four-level hierarchy. Such integration typically carries with it the requirement of solver integration as well, since different disciplinary traditions usually involve distinct solver technologies.

The most common examples of this kind of integration for MS/OR are those that involve database management. Obviously one can view the data needed for a MS/OR application in terms of one of the popular data models and use a database system to manage the data. Integration occurs at a conceptual level to the extent that the MS/OR model and the data model are unified rather than separate, and at a practical level to the extent that the MS/OR software and database software are unified.

Work toward integrating the modeling traditions of MS/OR and database management includes Beulens (1986), Blanning (1985), Bonczek, Holsapple, and Whinston (1978), Burger (1982), Dolk (1986), Sprague and Carlson (1982), and Stohr and Tanniru (1980).

Integration across the MS/OR and AI modeling traditions has recently been receiving a lot of attention. See, e.g., Hokans (1984) and some of the papers presented at the ORSA/TIMS Joint National Meeting in Miami Beach, October 1986; the theme of this meeting was 'Artificial Intelligence and MS/OR: Some Unifying Themes'.

See Bonczek, Holsapple, and Whinston (1981) for a book-length discussion attempting to integrate MS/OR (from the DSS point of view) both with database management and with artificial intelligence.

This kind of integration is likely to become more important in the future as the common modeling concerns of MS/OR, database management, programming languages, and artificial intelligence come to be better recognized. See Brodie, Mylopoulos, and Schmidt (1984) for the results of a research conference on precisely this issue for the last three fields mentioned. See also Brodie and Mylopoulos (1986) for an in-depth discussion of this issue for database management and artificial intelligence.

2.2. SOLVER INTEGRATION

A solver is a method or program for purposefully manipulating a model. Several types of manipulation are important:

- 'Definitional Calculation' (as in a spreadsheet; in structured modeling this is called 'evaluation')
- 'Satisfaction' (solve a system of linear or nonlinear equations and/or inequalities)
- 'Optimization'
- 'Query Processing' (as in database management)
- 'Inference' (as in artificial intelligence).

The first three of these are associated with MS/OR, and the last two with the disciplines noted.

A solver is always paired with a model class, modeling paradigm, or even modeling tradition. Thus a solver can be categorized by the type of manipulation it can perform, per the above list of five, in conjunction with its associated particular model class, paradigm, or tradition. This categorization sets the stage for thinking meaningfully about solver integration.

Solver integration is a common concomitant of model integration across modeling paradigms because different paradigms usually have different kinds of solvers associated with them. A similar observation holds with respect to model integration across modeling traditions. These observations are sufficient to establsh the importance of solver integration. We now illustrate both observations.

Roy, Lasdon, and Lordeman (1986) provides an example of how model integration across paradigms can induce the need for solver integration. This paper combines the multi-period spreadsheet paradigm for financial planning with the optimization modeling paradigm in a mainframe environment. Associated with the financial planning paradigm is an algorithm for solving simultaneous equations, and associated with the optimization modeling paradigm are linear and nonlinear programming algorithms. Similar work has been done in the personal computer environment to marry the popular spreadsheet modeling paradigm with the LP model paradigm (Cunningham and Schrage, 1986).

The Guru package (from Micro Data Base Systems) provides an example of how model integration across modeling traditions can induce the need for solver integration. It combines solvers for spreadsheets, relational database query processing, and rule-based inference.

Integrating distinct solver technologies applicable to the same modeling paradigm can yield improved hybrid technologies. This gives another reason for the importance of solver integration. A good example in the context of optimization-oriented models is the union of implicit enumeration and linear programming technologies to produce the branch-and-bound approach that has dominated the integer programming scene for the past two decades. See also Geoffrion (1977), which explores some of the ways in which discrete/combinatorial optimization techniques can be integrated with linear/nonlinear programming techniques.

Decomposition theory from large-scale mathematical programming furnishes a powerful framework for integrating optimization algorithms for special model classes.

Birta (1984) discusses some of the ways in which simulation and optimization can be integrated.

Glover (1985) discusses several techniques from artificial intelligence that appear to be useful for integer and combinatorial programming. An early paper in a similar vein is Lauriere (1978).

Kendrick, Krishnan, and Carl-Mitchell (1984) discuss some of the particulars of sequentially integrating a database system (Ingres) and an optimization system (GAMS).

Ho, Nygard and Shapiro (1986) demonstrate that some optimization algorithms (e.g., shortest path, bin packing, simplex) can be coded in the query language of a database management system with surprisingly good performance. This opens up a possible path for integrating the previously distinct solver technologies of optimization and query processing.

A final point about the importance of this type of integration is that different kinds of solvers may be needed at different stages in the life-cycle of a modeling application.

For example, a query processor may be needed to facilitate management access to a database containing optimization results after the optimization stage. Supporting the whole life-cycle is the need behind the third and final kind of integration.

2.3. UTILITY INTEGRATION

Recall the modeling life-cycle description in Section 1. There are at least three main categories of tools or utilities that could be useful at different stages of the life-cycle, and within these categories several specific examples come readily to mind:

Communication
- Business Graphics
- Report Writer
- Telecommunications
- Text Editor

Organizing Things and Ideas
- Data Management
- File Management
- Outlining
- Project Management

Quantitative Analysis
- Interactive Data Analysis
- Mathematical Computation
- Spreadsheet Analysis
- Statistical Analysis

The advantages of having access to such utilities as an integral part of a modeling system throughout the entire life-cycle should be obvious.

Conventional modeling systems do not provide many of these facilities. Consequently, it is common practice for MS/OR professionals to use several relatively specialized tools for different functions. This involves a good deal of inefficiency.

In recent years there has emerged a class of personal computer software packages of sufficiently broad capability to provide many of the required utilities to some degree. These are the 'integrated personal productivity' packages: Framework (from Ashton-Tate), Symphony (from Lotus Development), and others (e.g., Bonner, 1985). By making one of these the host medium of a modeling system, many important activities can be supported directly (word processing, flat file data management, spreadsheets, business graphics, telecommunications) and other more specialized functions can be developed in the special programming languages that come with such packages or in standard languages via programs that can be run without leaving the package's environment. However, this approach is practical at present only so long as the amount of data involved is not too great.

UNIX has been a more practical vehicle for utility integration when large amounts of data are involved. See, for example, Balci and Nance (1987) and Childs and Meacham (1986).

Utility integration has reached a high degree of development in the context of 'programming environments' in the neighboring field of software engineering. A good survey is provided by Barstow, Shrobe and Sandewall (1984). Much of the technology created for programming environments should be applicable toward utility integration in future integrated modeling systems.

3. An Assessment of How Structured Modeling Supports Integration

This section considers how structured modeling does or does not support each of the kinds of integration discussed in Section 2. The reader needs to be familiar with structured modeling at the level of Geoffrion (1987) in order to understand what follows.

The four-level abstraction hierarchy of models introduced at the beginning of Section 2.1 has a counterpart in structured modeling:

Abstraction Level	Structured Modeling Counterpart
Specific model	Elemental detail tables (together with a schema)
Model class	Schema
Modeling paradigm	Definitional system
Modeling tradition	MS/OR (by genesis)

A clarification is in order concerning the 'paradigm' used by structured modeling. Mathematically speaking, one can view structured modeling as using acyclic, attributed, directed graphs as its paradigm; however, structured modeling aspires to be paradigm-free. It is intended as a *lingua franca* within which model classes from a wide variety of modeling paradigms can be expressed – much as English is so used, except that the language of structured modeling is much more structured and amenable to direct computer execution. The structure comes from the requirement that each model class be expressed as a system of definitions with certain properties.

3.1. MODEL INTEGRATION

3.1.1. *Different Specific Models Within One Model Class*

Integration of different specific models within a single model class seldom poses serious problems for any modeling system so long as it is acceptable for the user to perform the integration under manual direction, and so long as total model size remains within reasonable limits. If a modeling system can handle one submodel, then it should be able to handle the others since they all have the same structure. Of course, if the number of submodels or their size is significant, then the issue of efficiency arises.

Structured modeling has no particular difficulty accommodating the integration of specific models that all have the same, or nearly the same, schema. It is largely a matter of manipulating elemental detail tables. Some simple schema changes may be necessary.

3.1.2. *Different Model Classes Within One Modeling Paradigm*

Integration across different model classes within a single modeling paradigm can pose serious problems for a modeling system, depending on how flexible the modeling system is as a host for models within the given modeling paradigm.

Dealing with a wide variety of model classes is one of the explicit aims of structured modeling. To the extent that this aim is achieved, a structured modeling system will be able to represent as schemata the model classes to be integrated. Integration of these schemata ought not to be difficult because they are represented in the same language and style; it should mostly be a matter of concatenation, with appropriate editing before and after.

Section 3.2 of Geoffrion (1987) gives an example of this kind of integration wherein the classical transportation model is combined with the classical multi-item EOQ model. For further details, see Geoffrion (1989).

Structured modeling should facilitate this kind of integration in general because it is fully explicit about the thing that usually causes the greatest difficulty when one tries to combine two model classes: interdependencies among model components. With proper attention to modular structure, the final integrated model usually will preserve the conceptual integrity of the parts from which it was composed.

3.1.3. *Different Modeling Paradigms Within One Modeling Tradition*

Structured modeling is not tied to any one of the traditional modeling paradigms. Its expressive power is sufficient that it can represent models and model classes from a wide variety of different paradigms, and it can also represent many paradigms themselves. This facilitates integration across modeling paradigms, whether paradigms per se are to be integrated or just model classes from different paradigms.

Either way, integration across different modeling paradigms within a single discipline-specific modeling tradition usually reduces to the type of integration discussed in the previous subsection.

3.1.4. *Different Modeling Traditions*

The expressive power of structured modeling is great enough to encompass a variety of paradigms from a variety of modeling traditions. For example, it has been shown that structured modeling accommodates the relational and entity-relationship data model paradigms from database management, the spreadsheet paradigm, and some versions of the semantic network paradigm from artificial intelligence. It can also be shown that structured modeling accommodates paradigms from accounting, finance, marketing, and other functional areas of business.

For work on using structured modeling to integrate different discipline-specific modeling traditions, see Farn (1985) (MS/OR and DBMS) and Chari (1988) (MS/OR and AI).

To the extent that the expressive power of structured modeling crosses disciplinary boundaries, integration across different discipline-specific modeling traditions tends to reduce to the type discussed in Section 3.1.2.

3.2. SOLVER INTEGRATION

Structured modeling does not directly support solver integration at the algorithmic level. However, structured modeling can support solver integration in most aspects external to algorithmics.

For example, integrated capabilities for answering ad hoc queries (in the tradition of database management systems) and for doing definitional calculation (in the tradition of spreadsheet programs) is an explicit requirement for a structured modeling system (Section 1.2 of Geoffrion, 1987). The prototype system currently in development will achieve this when completed.

Structured modeling makes a sharp distinction between models and solvers, and recommends keeping them in separate libraries. Since standard model representations are always used, solver interface routines of considerable generality and ease of use are possible. Thus, solver integration can be well supported from the user's viewpoint.

3.3. UTILITY INTEGRATION

The ability to support most of the things that need to be done over the course of the typical modeling life-cycle is an explicit requirement for a structured modeling system (Section 1.2 of Geoffrion, 1987). If this is to be done effectively, then various utilities like those listed in Section 2.3 must be well integrated.

Structured modeling provides a novel opportunity to achieve user interface consistency across utilities because it appears to be expressive enough to permit modeling the behavior and/or invocation conditions of most utilities. Thus it should be possible to design a uniform user interface based on the structured modeling view for utility customization and use.

4. Summary

The notion of an integrated modeling system involves a tangle of concepts worth distinguishing from one another. Section 2 made these distinctions and gave a variety of examples.

Section 3 examined the aspects of structured modeling that bear on each type of integration. Overall, it appears that structured modeling presents a promising approach for achieving most kinds of integration.

Not examined at all in this paper, but certainly an important topic for designing integrated modeling systems in the future, are technical mechanisms by which the various types of integration can be implemented efficiently. Work on this topic is in progress.

Acknowledgements

This paper was prepared for presentation at the Conference on Integrated Modeling Systems held at The University of Texas, Austin, October 23–24, 1986.

The author appreciates the constructive suggestions made by Sri Chari, Dan Dolk, Georges Geis, Jim Jackson, Melanie Lenard, and Yao-Chuan Tsai.

This work was supported by the National Science Foundation, the Office of Naval Research, and the Navy Personnel R & D Center. The viewes contained in this report are the author's and not to be attributed to the sponsoring agencies.

References

Agin, N.I. (1978), 'The Conduct of Operations Research Studies', in J.J. Moder and S.E. Elmaghraby (eds.), *Handbook of Operations Research*, Van Nostrand Reinhold Company, New York.

Balci, O. and Nance, R.E. (1987), 'Simulation Model Development Environments: A Research Prototype', *J. Operational Research Soc.* **38**(8), 753–763.

Barstow, D.R., Shrobe, H.E., and Sandewall, E. (Eds.) (1984), *Interactive Programming Environments*, McGraw-Hill, New York.

Batini, C., Lenzerini, M., and Navathe, S.B. (1986), 'A Comparative Analysis of Methodologies for Database Schema Integration', *Computing Surveys* **18**(4), 323–364 (December).

Beulens, A.J.M. (1986), 'Transforming Algebraic Data Structures into Relational Data Structures in DSS', Working Paper, Rotterdam School of Management, Erasmus University.

Birta, L.G. (1984), 'Optimization in Simulation Studies', in T.I. Oren, B.P. Zeigler, and M.S. Elzas (eds.), *Simulation and Model-Based Methodologies: An Integrative View*, NATO ASI Series, Springer-Verlag, Berlin, 1984.

Blanning, R.W. (1985), 'A Relational Theory of Model Management', Working Paper 85–106, Owen Graduate School of Management, Vanderbilt University, June.

Bonczek, R., Holsapple, C., and Whinston, A. (1978), 'Mathematical Programming Within the Context of a Generalized Database Management System', *R.A.I.R.O.* **12**(2), 117–139 (May).

Bonczek, R., Holsapple, C., and Whinston, A. (1981), *Foundations of Decision Support Systems*, Academic Press, New York.

Bonner, P. (1985), 'Forks in the Road to Integration' and 'Integrated Packages', *Personal Computing* **9**(1) 82–95 (January).

Brachman, R.J. and Levesque, H.J. (1985), *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA.

Brachman, R.J., Fikes, R.E., and Levesque, H.J. (1985), 'KRYPTON: A Functional Approach to Knowledge Representation', in Brachman and Levesque (1985).

Brodie, M.L. and Mylopoulos, J. (1986), *On Knowledge-Based Management Systems: Integrating Artificial Intelligence and Database Management Systems*, Springer-Verlag, New York.

Brodie, M., Mylopoulos, J., and Schmidt, J. (1984), *On Conceptual Modeling*, Springer-Verlag, Berlin.

Burger, W.F. (1982), 'MLD: A Language and Data Base for Modeling', IBM Research Division, San Jose, Research Report RC 9639 (#42311), September 14.

Chari, S. (1988), 'Knowledge Representation Using Structured Modeling', Ph.D. Thesis, Anderson Graduate School of Management, UCLA.

Childs, C. and Meacham, C.R. (1985), 'ANALYTICOL – An Analytical Computing Environment', *AT&T Technical J.*, **64**(9), 1995–2007 (November).

Cunningham, K. and Schrage, L. (1986), 'Optimization Models with Spreadsheet Programs', The Institute of Management Sciences, Providence, RI.

Damon, W.W. and Schramm, R. (1972), 'A Simultaneous Decision Model for Production, Marketing, and Finance', *Management Science* **19**(2), 161–172 (October).

Dolk, D.R. (1986), 'A Generalized Model Management System for Mathematical Programming', *ACM Transactions on Mathematical Software* **12**(2), 92–125 (June).

Farn, C.K. (1985), *An Integrated Information System Architecture Based on Structured Modeling*, Ph.D. Thesis, Graduate School of Management, UCLA.

Federgruen, A. and Zipkin, P. (1984), 'A Combined Vehicle Routing and Inventory Allocation Problem', *Operations Research* **32**(5), 1019–1037 (September–October).

Gass, S. (1987), 'Managing the Modeling Process: A Personal Reflection', *European Journal of Operational Research* **31**(1), 1–8 (July).

Geoffrion, A.M. (1977), 'How Can Specialized Discrete and Convex Optimization Methods Be Married?', *Annals of Discrete Mathematics* **1**, 205–220.

Geoffrion, A.M. (1987), 'An Introduction to Structured Modeling', *Management Science* **33**(5), 547–588 (May).

Geoffrion, A.M. (1989), 'Reusing Structured Models via Model Integration', *Proceedings of the Twenty-Second Hawaii International Conference on System Sciences*, January 1989.

Glover, F. (1985), 'Future Paths for Integer Programming and Links to Artificial Intelligence', CAAI Report 85-8, Center for Applied Artificial Intelligence, University of Colorado, October.

Hailpern, B. (1986), 'Multiparadigm Languages', *IEEE Software* **3**(1) (January).

Hammond, J.S., III (1974), 'Do's and Don'ts of Computer Models for Planning', *Harvard Business Review* **52**(2), 110–123 (March–April).

Ho, L.-A., Nygard, K., and Shapiro, L. (1986), 'Data Base Management Systems as Solvers in Model Management Systems', working paper, CS/OR Department, North Dakota State University, December 12.

Hogan, W.W. and Weyant, J.P. (1983), 'Methods and Algorithms for Energy Model Composition: Optimization in a Network of Process Models', in B. Lev (ed.), *Energy Models and Studies*, North Holland, Amsterdam.

Hokans, R.H. (1984), 'An Artificial Intelligence Application to Timber Harvest Schedule Implementation', *Interfaces* **14**(5), 77–84 (September–October).

Kendrick, D., Krishnan, R., and Carl-Mitchell, S. (1984), 'Interfaces Between Database and Modeling Systems', Paper 84–12, Center for Economic Research, Department of Economics, The University of Texas, Austin, September.

Klingman, D., Phillips, N., Steiger, D., Wirth, R., and Young, W. (1986), 'The Challenges and Success Factors in Implementing an Integrated Products Planning System for Citgo', *Interfaces* **16**(3), 1–19 (May–June).

Laurière, J.-L. (1978), 'A Language and a Program for Stating and Solving Combinatorial Problems', *Artificial Intelligence* **10**(1), 29–127 (February).

McFadden, F.R. and Hoffer, J.A. (1985), *Data Base Management*, Benjamin/Cummings, Menlo Park, CA.

Phillips, R.L. (1981), 'A Methodology and Software System for Linking Economic Models', Decision Focus Incorporated, 5 Palo Alto Square, Suite 410, Palo Alto, CA 94304, paper presented at the ORSA-TIMS Joint National Meeting in Houston, October 11–14.

Ramamoorthy, C.V., Prakash, A., Tsai, W., and Usuda, Y. (1984), 'Software Engineering: Problems and Perspectives', *Computer*, October, 191–209.

Roy, A., Lasdon, L.S., and Lordeman, J. (1986), 'Extending Planning Languages to Include Optimization Capabilities', *Management Science* **32**(3), 360–373 (March).

Spiegelman, L. (1986), 'Firm Chooses 1-2-3 Clone to Consolidate Data', *InfoWorld*, November 3, p. 9.

Sprague, R.H., Jr. and Carlson, E.D. (1982), *Building Effective Decision Support Systems*, Prentice-Hall, Englewood Cliffs, NJ.

Stohr, E.A. and Tanniru, M.R. (1980), 'A Database for Operations Research Models', *International Journal of Policy Analysis and Information Systems* **4**(1), 105–121.

Tsichritzis, D.C. and Lochovsky, F.H. (1982), *Data Models*, Prentice-Hall, Englewood Cliffs.

Weyant, J.P. (1985), 'General Economic Equilibrium as a Unifying Concept in Energy-Economic Modeling', *Management Science* **31**(5), 548–563 (May).