Integrating Modelling Systems for Environmental Management Information Systems

David J. Abel, Kerry Taylor, Dean Kuo
CSIRO Mathematical and Information Sciences*
{david.abel,kerry.taylor,dean.kuo}@cmis.csiro.au

Abstract

Special purpose modelling packages can become more accessible and more effective for decision support when integrated into a spatial information system. Integration is made difficult by differences in the models due to scope, underlying data models, and command languages. This paper extends a federated information systems design methodology and architecture by identifying parallels of the model integration problem with the database integration problem in federated database design. A schema architecture is proposed together with associated schema translation functions. The role of a problem statement, analogous to a federated database query, is defined. Our design approach is demonstrated in HYDRA, a decision support system for water quality management.

1 Introduction

Many investigations in environmental management require the complementary facilities offered by spatial information systems and computational modelling systems. Spatial information systems (SIS) provide powerful facilities for assembly, fusion and visualisation of spatial data, while computational models are necessary to analyse the behaviour of complex physical processes.

It is difficult to integrate SIS with computational models, especially when existing modelling packages must be re-used rather than re-implemented. The systems integration task is complicated by the differences between the data models used in the modelling packages and the SIS, and the idiosyncratic command languages and data transfer facilities of independently-designed software systems. These issues are similar to issues in *schema translation* and *schema integration* for federated databases [SL90]. While there are many case studies of the integration of geographical information systems and modelling systems [GSP96], the approaches followed are typically specific to the component subsystems and to the narrow application focus of the integrated system. It is attractive to investigate more general approaches as a means of expediting design and implementation of specific applications and enabling the development of specialist systems integration tools.

A suitable candidate is the *federated information systems* approach [AKC94, AKD94]. It is based on concepts from the study of federated database (or *multidatabase*) systems [SL90], but extends those concepts to include external computational processes. Review of various integrated systems revealed that the usual types of *connector* functions are present, even if not explicitly recognised. A prototype system (HYDRA) confirmed the feasibility of a systems integration methodology that focusses on recognising the connectors needed.

This paper further develops the federated information systems methodology. We are particularly concerned with applications requiring the integration of several modelling systems. The key concept introduced is a *problem*, which has an analogous role to the *query* in a multidatabase system. The problem acts as both a representation of the data required to be predicted by the end-user and a statement of the functionality of a modelling system.

The approach is demonstrated by a case study involving design of a Spatial Decision Support System for water quality management in streams in an urban catchment.

2 A Motivating Example

To act as a running example, we consider the design of a Decision Support System (DSS) for river water quality planning. In a region subject to increasing urbanisation, a typical investigation would aim to maintain acceptable concentrations of nutrients in the streams over a planning period. The DSS would support consideration of a plan to build or upgrade sewage treatment plants.

Nutrients enter the river system in tributaries, sewage treatment plant discharges, and runoff from residential and agricultural land. The nutrients are

*GPO Box 664 Canberra ACT 2601 Australia

retained in the system by absorption in riverbed sediments and in-stream vegetation and are discharged in flow at the river system exit. The physical processes involved are complex.

A specialist class of computational modelling packages, hydrological modelling systems, can estimate nutrient levels for streams under a variety of conditions over a period of time. These systems are based on an underlying data and process model that represents the physical system as a set of objects and the functional relationships between them, typically defined by differential equations for the transport of mass and energy. The models are based on a hydrological network organisation of the objects; there is typically no recognition of the objects' geographic locations. As the river system can include many specialised but connected sub-systems (such as freshwater and tidal systems), several modelling packages may be required to deal with an extensive geographic region. Each modelling package must be customised for a given region by preparing a representation of the region using the constructs offered by its native process and data model. This requires the acquisition of extensive empirical data on the region and the estimation of property values not directly measurable.

The modelling packages typically qualify as *legacy* systems because they are large applications developed over a long period with idiosyncratic command languages and data models that differ significantly in their detail. They comprise proprietary code or are written in old FORTRAN dialects and so are not easily integrated into an open systems environment. They have large internal databases (typically implemented as native file systems) for the persistent storage of invariant data and execution output.

The DSS incorporating these packages aims to make their facilities readily accessible to an investigator who is not a specialist hydrological modeller. Consequently, a basic design intent is to allow him to specify conditions and to view model results in forms he can readily understand. A map-like representation of the region is particularly attractive, locating rivers, wetlands, land uses, and sewage treatment plants.

A fuller report of this example is given in reports of the HYDRA project [AKC94, AKD94, AKDD93] which is aimed at a DSS for water quality planning for the Sydney region.

3 Representations

3.1 Solvers and Problems

The integrated system architecture considered in this paper comprises a front-end component calling on a collection of heterogeneous, autonomous modelling packages for external computational services such as predictive simulation, optimisation and inferencing. We term the modelling packages *solvers* and the integrated system a *federated system*.

Given a partial description of a part of the physical system as values for some attributes, a solver applies numerical and other procedures to complete the description by deriving the values for other attributes included in its data model. We term the combination of a statement of attributes which are given by the user and a statement of the attributes required by the user as a *problem*.

For example, consider the following problem that might be solved by a solver for hydrological modelling. Given nitrate levels and flows of inputs from tributary systems and from sewage treatment outfalls, determine the nitrate concentration at hourly intervals for a period of two years for each stretch of the river. The solver might refer to a network tree showing the connections between tributaries, outfalls and stretches and the width, depth, length, bottom type and vegetation of each stretch to solve a set of second-order differential equations that predict the nitrate concentration for each stretch.

A solver has a limited domain of application in which it is able to derive some attribute values from a given input set. One limiting factor is the design scope of the process model underlying the solver. For example, if a solver's process model caters only for unidirectional flow of water, the solver cannot be used for river stretches where the flow is tidal. Another limiting factor is the availability of necessary data held privately by the solver. For example, if a hydrological modelling package refers to the soil types in a catchment, it cannot be used for a catchment where soils data is unavailable.

To derive all required attributes, it is typically necessary to combine several solvers, each solving a subproblem defined in terms of a physical subsystem. We term the subproblems, the selection of the solvers and the order in which they are applied as the *execution plan*. A problem is *well-defined* if it can be reduced to a set of subproblems which can be solved by the available modelling systems.

The key concepts, formally defined below, are objects, problems, solvers, execution plans and well-defined problems.

An *object* is composed of a sequence of attributes. They represent the physical properties of the object. For example, a sewage treatment plant (STP) would include attributes that describe its location and the rate at which nutrients are discharged into a river. In the definition, the *null* ($\emptyset$) value is used to represent those attributes that have no assigned value.

Definition 1. *Object*

$$Object = \langle att_1, \cdots, att_n \rangle$$

where $\forall i, 1 \leqslant i \leqslant n$, att_i *is a typed attribute value (a predefined set of values) or the null value* ($\emptyset$).

A problem, formally defined in definition 2, is a set of objects. There may be more than one instance of an object type within a single problem. For example, a problem may contain a number of STPs but there is an attribute for the STP object, such as a name, that differentiates the STPs.

Definition 2. *Problem*

$$Problem = \{obj_1, \dots, obj_n\}$$

where $\forall i, 1 \leqslant i \leqslant n$, obj_i *is an object.*

Given a problem, the attributes that require derivation from the modelling systems are those attributes whose value is not set ($\emptyset$). When all attribute values are set, then we say that the problem is solved. The formal definition of a solved problem is given in definition 3.

Definition 3. *Solved Problem*
Let

- $P = \{obj_1, \cdots, obj_n\}$ *be a problem;*
- $\forall i, 1 \leqslant i \leqslant n, obj_i = \langle att_1, \dots, att_{im} \rangle$.

The problem P *is a solved problem if*

$$\forall i, 1 \leqslant i \leqslant n, \forall j, 1 \leqslant j \leqslant im, att_{ij} \neq \emptyset.$$

Given a problem, a solver derives some, but generally not all, of the required data. A solver can be viewed as a function where both its input and output are problems. The output, or solution, has fewer null value attributes than the input. We represent a solver as a set of problem pairs – the input problem and the output problem. An additional constraint specifies that solvers can not change a non-null value – that is, it can not derive values for attributes whose value has already been derived.

Definition 4. *Solver (Modelling System)*

$$Solver = \{\langle p_1, p_1' \rangle, \dots, \langle p_n, p_n' \rangle\}$$

where $\forall i, 1 \leqslant i \leqslant n$,

- $p_i = \{obj_1, \cdots, obj_m\}$ *and* $p_i' = \{obj_1', \cdots, obj_m'\}$ *are problems;*
- *let* $\forall j, 1 \leqslant j \leqslant m$
  - $obj_j = \langle att_{j1}, \dots, att_{jl} \rangle$
  - $obj_j' = \langle att_{j1}', \dots, att_{jl}' \rangle$

  then $\forall k, 1 \leqslant k \leqslant l$,

  $$att_{jk} = att_{jk}' \text{ or } att_{jk} = \emptyset.$$
- $\exists j, 1 \leqslant j \leqslant m$, $obj_j \neq obj_j'$.

An integrated modelling system can invoke a number of solvers to derive the required data to solve a problem. The order in which the solvers are invoked is an execution plan for the problem: there may be more than one. In definition 5, we formally define an execution plan for a given problem.

Definition 5. *Execution Plan*
Let p_0 *be a problem.*
A sequence of solvers $\langle S_1, \dots, S_n \rangle$ *is an execution plan for* p_0 *if*

- $\forall i, 1 \leqslant i \leqslant n$, $\exists p_{i-1}, p_i$ *such that* $\langle p_{i-1}, p_i \rangle \in S_i$ *and*
- p_n *is a solved problem*

The integrated modelling system may be unable to solve an arbitrary problem because there is insufficient information, that is, too few given values. A well-defined problem, defined in definition 6, is a problem that can be solved by the modelling system – that is, there is an execution plan for the problem.

Definition 6. *Well-defined Problem*
Let p *be a problem. Define*

$$Well\!-\!defined(p) = \begin{cases} true & \exists \text{ an execution plan for } p; \\ false & otherwise \end{cases}$$

3.2 Solver Schemas

The solvers are equivalent to the local databases of a multidatabase system. In this section a solver is placed in the context of an extended federated database architecture [SL90]. Each solver has a *local* conceptual schema described in terms of its own *native* data model. Typically, the native data model is informal and complex, but by defining an appropriate schema translation between the model and a *common data model* we can derive a *component* schema expressed in terms of the common data model. The schema translation process is performed in a transforming processor of the software module we term a *solver driver*. The solver and the solver driver comprise the *solver subsystem*, offering an *export* schema that describes the functions provided by a solver in terms of the common data model. Unlike the databases of a federated database system, we assume that a solver's component schema coincides with its export schema, because there seems to be little advantage to constraining the functionality of a solver according to user privileges.

Notionally, an export schema for a solver includes:

- the solver data model, as a description of the object types recognised by the solver;
- a statement of the types of problems able to be solved, expressed as two lists of attributes: the given and the required (solver inputs and outputs respectively);
- a statement of the domain of application for each problem type;
- the operations to import a problem prior to solver execution and to export the result problem afterwards.

The schema for the front-end system (the "user schema") is a statement of the data model underlying the front-end system and the problems able to be developed in the front-end system. While these problems are also defined as sets of objects with given and required attributes, we do not assume that the user problems are defined by reference to the solver problems.

4 Transformation Functions

As the solver schemas and the user interface schema have been autonomously derived, passing a problem instance between the subsystems requires mapping between those schemas. Mapping object instances and attribute values across problems requires closely similar techniques to those of federated databases. Broadly, mapping of objects between the the end-user problem schema and a solver's export schema requires possibly $1 : m$ and $m : 1$ mappings, and the mapping might require recognition, specialisation, and aggregation of objects. We have encountered no cases of $m : n$ mappings. As examples,

- a river might be a single object instance in the end-user schema, but a set of reaches in a solver;
- regions with differing land uses from which water runs off into a river might be represented as separate spatially-defined entities in the user schema, but collectively treated as a single diffuse source of water and other materials by a solver;
- sewage treatment plants and factories might be modelled as separate object types in the end-user schema, but both modelled as *point sources* by a solver.

Recognition of object is also necessary. There might not be a common key for all forms of an object in the user and solver schemas, so that transformation procedures need to maintain the mappings between key values across the subsystems. For example, while the user interface subsystem might identify a river by a name (such as *Cattai Creek*), the equivalent stretch might be identified in a solver by an arbitrarily assigned integer.

5 Solver Execution Plans

The equivalent of a *query execution plan* in a federated database is a *solver execution plan*, a serially-executed sequence of functions by solvers and databases and the data assembly operations which solve a problem. To simplify the discussion, we consider only sequential execution of solvers.

A solver execution plan could be developed by an exhaustive enumeration algorithm. The algorithm incrementally adds a solver to the sequence and tests feasibility by matching its given attributes against the givens of the problem (as modified at that point), transforming the user schema to the solver schema.

In some cases, knowledge of the application domain (the physical processes and the design scope of the integrated system) will allow solver execution planning to be performed by problem decomposition. Here the problems of interest can be expressed as a set of sequential subproblems with a limited choice for solvers for each subproblem. For example, hydraulic (water quantity) modelling for a geographic region typically proceeds by subdividing the region into catchments (subregions in which all water entering drains to a single exit point). A catchment can receive inputs from a number of tributary catchments. The region can then be represented as a lattice of catchments. That is, solver execution planning for the region can be performed in terms of determining the solvers for the subproblems corresponding to each catchment and running the plans in a post-order traversal of the lattice of catchments.

6 The HYDRA System

The concepts have been tested in development of the HYDRA system, an experimental spatial decision support system for water quality planning by the Sydney Water Corporation. HYDRA (Figure 1) integrates water quality modelling systems already in place in Sydney Water. The region has been partitioned into catchments and a trunk river system. Catchments are modelled using the HSPF [BK84] or CMSS [DNBL91] systems and the trunk river system by SALMON-Q. This policy is motivated by the high cost of developing a model, acquiring data and estimating parameters for physical properties unable to be directly observed, and by matching the physical characteristics of catchments to the strengths and weaknesses of the solvers. The catchments can be organised as a lattice on the basis of the flow of water through the region. For the current (base) conditions, data is available for each solver for its assigned catchments or rivers.

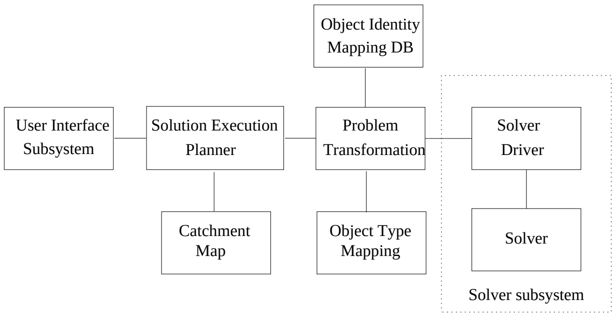

Figure 1: HYDRA

The design objective of HYDRA is to support investigation of the effects on water quality of changing land uses and of new or modified sewage treatment plants and artificial wetlands in western Sydney. The operation of the system, to the end-user, is structured into three modes: specification of a scenario (the changes from the current conditions in the region), evaluation of the outcomes of the scenario and examination of the outcomes. The principal site of interaction in the graphical user interface is a map panel. A scenario is specified by editing an existing scenario through the map. For example, a new sewage treatment plant is added by dragging and dropping an icon for a sewage treatment plant to the chosen location and changing its discharge characteristics by editing a tabular report. A local database stores existing scenarios, spatial data for map production and so on.

The concept of a scenario as a set of changes from the current state of the physical system simplifies generation of a solver execution plan. Catchments changed from the base conditions in a scenario are detected by using a spatial search to identify changes in the scenario with the catchments in which they occur.

A subproblem is passed to its assigned solver, after transformation to the solver's export schema, as the changes within the catchment. The changes are expressed as a sequence of Append, Delete and Update operations on the subproblem stored in the solver subsystem. These operations have the usual database interpretations of adding or deleting objects or changing attribute values. The solver driver then forms a description of the subproblem by edit operations on a private description to build a command set in the native command language of the solver.

7 Discussion

The topic addressed here has points of similarity with those considered in several other research fields. In decision support systems, the topic of model management systems [CHW93] deals with the description and management of collections of solvers. In contrast with most work in this area, with its emphasis on building collections of easily connected solvers, our investigation has been directed towards overcoming the problems of connecting independently-designed solvers. In the database field, the closest parallel to our approach is the external operations service of Schek and Wolf [SW93] which also draws on multi-database concepts in considering access to external computational facilities from a database system. The work presented here can be viewed as extending the utility of the external operations service by allowing significant differences between underlying data models and by providing the database designer with a high-level view of a collection of external services.

A desirable goal of the research on federated systems is an analysis and design methodology able to be applied reproducibly and reliably, despite the idiosyncratic nature of the modelling systems of interest. The architecture of HYDRA suggests that a structured approach is possible. Essentially the solver execution planner, the transformation of subproblems, and the solver driver are separable design tasks, each with a restricted scope. The solver driver design task, for example, can be approached as implementation of Append, Delete and Update operations on objects represented by the local schema of the solver. There is also the possibility of re-using a solver subsystem in a number of decision support systems.

Some difficult research issues remain. The HYDRA solver execution planner, for example, rests on the presence of a physical property (that water flows downhill) which enables a very simple problem decomposition algorithm. Intrinsically, the applicability of problem decomposition will depend on discovery of physical properties, case by case, which allow definition of a set of subproblems which can be organised as a tree and so solved sequentially. It remains an open question, perhaps addressed only by experience with building many applications, how frequently such properties can be found.

The discussion has ignored some important issues of semantics. For example, while two solvers might both estimate a value for nitrate levels at a point on a river, the estimate must be interpreted as having a certain reliability and precision which are dependent on the source data and the solution techniques used. It is unlikely that estimates from two solvers would have precisely the same meaning. The approach presented here essentially assumes that the user's requirements on reliability and precision have been taken into account in formulating the problem and solvers' schemas and in the solver execution planning algorithm.

8 Conclusions

This paper has offered further steps towards an understanding of the task of building environmental management information systems which incorporate spatial information systems and multiple, pre-existing modelling systems. The specific contributions are further definition of the mappings between data models and between process models. The key concept proposed is the *problem* which serves as both the representation of the functions of a modelling system and a statement of available input data and required output data.

The further detail of functions required in a federated system shows that, while there is a strong similarity to multidatabase systems, some significant differences exist. The major variation is the replacement of the multidatabase query execution planning function by the solver execution planning operation. The HYDRA design, with its split of the constructor function into separate model-model mapping and command-command mapping, also suggests that a two-step approach to connector functions is desirable to contain the complexity of some connector functions. Generally, the presence of a stronger conceptual foundation for federated systems and the successful implementation of a complex application integrating three modelling systems confirms the feasibility of the approach.

9 Acknowledgement

The HYDRA project is a component of the CSIRO Urban Water Research Priorities Program and has been supported, technically and financially, by the Sydney Water Corporation. Several colleagues have contributed to design and implementation of versions of HYDRA: we thank Richard Davis, Sue Cuddy, Gavin Walker, Michael Alexander, Trevor Farley, Zhexue Huang, Philip Kilby, and Graham Williams.

References

[AKC94] D. J. Abel, P. J. Kilby, and M. A. Cameron. A federated systems approach to design of spatial decision support systems. In *Proceedings of Sixth International Symposium on Spatial Data Handling*, pages 46–59, September 1994.

[AKD94] D. J. Abel, P.J. Kilby, and J.R. Davis. The systems integration problem. *International Journal of Geographical Information Systems*, pages 1–12, 1994.

[AKDD93] D. J. Abel, P. J. Kilby, J. R. Davis, and A. Deen. The spatial water modelling program: a case study in integrated spatial information systems. In *Proceedings of the Twenty-second AURISA Conference*, pages 406–411, Nov 1993.

[BK84] T. D. Barnwell jr. and J. L. Kittle. Hydrologic simulation program — FORTRAN: Development, maintenance and applications. In *Proceedings, 3rd International Conference on Urban Storm Drainage*, pages 493–502, Goteburg, Sweden, June 1984.

[CHW93] Ai-Mei Chang, Clyde W. Holsapple, and Andrew B. Whinston. Model management issues and directions. *IJDSS*, 9:19–37, 1993.

[DNBL91] J. R. Davis, P. M. Nanninga, J. Biggins, and P. Laut. Prototype decision support system for analysing the impact of catchment policies. *Journal of Water Resources Planning and Management*, 117(4):399–414, 1991.

[GSP96] M. Goodchild, L. Steyaert, and B. Parks. *GIS and Environmental Modeling: Progress and Research Issues.* GIS World Books, 1996.

[SL90] A. Sheth and J. Larson. Federated database systems for managing distributed heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.

[SW93] Hans-J. Schek and Andreas Wolf. From extensible databases to interoperability between multiple databases and GIS applications. In David Abel and Beng Chin Ooi, editors, *Advances in Spatial Databases: Third International Symposium SSD 93 Proceedings*, pages 207–237, Singapore, June 1993. Springer.