Presented at 15th Conference on Hydrology, AMS, January 9-14, 2000, Long Beach, CA

8.11 DEVELOPMENT OF A DATA ARCHITECTURE FOR THE NWS HYDROLOGIC SERVICES PROGRAM

Geoffrey M. Bonnin[1] and Daniel Urban[2]
National Weather Service, Office of Hydrology, Silver Spring, Maryland

1. INTRODUCTION

The development of effective integrated software systems implies examination and classification of the data needs of the software system's domain. The Hydrologic Research Laboratory of the National Weather Service (NWS) Office of Hydrology (OH) has been modeling the data needs of the Hydrologic Services Program of the NWS. The data architecture is expressed as a logical data model using class association diagrams, one of several artifacts of the UML approach to object-oriented analysis and design. This paper describes the rationale for this effort, the data modeling process that is being used, and a description of the products produced and their relevance to the more general software engineering approach. An example of the data model is discussed.

2. DATA ARCHITECTURE IS CRUCIAL

A system generally consists of a number of aggregations of functionality commonly referred to as applications. Large systems are composed of many applications that in their aggregate provide the general purpose of the system. In an integrated system, applications work cooperatively and are used in combinations to achieve a purpose that is broader than the domain of an individual application. This is quite different from a system where applications are not used together but fulfill complete purposes by themselves. For example, AWIPS consists of applications such as message communication, precipitation analysis, streamflow modeling, and product preparation that can be used together to produce a flood forecast. AWIPS also consists of a large number of other applications used in different combinations to produce other forecast products.

The “glue” that allows applications to be used together (to cooperate) is the ability of the applications to share information. In order for applications to share information, they must be able to communicate in a mutually understandable fashion. In other words, a necessary condition for an integrated system is infrastructure for providing a common understanding of and for sharing data. That infrastructure is developed through the implementation of a data architecture

It is clear then that a data architecture is a requirement for integrated processing systems of the nature and scope of the AWIPS.

[1] *Corresponding author address:* Geoffrey M. Bonnin, W/OH1, 1325 East-West Hwy, Silver Spring, MD 20910; e-mail: Geoffrey.Bonnin@noaa.gov

[2] Urban Architectures and COMSO Inc.

3. DESCRIBING THE DATA ARCHITECTURE

One of the keys to creating a data architecture is the ability to describe it. In the same manner as a building architect uses plans and drawings, a data architect uses graphical and textual presentations to model data.

3.1 *The OH Data Domain*

The term “data domain” refers to all of the types of data used by a system. The data domain of the NWS Hydrologic Services Program includes such things as surface observations, rivers and streams, gridded radar and satellite data, forecast model parameters, and observer names and addresses. In other words, the data domain is extensive and covers all information that is used to conduct the Hydrologic Services Program.

3.2 *Reference Information*

An often overlooked area of the data domain is reference information; the relatively static information required to fully describe any piece of data. One significant element of reference information is the “whereness” of things. For example, an individual element of data in a grid needs to have attributes describing where it is in the grid and whether the value refers to a point at that location or perhaps is an areal average for a cell centered on the grid point. The grid itself needs to be defined and located in some projection system and the projection system must be referenced to natural earth coordinates. A grid may also be related to other grids in a hierarchical relationship. Each of these pieces of “reference” information may be necessary to describe the “whereness” of the data element.

Reference information can often comprise a very large portion of a data system. A feature of the data architecture of integrated systems is that it includes a comprehensive model of reference information that once in place, can be used by all applications in the system.

3.3 *Logical Models*

A data architecture is expressed as different views or models. Two important models of the data are the logical and the physical models. The physical model of the data is a complete and detailed description of the structure of the physical data storages. An example of this could be the details of disk file layouts or relational table structures. However the physical layout of data in an implemented system often reflects compromises to improve performance and to account for the nature of the physical storage mechanism.

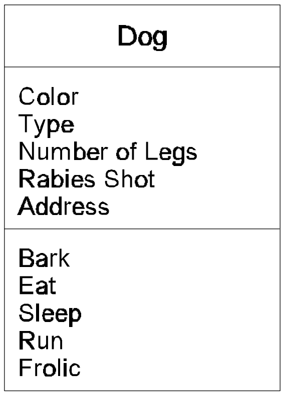

Figure 1. The "Class" Dog

A logical model is constructed prior to making those compromises. The logical model is concerned with the data itself rather than how it is stored. It provides a complete and detailed description of the data, its attributes and interrelationships without regard for physical implementation. The logical model includes such details as the "type" of the data and its constraints. For example temperature might be in degrees Celsius, stored in a floating point variable, with a precision of one decimal place and a range of 60 to -90.

Because the data domain of the data architecture is the domain of the system rather than the data domain of any single application, the logical modeling process proceeds from an understanding of the data rather than the applications. It is a "data centric" rather than an "application centric" approach. During the physical modeling stage when performance becomes the issue, a more application centric approach is adopted. However it is highly unlikely that an optimum physical model will produce optimum performance for each application because individual applications generally have different patterns of data access. Therefore, the application centric approach of the physical modeling stage involves compromises between the performance of individual applications.

Sound software engineering requires the preparation of a logical model prior to the preparation of the physical model so that a proper understanding of the data is developed prior to dealing with the complexities of performance optimization.

The importance of a logical model to an organization cannot be over-stressed. It is the comprehensive answer to the question; "What is our data?" It should be regarded as a long term intellectual asset of any organization.

OH has chosen a process for the production of a logical model by starting with conceptual definitions and frameworks of the data and then drilling down to full logical definitions in selected areas.

3.4 *Object-oriented Approach*

OH began its data modeling using entity relationship diagrams but later moved to an object-oriented approach. This switch was made to take advantage of the capability of the object-oriented approach to provide more semantic precision in the definition of relationships. The object-oriented paradigm formalizes concepts such as encapsulation and hiding of the internals of software behind an externally exposed signature. Everything that is necessary to answer the question "What can this piece of code do for me?" is captured in a class signature. The class signature consists of a class name, class data and class behavior or "methods".

Figure 1 depicts a class that represents a dog. It shows the class name as "Dog"; that there are several things we know about the class (its data) such as its color and whether it has had its rabies shots; and its behavior such as its ability to bark. Note that we do not know the dog's name because that item of data does not exist and the dog does not seem able to jump. We also do not know how it goes about barking, sleeping or frolicking only that it can exhibit this behavior.

4. RELEVANCE TO THE MORE GENERAL SOFTWARE ENGINEERING APPROACH

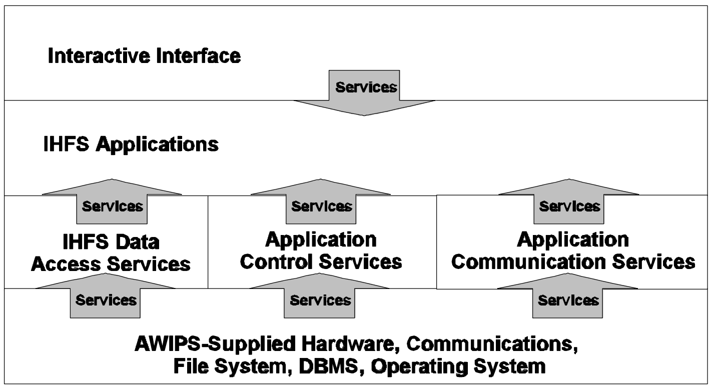
Interactive Interface
Services
IHFS Applications
Services
Services
Services
IHFS Data Access Services
Application Control Services
Application Communication Services
Services
Services
Services
AWIPS-Supplied Hardware, Communications, File System, DBMS, Operating System

Figure 2. The IHFS Software Architecture

We have adopted the architecture depicted in Figure 2 for our Integrated Hydrologic Forecast System (IHFS). The figure depicts a Data Access Services layer that is a typical feature of software architectures today.

Typically a data access services layer is concerned only with data and not with behavior. It provides all of the services associated with data access and hides the details of how this is done from applications so that the applications can be freed from these complexities. Application behavior belongs in the applications layer with the behavior of other services in layers such as the Applications Control or Applications Communications Services layers.

This presents a problem for the object-oriented approach because data and behavior are combined in the class signature. The solution that OH has adopted is also the solution that in common in the industry. That is to separate data and behavior into two separate but related classes or components (Coad, 1999). One of these classes is the "pure data class" associated with the data access layer and that inherits data access behavior from within the data access layer. The other class contains both the class behavior and the data, where the data is inherited from the pure data class. The second class is associated with the applications layer. With this separation we can apply an object-oriented approach to the data modeling problem.

5. CREATING THE DATA ARCHITECTURE

Creation of a data architecture requires appropriate skills, processes and tools. The following paragraphs describe the approach in use at the Office of Hydrology.

5.1 *The Skills Required*

The skills required for preparing a data architecture fall into three categories, data modeling, domain knowledge, and leadership. A combination of these skills may be applied by a single individual.

Data modeling is a highly developed skill requiring extensive experience and detailed knowledge of the logic and patterns of data models. For a large data model such as is required for an integrated system, experience in the development of large data models is also a requirement. The skills of a data modeler are quite different from those of a database administrator. A database administrator is concerned with the operation and characteristics of the database management software used to manage the implementation of a physical model and its hardware platform. The data modeler must be able to disregard physical issues during the logical modeling stage and then focus on them while retaining the essence of the logical model during the physical modeling stage. This intellectual compartmentalization of the problem is not a simple task. During the physical modeling stage the data modeler will draw upon the knowledge and skills required of a database administrator and on those of the application and system designers. The data modeler must have strong active listening and semantic skills to draw out meaning from domain experts from whom a comprehensive understanding of the domain data is obtained.

The domain expert is classed as expert because there must be a comprehensive understanding of the semantics of the data. It is likely that there is no single person with a sufficiently comprehensive knowledge of all of the data that must be modeled and so the data modeler must synthesize the information gleaned from often overlapping sources who provide shades of meaning which are often different, and occasionally contradictory. For example the familiar term “observation” is used to describe items ranging from the value of a temperature reading to a textual message encoded according to a particular code form and containing a set of information. Domain knowledge can also be drawn from documentation - if it exists.

Leadership skills are brought into play to identify sources of domain knowledge and to help the data modeler determine whether the information gathered is relevant to the modeling of the data domain. When combined with the data modeling skills, leadership skills are necessary to assess and drive toward completeness and comprehensiveness of the data model.

5.2 *The Process*

The data modeler goes about the task by reading documentation and discussing the data with domain experts, synthesizing what has been learned into models, and then presenting the models back to the domain experts see if they agree with the interpretation. This is an iterative and informal process. It is conducted one-on-one or with perhaps three people present. It is not a process that is conducive to large group or formal meetings. Toward the end of the process, the more formal design review involving a greater number of people is valuable.

At the Office of Hydrology a single person was engaged as the data modeler. An extensive set of documentation exists, both as textual documentation of structured data file systems and application input/output characteristics, as well as entity relationship diagrams and a data dictionary of relational database tables. The data modeler began by reviewing existing documentation and then moved progressively to iterative one-on-one discussions with about a half dozen domain experts. These discussions focused on narrow subject areas. However the subject areas and the general model state were continuously reviewed by one or two key domain experts to help clarify semantic differences and to review the overall model structure. During the discussions additional documentation was revealed that was then used to augment the developing model.

Systems tend to exhibit recurring structural patterns. The object-oriented approach is conducive to the use of recurring patterns and one of the skills of the data modeler is to recognize and use existing patterns. OH has used the patterns developed by the Open GIS Consortium for describing geospatial information (Buehler, 1996).

5.3 *The Tools*

The data model is expressed using graphical and textual presentations. Figure 3 presents a “class association diagram” that is one of the diagrammatic forms available in the Unified Modeling Language (UML) (Rational Software Corp., 1997). UML is a combination of notation standards and an approach for capturing and expressing knowledge (Alhir, 1998). The class association diagram is the primary diagram of UML and is used exclusively by OH in the development of the data architecture along with textual information such as a data dictionary and engineering notes.

OH uses a commercial Computer Aided Software Engineering (CASE) tool for drawing the diagrams and documenting the data elements. We also use it for class association diagrams of the application and services software and to directly generate computer code. The CASE tool is much more than a drawing tool structured to implement the notation and semantics of UML. It also provides powerful validation checks, a consolidated repository of the design information, automated techniques for direct code generation, and configuration and version control mechanisms. Some inexperienced modelers have tried to use simple drawing tools that are useful for very small projects on large modeling projects and have become bogged down in the mechanics of drawing diagrams. They have lost their focus on the

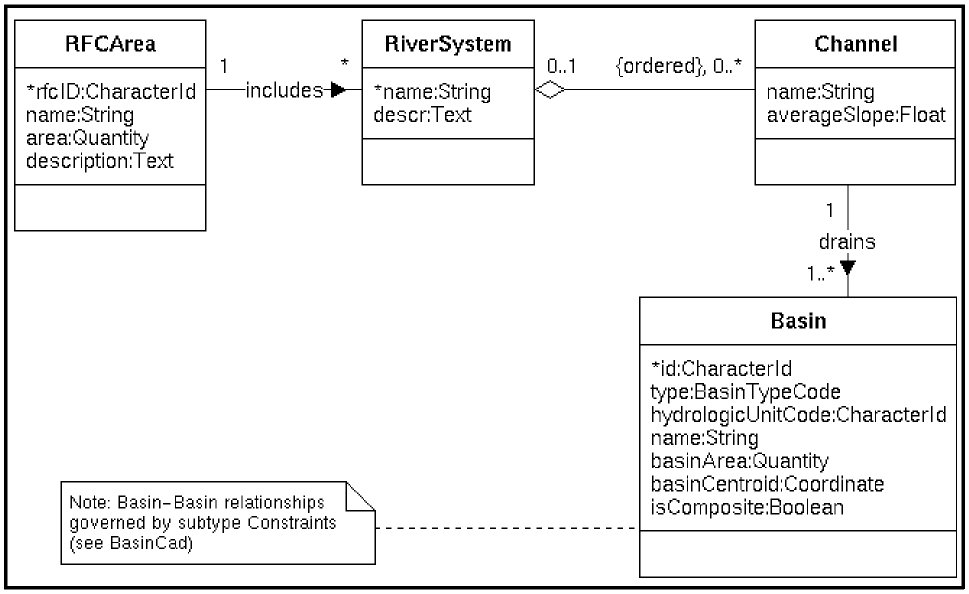

Figure 3. River System Class Association Diagram

modeling task and have incurred large overhead costs in the process.

6. SAMPLE PRODUCT

Figure 3 is a UML Class Association Diagram. It presents a fragment of the logical data model of a river system illustrating four of the class templates (signatures) and their associations.

RFCArea is a class that models the geographic area of responsibility of an NWS River Forecast Center. On other diagrams, this class would be shown with all of it geospatial associations. The RFCArea signature shows data attributes consisting of an RFC Identifier, a name, an area and a free text description. The “type” of each attribute is defined, for example, area is of type “Quantity” which is further defined elsewhere in the logical model. The data item “area” could be computed on the fly or stored in the physical implementation however this is a physical not a logical modeling issue.

RFCArea is associated with RiverSystem in a one to many relationship. In other words, there are many river systems related to each RFC area. A river system is composed of many tributaries, streams and channels. The association with Channel is a zero or one to zero or more aggregation. In other words, each river system is composed of channels but the river system object can exist without a channel having been defined and vice-versa. The dendritic nature of channels in river systems is shown on other diagrams.

There is a one to one or more relationship between Channel and Basin. In other words, each channel is related to at least one and maybe more than one catchment areas. Two interesting data attributes of the Basin signature are hydrologicUnitCode and isComposite. The hydrologic unit code is a river basin code assigned by the U.S. Geologic Survey and is a commonly used identifier for river basins. isComposite is a code used to decide if the basin is composed of other basins. That recursive association is shown on other diagrams elsewhere in the model as mentioned in the note on the diagram.

Together, the features of the class association diagram shown in Figure 3 (and elsewhere) provide a well-defined model of the data and interrelationships of information associated with river systems. Other features are not illustrated in this figure, however the diagram captures and presents in a clear and concise manner a large body of that information.

7. CONCLUSION

The development of a data architecture is key to the success of integrated systems. An object-oriented approach that focuses on data-only classes provides a powerful tool for the development of q data architecture that is consistent with common software architectures. However, the approach relies on the knowledge and skills of experienced data modelers and domain experts, and effective processes and CASE tools for its success. The Hydrologic Research is applying these resources and processes to the development of a data architecture for the Hydrologic Services Program of the NWS.

8. REFERENCES

Alhir, Sinan Si, 1998: *UML in a Nutshell*. O’Reilly & Associates, Inc., 273 pp.

Buehler, K, and McKee, L, editors, 1996: *The OpenGIS^{TM} Guide*. Open GIS Consortium, Inc.

Coad, P, and Mayfield, M, 1999: *JAVA Design: Building Better Apps and Applets*. Yourdon Press. 303 pp.

Rational Software Corp. and Partners, 1997: *Unified Modeling Language Version 1.1.* The Object Management Group.

Rumbaugh, J, Blaha, M, Premerlani, W, Eddy, F, and Lorensen, W 1991: *Object-Oriented Modeling and Design*. Prentice Hall, Inc., 490 pp.