Prototyping the Data Access Layer of the Integrated Hydrologic Forecast System

For presentation at the Fourth International Conference on Hydroinformatics, July 23-27, Iowa City, Iowa

G. Bonnin
National Weather Service, Office of Hydrology, Silver Spring, Maryland, USA

J. Gofus
National Weather Service, Office of Hydrology, Silver Spring, Maryland, USA

Y. Qu
COMSO Inc., Greenbelt, Maryland, USA

D. Urban
Urban Architectures, Inc., McLean, Virginia, USA

ABSTRACT: The Hydrologic Research Laboratory of the National Weather Service Office of Hydrology is developing its next generation of software for the conduct of the National Weather Service Hydrologic Services Program. The Integrated Hydrologic Forecast System (IHFS) project is currently focused on developing and prototyping software and data architecture components of the system. The prototype involves an element of operational functionality that exercises the data access layer as well as an object oriented approach to analysis and design. The prototype is designed for testing as an integrated component of the current operational forecast system. The IHFS software architecture is highly modular and considers a gradual and planned transition from current operational software to the new system. This paper presents progress in developing a prototype of the data access layer and the logical data model. The rationale for this effort is described and details of the data access architecture and the data access prototype are presented.

1 INTRODUCTION

The Hydrologic Research Laboratory (HRL) of the National Weather Service (NWS) Office of Hydrology (OH) is developing the Integrated Hydrologic Forecast System (IHFS), its next generation of software for the conduct of the NWS Hydrologic Services Program. Current software consists of code that was initially developed almost 20 years ago, which as been continually augmented including recent major additions. The older code is based on old mainframe architectures and technologies and has become increasingly costly to maintain and enhance. Furthermore, there are gaps in the functionality of the nationally supported software that require individual NWS field offices to create their own locally developed applications in order to provide end-to-end support for production of hydrologic services.

The IHFS will overcome these problems by recasting the current functionality using modern software technologies. The effort will reduce maintenance costs and make it easier to

introduce new science and techniques to the operational system.

An important consideration is the size, complexity and operational nature of the current system. There are over 2 million lines of code in the current system. It is an integrated system providing a broad range of functionality ranging from data decoding, ingest and management through hydrologic, hydrometeorologic and hydraulic calibration, modeling and display, to forecast preparation and dissemination. The current system represents hard won lessons learned over many years in both the science and user operations concept areas as well as in the areas of software design. The system is used every day at 13 River Forecast Centers and over 120 Weather Forecast Offices by professional hydrologists and meteorologists who through formal training and on-the-job experience are intimately familiar with its use. It is an operational system that is relied upon for the safety and well being of the public and is subject to extensive configuration management. In summary, there are significant issues to be considered in making changes to the system.

2 THE IHFS ARCHITECTURE

The planned IHFS will not replace the current system. Rather, modern software technologies and software engineering practices will be applied to recast and extend the current functionality. Current software will be reused to the greatest extent practical with new code being developed where reuse is not practical. IHFS software architecture is based on object oriented concepts. It will be highly modular and considers a gradual and planned transition from current operational software to the new system. Figure 1 shows the major architectural components of IHFS. The diagram shows that individual independent applications exist within the system. However they rely on services provided by other architectural layers of IHFS. This paper focuses on the Data Access Services layer.

3 THE DATA ACCESS PROTOTYPE

A key component of IHFS is the data access services layer. It is one of the aspects of legacy software that is rooted in the architectures of old mainframe technology and which consumes a requires significant maintenance. Experience gained during the merging of the WFO and RFC databases to produce the initial IHFS database (Roe et al., 1998) shows that application programmers were required to spend a considerable amount of time modifying applications software to accommodate change to the IHFS_DB. It became apparent that we could not afford to

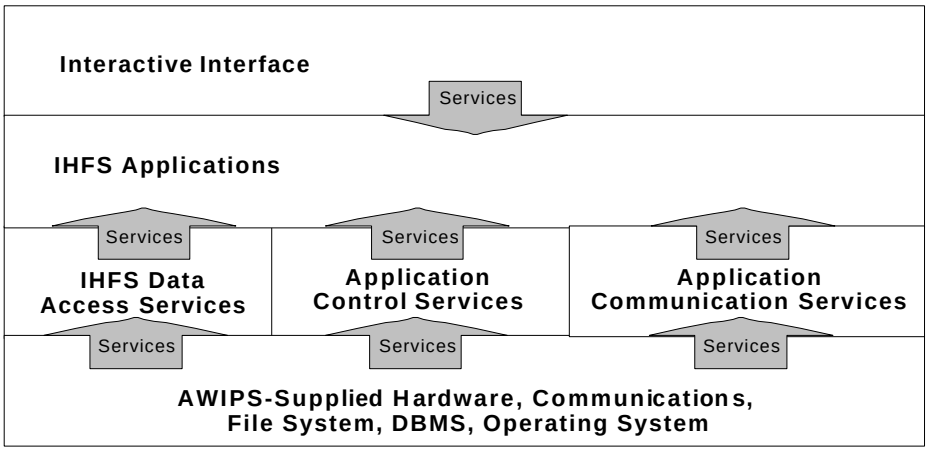

Figure 1: IHFS High Level Architecture

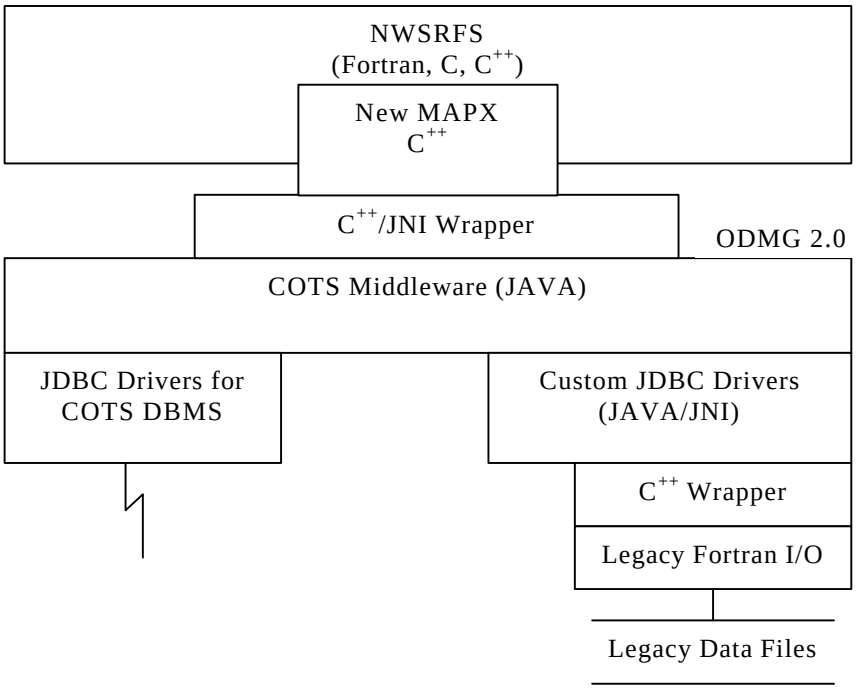

Figure 2: Prototype Data Access Services

expose applications software to continual changes in database design. In order to avoid this exposure, address one of the major components of IHFS, and promote the fundamental software design principle of modularity, we have chosen to develop a prototype of the IHFS data access layer.

The data access prototype is to demonstrate that such an access layer can be built, with the emphasis on proof-of-concept. This prototype is not likely to be the complete data access layer. However, the lessons learned from the prototype will be used subsequently to build the IHFS data access layer.

The data access services are responsible for retrieval, storage, and management of all data used by the IHFS. The data access services isolate application programs from the physical location and structure of data, provide a logical data structure for both the storage and retrieval of data, provide both temporary and persistent data storage, filter requested data based on the run environment (e.g., owner), annotate new and updated data with information from the run environment, manage buffering of data needed for I/O efficiency, ensure data values are synchronized across all programs in a run, manage data archival (and retrieval of archived data), delete expired data, enforce data access security permissions by user and run, ensure referential integrity of data, and verify the consistency of data across facilities (Computer Sciences Corporation, 1999).

The data access services are structured as logical storage definitions, APIs, and utility programs. Logical data storage definitions define the logical data storage structure that is used by application. The logical data structure will be based on hydrologic concepts and needs, not on the physical storage of data. (Bonnin et al., 2000).

4 PROTOTYPE FUNCTIONALITY

We chose the MAPX function from the current National Weather Service River Forecast System (NWSRFS) to realistically evaluate the affect of the software changes. MAPX computes mean areal precipitation using gridded NEXRAD radar precipitation

estimates. The original version is written in Fortran. The new version is written in C^{++} using an object oriented design.

The new version of MAPX is integrated into the legacy NWSRFS as a new modular function that has the same interface and parameter set as the original but a different name. This allows operational sites to implement the prototype with virtually no changes to the system parameters.

5 PROTOTYPE DESIGN

The design of the prototype is shown in Figure 2. One of the functions of the data access services layer is to isolate application programs from the physical location and structure of data. The ODMG 2.0 interface (Cattell et al., 1997) fulfills this requirement and we chose a middleware commercial off-the-shelf (COTS) product to provide the API and the bulk of the functionality. The middleware uses JDBC and assumes a relational model for connectivity to the data.

Most COTS database management systems provide JDBC drivers allowing us to complete the thread to the data however the legacy data files are custom designed indexed files that were developed in the batch mainframe era. In order to complete the thread to the legacy data files we have designed our own JDBC drivers based on the SimpleText model (Patel et al., 1997) Our custom JDBC drivers preserve the legacy I/O software, illustrating the software reuse goal of IHFS.

Once the prototype has been implemented, we will be able to eliminate the physical data storage of the legacy files and replace them with COTS database products, further reducing the amount of custom code that must be maintained. The insulating features of the data access services layer will allow us to make the change without affecting any of the applications code that uses the layer.

Bonnin et al., 2000 have described the approach for developing the logical data model for IHFS. The new MAPX component uses the objects from the logical data model and the COTS middleware instantiates and manages them based on service requests through the ODMG 2.0 interface.

A CASE tool is being used for object oriented analysis and design (Rumbaugh et al., 1991) and for code generation. The CASE tool implements the Unified Modeling Language (UML) (Rational Software, 1997, Alhir, 1998)

6 LANGUAGE ISSUES

The prototype involves a variety of languages; Fortran, C, C^{++}, JAVA and JNI as well as interface standards such as JDBC and ODMG 2.0. The use of these languages is shown on Figure 2. The mixture of languages allows preservation and reuse of legacy code at the same time as introduction of state-of-the-art software technologies. The language mixture has been achieved using software wrappers and carefully designed procedures for binding executables. The module wrappers involve careful treatment of language differences particularly in the area of memory structures and pointers to preserve data integrity across the language interfaces.

7 MAPX DESIGN

Figure 3 presents the collaboration diagram for the object oriented version of MAPX. The collaboration diagram is one of the design artifacts of the UML approach to object oriented analysis and design. The diagram shows the sequence of interactions between the various objects that are combined to produce the MAPX function. The diagram also shows the explicit use of the object signatures.

The object hierarchy has been specifically designed to allow new computational methods to be added by expansion of the object model. This feature will make it relatively easy to add new functionality in the future.

8 CURRENT STATUS

At the time of writing the prototype exists in several processing threads:

- The new version of MAPX has been integrated with NWSRFS and has been tested successfully with a test harness for data access. The computations performed

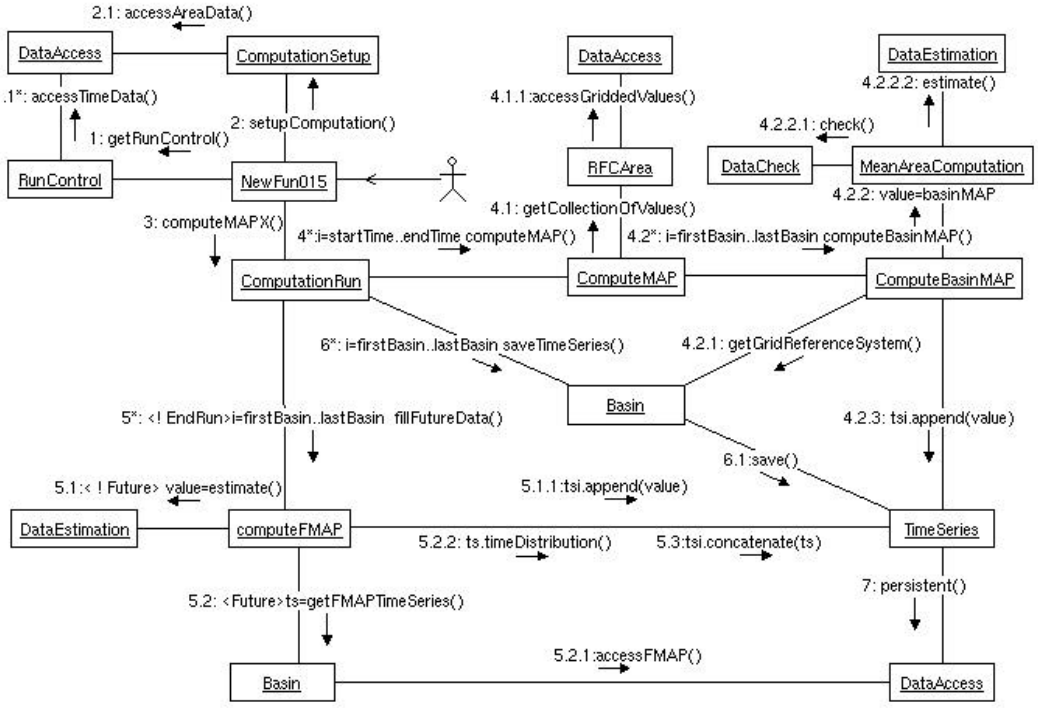

Figure 3: MAPX Collaboration Diagram

by the new object oriented version have been confirmed against the legacy version.

- A test application can successfully manipulate objects from the IHFS data model with a processing thread running from the applications layer through the middleware to a COTS relational database management system.
- A test harness can use the middleware and the custom JDBC driver to access data in the legacy data files.

The custom JDBC driver must be expanded to accommodate data in the full range of legacy data files. When this is complete, a complete integration test can be performed and performance testing and tuning can be done.

REFERENCES

Alhir, S, "*UML in a Nutshell*", O'Reilly & Associates, Inc., 273 pp., 1998.

Bonnin, G., and D. Urban, "Development Of A Data Architecture For The NWS Hydrologic Services Program," *16th Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, 2000.

Cattell, R, and D. Barry, editors, "*The Object Database Standard: ODMG 2.0*", Morgan Kaufmann Publishers, Inc., 270 pp., 1997.

Computer Sciences Corporation, "*Integrated Hydrologic Forecast System (IHFS) Software Architecture*", contract deliverable prepared for the National Weather Service, Office of Hydrology, Hydrologic Research Laboratory, 1999.

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

Roe, J., G. Bonnin, M. Glaudemans, C. Gobs, and P. Tilles, "Recent Database Developments At The National Weather Service Office Of Hydrology", *14th Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, 1998.

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

Patel, P and K. Moss, "*Java Database Programming with JDBC*", 2nd Edition, Coriolis Group Books, 491 pp., 1997