Available online at www.sciencedirect.com
ScienceDirect
Environmental Modelling & Software 22 (2007) 1805–1810
Environmental Modelling & Software
www.elsevier.com/locate/envsoft

Short communication

Implementation of an object oriented data model in an information system for water catchment management: Java JDO and Db4o Object Database

Andrea Leone*, Daoyi Chen

Department of Engineering, University of Liverpool, Liverpool L69 3GQ, UK

Received 26 October 2006; received in revised form 23 May 2007; accepted 29 May 2007
Available online 3 July 2007

Abstract

Object-oriented technologies are playing increasingly important roles in every level of software application for water resource management and modelling, except for data management levels where the relational logic is still the uncontested choice of information system developers despite the object–relational impedance mismatch. In this paper, we would like to present our experience concerning two different technologies for developing the object-oriented data management layer in information systems for water resources management: (i) the Java solution to obtain transparent persistence, the Java Data Object (JDO) technology; (ii) a purer object solution with a light open source Object Database, Db4o. The process for implementing the two technologies in a Java-based hydro-information system is described, and the two different solutions were analysed and compared.

Keywords: Water catchment; Water resources management; Information system; Object-oriented database; JDO; Java; Object-oriented data model; Object–relational impedance mismatch; Open source; GIS

1. Introduction

It has been demonstrated by Spanou and Chen (2000, 2001, 2002) and others (Maidment et al., 2002) that object oriented (OO) technologies offer great benefit to river water quality and catchment hydrological modelling. Recently, hydro-information systems (HISs) have been developed by Chen (2002) and Leone et al. (2006), evolving to an OO based system with extensions of OpenGIS standards (OpenGIS, 2006) and connections to various relational database management systems (RDBMS). Such modern HIS can be seen as a general framework and data exchange environment, embedding the following common subsystems: GIS, simulation models and management tools, databases and other external data sources, and a user-friendly interface. These different components of hydro-information systems are currently being developed with the use of OO logic and technology except for the data layer, where RDBMS are still the preferred solution in terms of the reliability of performances.

These integrated tools allow end users and developers to analyse real-time environmental data, and through integrated models, to assess environmental trends and possible early warnings (flooding, environmental hazards, etc.). The increasing complexity of environmental data, also in terms data structure of data models, amount of information and networking solutions (LAN, internet, etc.), needs the power of the OO logic to be managed.

In general, in modern information systems for water resources management, the data model is designed with OO logic and is then integrated in the system as persistent layers through manual object–relational (O–R) mapping with RDBMS. The O–R mapping is not standardised and the interface is complicated by the ''impedance mismatch'' between the domain object model of the application and the relational model of the RDBMS (Ambler, 2006).

* Corresponding author.
E-mail address: andrea.leone@liverpool.ac.uk (A. Leone).

1364-8152/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.envsoft.2007.05.016

"Impedance mismatch" is a definition originated from electrical engineering and is used in system analysis to identify the inadequate or excessive ability of one system to accommodate input from another (Ambler, 2006). It is caused by the fact that OO logic is based on software engineering principles that model the objects in the problem domain, while the relational model is based on mathematical principles that organize data for efficient storage and retrieval (Paterson, 2004).

Therefore, the incongruence between the "relational logic" of data layers and the "object logic" of other components of the HIS needs to be solved to increase the overall performances. The obvious solution is to develop a data layer with an object-oriented logic.

In the present work, two different object oriented solutions for data management will be tested: the Java Data Object (JDO) technology and an Object Database (ODB). The test bed is a Java-based hydro-information system (3O-HIS) (Leone et al., 2006). This information system has been developed using only open source technologies and software; its evolution will follow the same path.

2. Design of the software architecture

2.1. System evolution towards object oriented persistence

Technology and implementation choices solutions are oriented to our information system as an alternative to its first and more conventional solution where the information system (IS) was interfaced and "manually mapped" with an RDBMS.

With regards to Java technologies, the core language of the IS, and more general OO technologies, persistence describes the ability of an application or service to transfer the state of transient objects to some type of data storage. According to the Object Database Management Group (ODMG, 2000), transient objects are stored in and managed by the runtime system. They cease to exist and are removed from the volatile memory at the end of a process.

In order to improve the IS environmental data management capacities, the ultimate goal is to have an OO implementation of the logical data model in the data layer, built of persistent objects, directly managed from the IS.

2.2. System requirements

The objective of the present study is to test two different solutions for OO data management solving the O–R impedance mismatch of the system. Besides, the OO data layers will implement the following functionality requirements:

- capacity to manage and store objects;
- adaptability in the number of objects and storage capacity;
- working on both sides of client–server architectures;
- managing complex queries, similar to the query capacity level of SQL (query language of RDBMS);
- real-time data management from data sources across the Internet or sensor networks;
- open source software based; and
- ability to integrate existing data sources (stored in RDBMS) with data object model.

2.3. Choice of persistence standard

Persistence is the ability of data to outlive an instance of a program (Bauer and King, 2006). There are several open source technologies available to implement persistence standard in Java-based information systems. Some of these technologies, used to implement OO persistence in general and Java object persistence in particular, such as Hibernate (Bauer and King, 2006) or Java Data Object (JDO) (Jordan and Russell, 2003), are designed to provide the developer with transparent persistence; the application deals with persistent objects without the need for SQL to be embedded in the Java code. Another solution like Container Managed Persistence (CMP) has similar performances for Enterprise JavaBeans (EJB) containers, but it is not a general persistence facility for the Java platform. In all these technologies, the objects are "automatically" mapped to tables of an RDBMS by the underlying framework. The developer does not have to deal with time and performing consuming O–R mapping operations.

To define the mappings, the developer needs to create descriptors, typically XML files. Inheritance and many-to-many relationships augment complexity as these cannot be conceptually managed in the relational model. The more complex the data model the more difficult the O–R mapping.

Purer solutions, conceptually different from O–R mapping, are databases based on the object model. An Object Database Management System (ODBMS) is a DBMS that supports the modelling and creation of data as objects.

There is no official standard for ODBMS. The *de facto* standard is the final release of the last Object Database Management Group (ODMG, 2000), the ODMG 3.0. The ODMG Java binding has been now superseded by JDO.

ODBMS combines the elements of object orientation and OO programming languages with database capabilities. They extend the functionality of OO programming languages (e.g., C++, Smalltalk, and Java) to provide full-featured database programming capability. The result is a high level of congruence between the data model for the application and the data model of the database, resulting in less code (25–35% less), more natural data structures, and better maintainability and reusability (McClure, 1997).

In Fig. 1a, classes of an OO IS have to be mapped onto RDBMS tables; additional tables are needed to represent OO relationships in relational logic. In Fig. 1b, classes of an OO IS are directly stored in an ODBMS.

3. Implementation and case study

Two different kinds of technologies, the transparent persistence and the ODBMS, have been implemented in the IS data model in order to analyse their performances dealing with