SERF: ODMG-Based Generic Re-structuring Facility *

E.A. Rundensteiner, K. Claypool, M. Li, L. Chen, X. Zhang, C. Natarajan, J. Jin, S. De Lima, S. Weiner
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609
tel.: (508) 831–5815, fax: (508) 831–5776
{rundenst,kajal,mingli,lichen,xinz,chandu,jing,delima,weiner}@cs.wpi.edu

1 Overview

The age of information management and with it the advent of increasingly sophisticated technologies have kindled a need in the database community and others to *re-structure* existing systems and move forward to make use of these new technologies. Legacy application systems are being transformed to newer state-of-the-art systems, information sources are being mapped from one data model to another, a diversity of data sources are being transformed to load, cleanse and consolidate data into modern data-warehouses [CR99].

Re-structuring is thus a critical task for a variety of applications. For this reason, most object-oriented database systems (OODB) today support some form of re-structuring support [Tec94, Obj93, BKKK87]. This existing support of current OODBs [BKKK87, Tec94, Obj93] is limited to a *pre-defined* taxonomy of *simple fixed-semantic* schema evolution operations. However, such simple changes, typically to individual types only, are not sufficient for many advanced applications [Bré96]. More radical changes, such as combining two types or redefining the relationship between two types, are either very difficult or even impossible to achieve with current commercial database technology [Tec94, Obj93]. In fact, most OODBs would typically require the user to write ad-hoc programs to accomplish such transformations. Research that has begun to look into the issue of complex changes [Bré96, Ler96] is still limited by providing a *fixed* set of some selected (even if now more complex) operations.

To address these limitations of the *current* re-structuring technology, we have proposed the SERF framework which aims at providing a rich environment for doing complex user-defined transformations *flexibly*, *easily* and *correctly* [CJR98b]. The goal of our work is to increase the usability and utility of the SERF framework and its applicability to re-structuring problems beyond OODB evolution. Towards that end, we provide *re-usable* transformations via the notion of **SERF Templates** that can be packaged into **libraries**, thereby increasing the *portability* of these transformations. We also now have a first cut at providing an assurance of consistency for the users of this system, a semantic optimizer that provides some performance improvements via enhanced query optimization techniques with emphasis on the re-structuring primitives [CNR99]. In this demo we give an overview of the SERF framework, its current status and the enhancements that are planned for the future. We also present an example of the application of SERF to a domain other than schema evolution, i.e., the web re-structuring.

*This work was supported in part by the NSF NYI grant #IRI 94-57609. We would also like to thank our industrial sponsors, in particular, IBM for the IBM partnership award and Informix for software contribution. Special thanks also goes to the PSE Team specifically, Gordon Landis, Sam Haradhvala, Pat O'Brien and Breman Thuraising at Object Design Inc. for not only software contributions but also for providing us with a customized patch of the PSE Pro2.0 system that exposed schema-related APIs needed to develop our tool.

2 OQL-SERF: Our ODMG Based System

2.1 System Architecture

In order to validate our concept of SERF transformations [CJR98b], we have developed a working system, called OQL-SERF [CJR98a]. OQL-SERF serves both as proof of concept as well as helps to explore the suitability of the ODMG standard as the foundation for a template-based re-structuring framework. For OQL-SERF, we have used an extension of Java's binding of the ODMG model as its object model, our binding of the Schema Repository for the Metadata Dictionary and OQL as the database transformation language. OQL-SERF is written entirely in Java and uses

Object Design Inc.'s 100% Pure Java PSE as its persistent store. As part of our development effort we have also built on top of PSE, a schema evolution facility, **Schema Evolution Primitive Manager**, the **Schema Repository** and **OQL Query Engine** which are not directly a part of the SERF system. Figure 1 gives the overall architecture of our system.

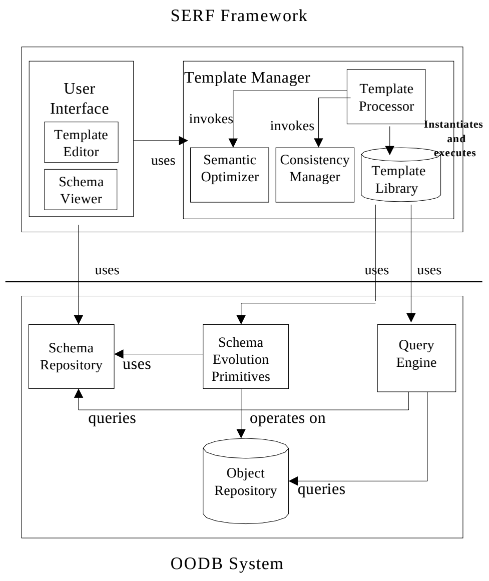

Figure 1: Architecture of the SERF Framework

2.2 Tools for the SERF System

Template Processor. The Template Processor is an interface between the user and the SERF framework for the execution of a template. Figure 2 shows the steps performed by the Template Processor for the execution of a SERF template. The template processing begins with the user supplying the input parameters. These parameters are a particular `Class` or `Property` in the application schema to which the user wants to apply the SERF template transformation. A *type-check* ensures that the types of the parameters match and they exist in the system as well as the correct number of parameters are supplied by the user. This is followed by a *bind-check* which checks the existence of these actual parameters in the schema on which they are being applied by accessing the Schema Repository. The SERF template is instantiated using these parameters by replacing each variable with its bound parameter after all the checks are completed successfully. The instantiated SERF template now corresponds to pure OQL statements, i.e., we now call it an OQL transformation. The OQL Query Engine provides an interface for the syntax-checking, the *parsing* and the *execution* of the OQL transformation.

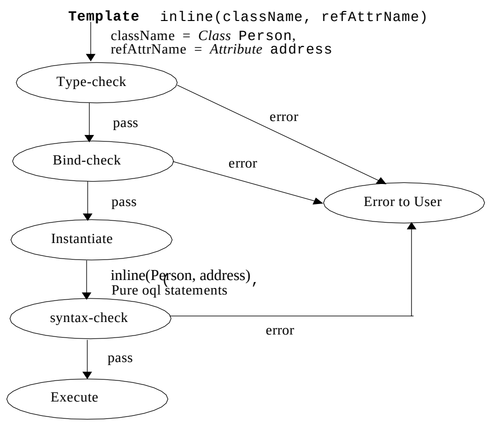

Figure 2: Steps for the Execution of a Template.

Template Library. SERF templates for a particular domain are collected in a Template Library. Search of a template in this library is supported via simple keyword search on all stored parameters of the template such as the input and output parameters, the name, the *contract* [1], and the description. We are currently in the process of defining *inheritance* and other semantic relationships between templates to facilitate a multi-level organization of the template library.

Consistency Manager. The current version of the Consistency Manager deals with both the *invariant consistency* as specified by the invariants of the object model as well as with *contractual consistency*, i.e., the consistency as specified by the individual *contracts* of a template. As part of the consistency manager, we have developed a language for the specification of *contracts*, i.e., the constraints that must be satified by the input schema given by the template and also the constraints that must be satisfied at termination time. The consistency manager uses these *contracts* for the detection of erroneous templates and provides a framework for the management of violations as well as for verification.

Semantic Optimizer. The Optimizer is a tool for optimizing SERF templates in terms of their execution times. The semantic optimizer uses a query graph as its internal representation. The optimization is based on a semantic analysis of the data dependencies in this graph, as well as heuristics of pairwise eliminating, canceling or merging schema evolution primitives [CNR99]. We have focused in particular on the optimization of evolution methods.

Application specific tool: XML Mapper. In the context of Re-WEB, a system for the generation and

[1] This is for the specification of consistency constraints as required by the consistency manager.

re-structuring of the web based on the SERF system, we have also developed the XML Mapper, a tool that allows us to produce `xml` files using the structure and the objects present in the object database and vice versa. The XML Mapper uses the web semantics we defined for the ODMG object model [CRCK98].

2.3 Highlights of the Demo

We will demonstrate the latest version of our OQL-SERF system using an example to walk through and show its core functionality in terms of the steps of execution for a transformation (Figure 2) using our library of templates [2]. We will also demonstrate the re-usability, flexibility and the extensibility of our SERF framework and its applicability to the web re-structuring via the following:

- Re-usability. We will demonstrate how a transformation can be generalized to a template as above and re-used for other meta objects in the schema.
- Flexibility. We will demonstrate the ease with which a user can change the semantics of a pre-existing template.
- Extensibility. We will demonstrate the extensibility of our system by showing how a template can be embedded in another template to make more complex transformations.
- Applicability Beyond Schema Evolution. We demonstrate how Re-WEB can be used to generate customized web pages. SERF makes restructuring of web sites a very manageable task.

References

[BKKK87] J. Banerjee, W. Kim, H. J. Kim, and H. F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *SIGMOD*, pages 311–322, 1987.

[Bré96] P. Bréche. Advanced Primitives for Changing Schemas of Object Databases. In *Conference on Advanced Information Systems Engineering*, pages 476–495, 1996.

[CJR98a] K.T. Claypool, J. Jin, and E.A. Rundensteiner. OQL_SERF: An ODMG Implementation of the Template-Based Schema Evolution Framework. In *Centre for Advanced Studies Conference*, pages 108–122, November 1998.

[CJR98b] K.T. Claypool, J. Jin, and E.A. Rundensteiner. SERF: Schema Evolution through an Extensible, Re-usable and Flexible Framework. In *Int. Conf. on Information and Knowledge Management*, pages 314–321, November 1998.

[CNR99] K.T. Claypool, C. Natarajan, and E.A. Rundensteiner. CHOP: An Optimizer for Schema Evolution Sequences. Technical Report WPI-CS-TR-99-06, Worcester Polytechnic Institute, February 1999.

[CR99] K.T. Claypool and E.A. Rundensteiner. SERF: Transforming your Database. In *IEEE Bulletin - Special Issue on Database Tranformation Technology*, 1999. to appear.

[CRCK98] K.T. Claypool, E.A. Rundensteiner, L. Chen, and B. Kothari. Re-usable ODMG-based Templates for Web View Generation and Restructuring. In *CIKM'98 Workshop on Web Information and Data Management (WIDM'98), Washington, D.C., Nov.6*, 1998.

[Ler96] B.S. Lerner. A Model for Compound Type Changes Encountered in Schema Evolution. Technical Report UM-CS-96-044, University of Massachusetts, Amherst, Computer Science Department, 1996.

[Obj93] Object Design Inc. *ObjectStore - User Guide: DML. ObjectStore Release 3.0 for UNIX Systems.* Object Design Inc., December 1993.

[Tec94] O_2 Technology. *O_2 Reference Manual, Version 4.5, Release November 1994.* O_2 Technology, Versailles, France, November 1994.

[2]This library of templates has been formulated by the case study of the different transformations that we have found in research literature.