

POSTER PAPER

Mediation Framework Modeling and Verification by SAM

Li Yang, Raimund K. Ege
School of Computer Science
Florida International University
Miami, FL 33199, USA
lyang03|ege@cs.fiu.edu

Huiqun Yu
Department of Computer Science
East China University of Sci & Tech
Shanghai 200237, China
yhq@ecust.edu.cn

ABSTRACT

The architecture-level design of mediation systems requires rigorous modeling and analysis techniques to assure the correctness the behaviors of the architecture. The Software Architecture Model (SAM) is a formal systematic software architecture specification and analysis methodology that is able to define and analyze different system aspects using different formalisms to improve understandability and reduce complexity. This paper proposes an adaptive mediation framework that provides an easily extensible, decentralized environment for sharing data from heterogeneous databases. We demonstrate how to apply SAM to specify the mediation architecture and analyze the temporal properties of the proposed architecture.

Categories and Subject Descriptors

D.2.4 [Software Engineering]: Software/Program Verification—*Formal methods*; D.2.12 [Software Engineering]: Software Architectures—*Domain-specific architectures*; H.4 [Information Systems Applications]: Miscellaneous

General Terms

Design, Reliability, Verification

Keywords

Mediation architecture, model, specification, SAM

1. INTRODUCTION

The trend in modern information systems is to access heterogeneous sources to serve a variety of client types.

Mediators are typically employed in a situation where the client data model does not coincide with the data model of the potential data sources. A mediator provides a mapping of complex models to enable interoperability between clients and sources. With the promise of integrating data with

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'05 March 13-17, 2005, Santa Fe, New Mexico, USA
Copyright 2005 ACM 1-58113-964-0/05/0003 ...\$5.00.

rich semantics, we propose a mediator architecture[1] for the identification of the physical feed source based on the availability of sources and factors derived from the client and network capabilities.

2. A THREE-LAYERED ARCHITECTURE

The proposed three-layered mediator architecture provides a flexible and general way to retrieve and update information in heterogeneous databases. The framework features three layers: presence, integration and homogenization/connector. The upper level is the presence layer. The presence layer is responsible to translate the heterogeneous request from user to XML format, extract the data type of request represented by XML schema, and translate the response from the XML format into the original user request format. The middle integration layer resolves the schema differences between the user needs and the source availability by schema mapping. The entities in the integration layer are the “Mediator_composers” who are able to decompose the schema if necessary and locate the destination data source for a specific schema by looking up its built-in distributed hash table (DHT). The bottom level homogenization layer contains “Mediator_connectors” that resides on top of actual data sources, and maps the data source schemas to XML schemas.

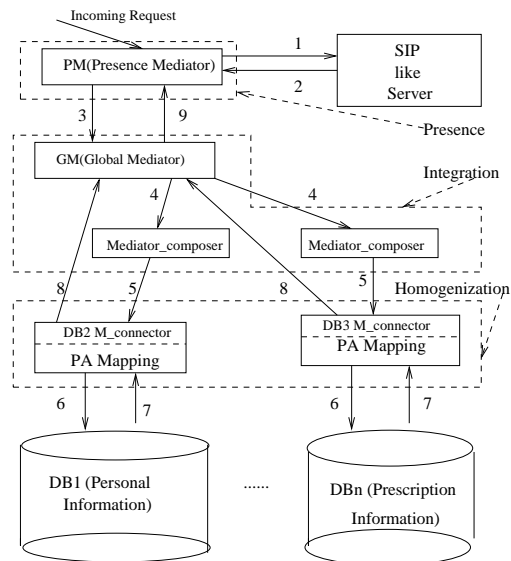


Figure 1: A three-layered architecture of mediation

Interaction among components of the mediation architecture

(as in Figure 1) is outlined as follows.

1. The presence mediator sends request to SIP like Server (*SS*).
2. The *SS* sends the global mediator ID to presence mediator after the execution of global mediator election.
3. The presence mediator sends the request to the global mediator.
4. The global mediator tracks down the connector ID for every schema piece among the Mediator group on the basis of DHT. The data type of the request is recorded in the global mediator and the global mediator maintain the data type for the request in case the Mediator_composers on the lookup path decompose the request schema.
5. Mediator_connector receives request from Mediator_composer.
6. The Mediator_connector retrieves the data from data sources or update the databases upon the request from Mediator_composers.
7. The databases return the queried data or update status to connector.
8. The global mediator integrates multiple data streams obtained from Mediator_connector by global mediator type it maintains or receives the update status.
9. The global mediator responses to the presence mediator by integrated XML object if the request is query or by success/failure flag if the request is update.

3. A FORMAL MODEL OF MEDIATION ARCHITECTURES

In SAM, a software architecture is defined by a hierarchical set of compositions in which each composition consists of a set of components, a set of connectors and a set of constraints to be satisfied by the interacting components. Basically, behaviors of components and connectors are modeled by Petri nets, while their properties (or constraints) are specified by temporal logic formulas [2]. The interfaces of components and connectors are *ports*, which are *places* in Petri nets.

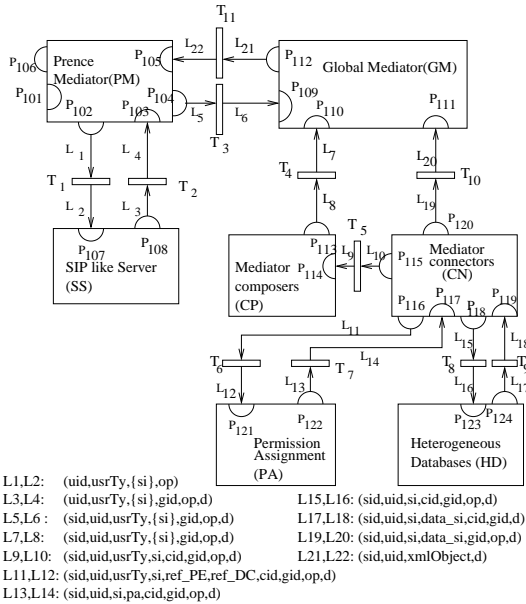


Figure 2: A mediation architecture model in SAM

Table 1. Variables in Figure 2

Variable	Description
uid	user id
usrTy	user Type
si	schema i
mid	mediator_composer id
cid	mediator_connector id
gid	global mediator id
ref_DC	object reference of a decision combinator
ref_PE	object reference of of policy evaluator
pa	permissin assignment, pa is either read, or write or undefined
data_si	data stream for specific data
xmlObject	xml Object returned to presence mediator
op	operation, op is either query or update
d	operation result, d is either success or fail

Using SAM, a formal mediation architecture model is established as in Figure 2. The model consists of seven system components (*PM, GM, SS, CP, CN, PA, HD*) as well as eleven connectors (T_1 to T_{11}). Each element (either a component or a connector) C is a PrT net, which is associated with a property specification C_S in temporal logic. The component PM is derived from presence mediator part in the three-layered architecture of mediation (as in Figure 1). Four ports P_{102} (output), P_{104} (output), P_{103} (input), P_{105} (input) are obtained as a direct result of message 1, 3, 2 and 9. The variables on the arc labels are derived from the message parameters. Connector T_1 with input P_{102} and output P_{107} derived from the message 1 passing in Figure 1. Other components and connectors are obtained likewise.

By [2], the composition-level property specification is obtained by conjoining the property specifications of all components and connectors:

DES: $PMs \wedge T_1s \wedge SSs \wedge T_2s \wedge T_3s \wedge GMs \wedge T_4s \wedge CPs \wedge T_5s \wedge CNs \wedge T_6s \wedge PAs \wedge T_7s \wedge T_8s \wedge HDs \wedge T_9s \wedge T_{10}s \wedge T_{11}s$
One overall mediation system requirement is that every user request must be eventually processed, which can be formulated as follows:

REQ: $\forall(uid, req, op). \exists(response, d).$

$\Box(P_{101}(uid, req, op) \Rightarrow \Diamond P_{106}(response, d))$

Using the verification mechanism in SAM, we have shown that conjunction of the lower-level property specifications implies that higher level property specification, i.e. $DES \vdash REQ$.

Acknowledgments: The authors are grateful to the anonymous reviewers for helpful comments. This work was partially supported by the NSF of the USA under grant HRD-0317692, and by the NSF of China under grant No. 60473055.

4. REFERENCES

- [1] Raimund K. Ege, Li Yang, Qasem Kharm, and Xudong Ni. Three-layered mediator architecture based on DHT. In *International Symposium on Parallel Architecture, Algorithm and Networks (I-SPAN)*, Hong Kong, China, May 2004. IEEE press.
- [2] X. He and Y. Deng. A framework for developing and analyzing software architecture specifications in SAM. *The Computer Journal*, 45(1):111–128, 2002.