5.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-01-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Department of Defense, Washington Headquarters Services Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
**PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

**1. REPORT DATE** *(DD-MM-YYYY)*
2003

**2. REPORT TYPE**
Conference Proceedings

**3. DATES COVERED** *(From - To)*

**4. TITLE AND SUBTITLE**
A Case Study in Object Oriented Modeling, Architecting, and Designing an Enterprise Monitoring Application

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHORS**
Sushil J. Louis*
John McDonnell, Doan Hohmeyer, Lisa Heinselman, Andrew Walker**

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
*Genetic Algorithms Systems Lab
University of Nevada
Reno, NV 89557

**SSC San Diego
53560 Hull Street
San Diego, CA 92152–5001

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**
This is the work of the United States Government and therefore is not copyrighted. This work may be copied and disseminated without restriction. Many SSC San Diego public release documents are available in electronic format at:
http://www.spawar.navy.mil/sti/publications/pubs/index.html

**14. ABSTRACT**
We describe the use of object oriented techniques for the specification, architecting, and design of an enterprise monitoring application. Monitoring applications provide situational awareness and monitor aspects of an enterprise's activities. Drawing from a wide variety of static and dynamic data sources, they typically allow a user to specify items of interest, and drill down to obtain real-time or near-real time information on such items. Our paper describes the use of standard UML for modeling and the issues in architecting and designing our J2EE framework based application.

Published in *Proceedings of the International Conference on Software Engineering Research and Practice*. Volume II, 2003.

**15. SUBJECT TERMS**
enterprise monitoring

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | John McDonnell, Code 246214 |
| | | | | | 19B. TELEPHONE NUMBER *(Include area code)* |
| U | U | U | UU | 6 | (619) 553-5762 |

**Standard Form 298** (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

# A Case Study in Object Oriented Modeling, Architecting, and Designing an Enterprise Monitoring Application

**Sushil J. Louis**

Genetic Algorithms Systems Lab
University Of Nevada, Reno
sushil@cs.unr.edu

**John McDonnell, Doan Hohmeyer, Lisa Heinselman, Andrew Walker**

SPAWAR Systems Center
San Diego, CA
mcdonn@spawar.navy.mil

**Abstract**

*We describe the use of object oriented techniques for the specification, architecting, and design of an enterprise monitoring application. Monitoring applications provide situational awareness and monitor aspects of an enterprise's activities. Drawing from a wide variety of static and dynamic data sources, they typically allow a user to specify items of interest, and drill down to obtain real-time or near-real time information on such items. Our paper describes the use of standard UML for modeling and the issues in architecting and designing our J2EE framework based application.*

## 1. Introduction

Enterprise monitoring also known as business process monitoring is a subset of business process management applications. The goal is to allow members of an enterprise to obtain a view of event flow within the enterprise and to and from external entities. This view is usually tailored to the particular member's interests and needs. Many issues impact the modeling and design of business process monitoring systems, in particular, we dealt with

- Multiple static data bases
- Multiple dynamic data sources
- Multiple information update rates, and
- Asynchronous communication

An important trend in current enterprise application design is to use enterprise application frameworks such as .Net and J2EE for developing scalable, reliable, and secure enterprise applications. We chose the J2EE framework for these reasons and, in particular, for the J2EE framework's open standards and open source availability of J2EE application servers.

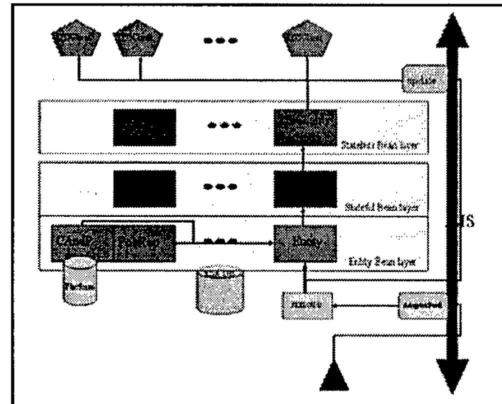

**Figure 1: Mapping SIREN's architecture onto the J2EE framework.**

Figure 1 depicts how our architecture maps into the three-tier J2EE application framework. Information flows from the bottom of the diagram (data producers) towards the top (data consumers) through entity, stateful session, and stateless session enterprise java beans. The triangle adaptor interfaces to a data source and asynchronously updates a Java Message Service (JMS) topic. The entity bean layer

20060425020

then persists this updated information in a database and relates this data with existing tables in the database. The stateful session bean layer provides a façade to the database for upper level clients. The stateless session bean layer provides a façade to presentation layer clients. The system can scale to handle hundreds of messages per second while providing real-time updates to dozens of clients.

Rather than delving into the details of a proprietary system, the main focus of this work is on how UML supports the modeling of our system within the J2EE framework [1, 2]. Many object oriented modeling and design methods appeared in the 90s and Booch, Rumbaugh, and Jacobson unified the various approaches into a standardized notation for design and analysis – the Unified Modeling Language [3]. UML provides a rich set of modeling tools to describe dynamic and static aspects of the system being designed. This case study describes our experience in using UML to model the SIREN situational awareness tool. We followed a spiral development model and made extensive use of UML for modeling and for design.

The next section describes the J2EE framework, the SIREN tool, and how we mapped SIREN requirements to the J2EE framework. The section uses UML diagrams to communicate different aspects of the system and our experience in using UML for requirements elicitation and feature specification. The last section concludes with the lessons learned from this project.

## 2. SIREN and the J2EE framework

The Java2 Platform, Enterprise Edition (J2EE) defines a standard for developing multitier enterprise applications. J2EE simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behavior automatically, without complex programming. Multitier scalable, reliable, and portable applications are hard to architect because they require bringing together a variety of skill-sets and resources, legacy data, and legacy code. In today's heterogeneous environment, enterprise applications have to integrate services from a variety of vendors with a diverse set of application models and other standards. Industry experience shows that integrating these resources can take up to 50% of application development time [5].

Security, messaging, transaction monitoring, and integration with legacy enterprise applications are all simplified within this enterprise application development framework. By providing this enterprise infrastructure, the framework allows for quick development, deployment, portability, and scalability.

SIREN is an enterprise state monitoring application that siphons data from a number of databases and data feeds, persists this data in its database, and feeds this organized data to client users. Clients specify the information that they would like to see. We use the application server's asynchronous messaging system to move data around this distributed system. Data sources are distributed over multiple servers on a network; each data source type has different control and data interfaces. Asynchronous communications among the components of this system make the system more robust.

### 2.1 Attaching an Adaptor

We use "adaptors" to connect to these different data sources and provide the server a façade to the underlying data source.

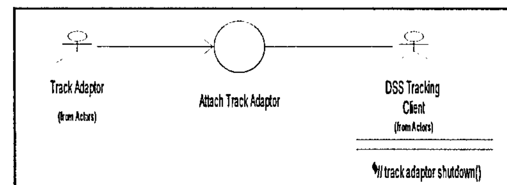


**Figure 2: Use case for attaching a track adaptor.**

Figure 2 depicts the Attach-Adaptor use case and Figure 3 shows the corresponding activity diagram.

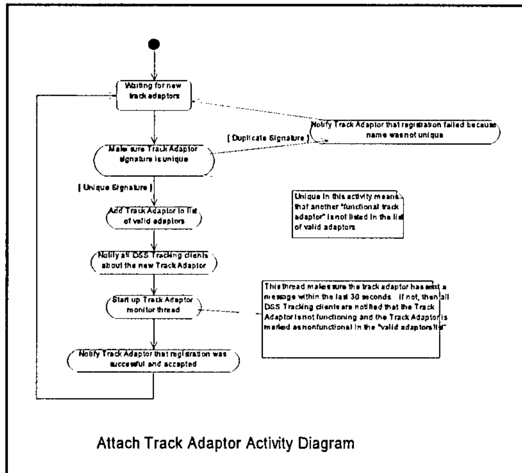


**Figure 3: Activity diagram for attaching a track adaptor.**

The adaptor sends the server a request-to-attach message on the message bus. The server authenticates the adaptor, sets up a communication channel and notifies all clients of the new adaptor. This sequence of events is best denoted in the sequence diagram in Figure 4.

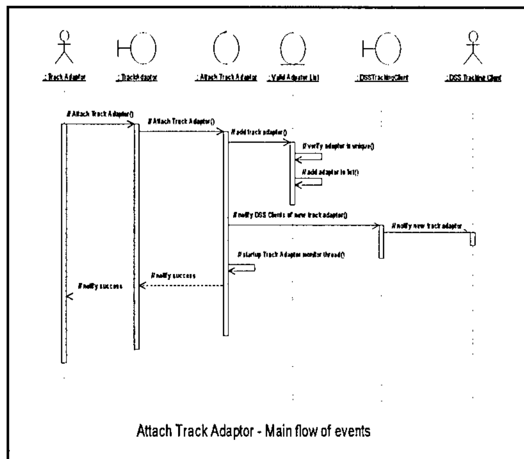


**Figure 4: Sequence diagram for attaching a track adaptor.**

Once the adaptor has registered with the server, it obtains a JMS topic and begins sending information updates to that topic. These messages trigger updates to the underlying database and corresponding updates to client subscribed topics. The class diagram in Figure 5 shows the adaptor specific interfaces in our system. Note that the *attachTrackAdaptor* method corresponds to our attach track adaptor use case. Once an adaptor signals its existence to the server, the server calls the *attachTrackAdaptor* method to register the adaptor and begins accepting messages from the adaptor. We will describe the shutdown track adaptor use case later, the next section deals with the server use case model.

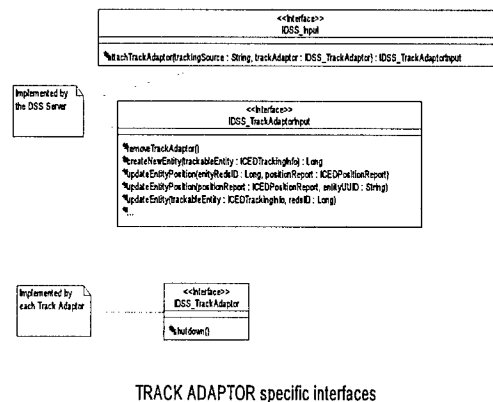


**Figure 5: Interfaces for the track adaptor**

## 2.2 Server Model

The server use case model below (Figure 6) depicts the overall operation of the server with respect to adaptor interactions.

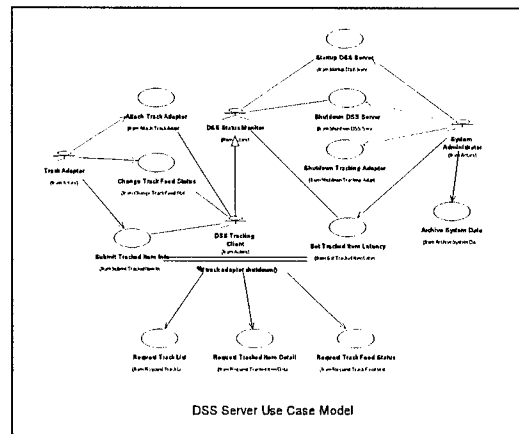


**Figure 6: The Server Adaptor interaction use case model.**

The server also relates a variety of static information to data from dynamic data sources. For example, entities that a client is interested in may have properties associated with them. A table of entity types and their properties thus needs to be related to entities being monitored. The application server provides support for modeling these database table relations in a database-vendor neutral manner. Clients clicking on an entity can therefore obtain both static and dynamic information on the entity.

The class diagram for an entity class is given below (Figure 7). Most *getXXX* methods correspond to getting the contents of a column in the selected row of the entity database table. Some *getXXX* methods correspond to relations and return rows in other related tables in the database. The entity class therefore serves as the superclass for all objects of interest in the system. One interesting design debate arose over whether to transport the relatively heavyweight entity object over the network to all clients; our eventual solution was to transport only those portions of the entity requested by the client. The entity's location and type were always transported over the network while other entity data was made available as requested.

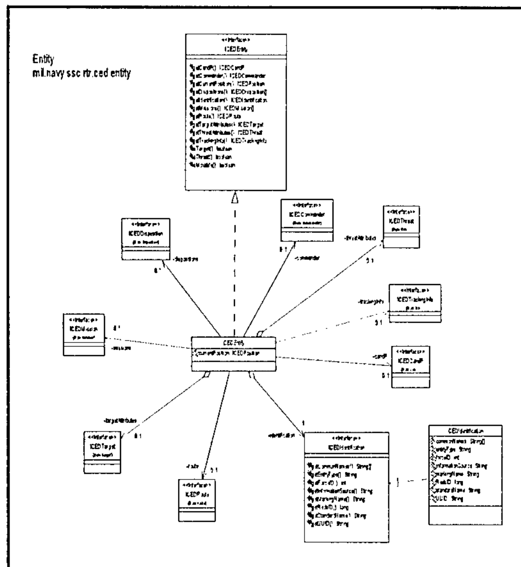


**Figure 7: Entity class diagram.**

All entities implement several interfaces relating to common properties. We impose a common implementation for entity identification, position, capabilities, and other properties through the use of interfaces and inheritance as shown in Figure 7.

An entity object is stored in the server database as a collection of rows in multiple tables. Essentially we had to create a set of database tables and relations that distributed an entity's contents over these tables and relations. Commercial database vendors have spent years optimizing their product's performance and scalability. Mapping the complex entity object to a set of relational tables, allows our server database to optimize its storage and indexing so that database access is fast, transactional, and reliable. In contrast, if we had stored an entity object as a blob, a serialized object, we would not be taking advantage of commercial database optimizations. The **object-relational** mapping to map an entity to a set of relational database tables was done manually and was an important component of the final design. This up-front work in establishing a mapping allows scaling the system to handle high message (input) and client (output) loads.

### 2.3 Shutting down an adaptor

Having delved deep into the system, we step back and describe the shutting down of an adaptor. Figure 8 shows that shutdown track adaptor use case. Note that clients need to be informed of adaptor shutdowns initiated by the system administrator.

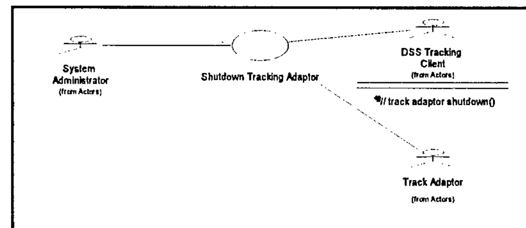


**Figure 8: Shutdown track adaptor use case.**

The activity diagram in Figure 9 helps flesh out the use case with the flow of events that need to be implemented.

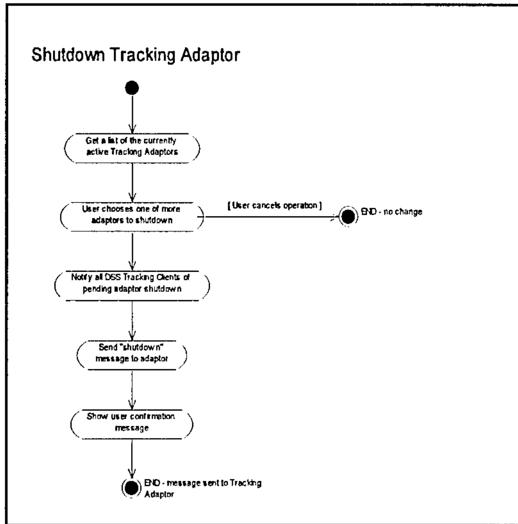


**Figure 9: Shutdown tracking adaptor activity diagram**

The sequence diagram in Figure 10 further details the process and provides almost enough detail for straightforward implementation.

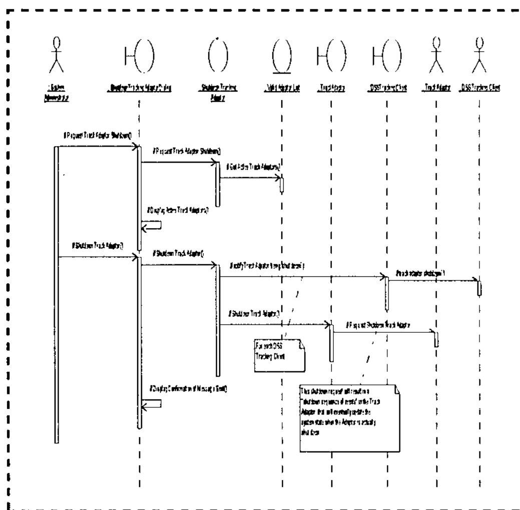

**Figure 10: Shutdown track adaptor sequence diagram.**

More importantly, we have found that these three sets of UML diagrams, use-case, activity, and sequence diagrams, allow us to communicate clearly with end users and domain experts and confirm our understanding of the system's requirements.

## 3. System Implementation and Results

The UML modeling was used to obtain and confirm the design team's understanding of SIREN's requirements. However, while more detailed models were being acquired from end-users and domain experts, the architecture team used its broad knowledge of requirements to design and implement an end-to-end skeleton of the system. Skeleton development had three objectives:

- Learning about the application server and the application server interfaces and services
- Testing the quality of our design through a concrete implementation of an end-to-end system
- Providing a template for a full implementation

We had a heterogeneous development environment consisting of linux and windows machines. During testing the application server and database server ran on different windows machines with clients and adaptors distributed among windows and linux boxes. We used BEA's weblogic application server, Java, and MS SQLserver[6, 7, 8]. Although we found that our business logic was portable, porting to a different database (Oracle) was quite difficult because of differences in support for some primitive data types.

We successfully tested the skeleton in this heterogeneous environment and are testing a complete implementation.

## 4. Conclusions

This paper described a case study in designing and implementing a large enterprise monitoring system. We found UML helped in acquiring and elucidating system specifications – both in dialogs with end users and domain experts and for

programmers in refining and implementing the system. Developing an end-to-end architectural skeleton helped us gain a good understanding of implementation issues and get a feel for the system's scalability and reliability. Although we also expected the skeleton to be used as a template for further development and fleshing out of the system, we found that going through skeleton code was not as easy as going through UML and developers preferred to design and implement with little reference to the skeleton. Our next spiral development objective is to add functionality, test scalability, and bring the system up to CMM Level 3 [9].

**5. Acknowledgements**

This material is based upon work supported by the Office of Naval Research under contract number N00014-03-1-0104.

**5. References**

1. Grady Booch, Ivar Jacobson and James Rumbaugh. *The Unified Modeling Language User Guide*. Addison Wesley, 1998.

2. Sushil J. Louis, John McDonnell, and Nicholas Gizzi. "Dynamic Strike Force Asset Allocation Using Genetic Search and Case-Based Reasoning." In *Proceedings of SCI-02,* pages 423 - 428, July 2002.

3. John McDonnell, Nicholas Gizzi, and Sushil J. Louis. "Strike Force Asset Allocation Using Genetic Search." In Hamid Arabnia et all, editor, *Proceedings of the 2002 International Conference on AI, CSREA Press,* pages 897 - 901, June 2002.

4. Richard Monson-Haefel. *Enterprise JavaBeans (3rd Edition)*. O'Reilly and Associates, 2001.

5. Sun's J2EE website at http://java.sun.com/j2ee/.

6. BEA's website at http://www.bea.com

7. Sun's Java website at http://java.sun.com

8. Microsoft's SQLServer website at http://www.microsoft.com/sql/default.asp

9. The Capability Maturity Model webpage at http://www.sei.cmu.edu/cmm/cmm.sum.html