Special Report

SEI-89-SR-14

Conformance Criteria for the SAME Approach to Binding Ada Programs to SQL

James W. Moore

August 1989

Special Report

SEI-89-SR-14
August 1989

Conformance Criteria for the SAME Approach to Binding Ada Programs to SQL

James W. Moore

IBM Systems Integration Division
in conjunction with
the Binding of Ada and SQL Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Conformance Criteria for the SAME Approach to Binding Ada Programs to SQL

Abstract: The structured query language (SQL) Ada Module Extensions (SAME) form a method for the design and construction of Ada database applications. The method is explained in a companion document: "Guidelines for the Use of the SAME," [2]. In order to enable the method to be referenced in requests for proposals (RFPs) and development contracts, there must be some method to determine if a given software design and implementation conform to the SAME guidelines. Such conformance criteria are contained in this report.

1. Background

The American National Standards Institute (ANSI) has provided a standard binding from the Ada programming language to the structured query language (SQL) database module language in [1] [3]. This standard treats the binding to Ada in a manner similar to the binding to other programming languages and does not exploit the specific strengths of the Ada programming language. The SQL-Ada Module Extensions Design Committee (SAME-DC) has developed an approach, the "SAME approach," for building an interface between an Ada program and the ANSI binding in a manner that exploits the particular strengths of the Ada language. This approach is described in [2]. Specific Ada code packages that may be used to implement a particular instance of the SAME approach are listed in [2]. The SAME-DC recognizes that it may be desirable to describe the SAME approach in a manner suitable for subsequent standardization. This paper is an initial attempt at achieving that goal. At the current state of description, proper usage of the SAME approach involves engineering judgment by the developer. So, the decision as to whether a particular implementation conforms to the SAME approach also involves some judgment. Therefore, the current conformance criteria are intended for use in the sort of peer review process that is oriented toward the evaluation of software design and implementation for conformance to sound software engineering practice.

2. Approach

The general approach for describing conformance to the SAME method is to provide qualitative statements regarding the role and function of the Abstract Module that must be present in a conforming binding. Various of the SAME support services would be useful in the development of an abstract module, and their usage is described. It must be understood, however, that these code specifications represent only an example of a way to achieve conformance and that, in a particular instance, other ways to achieve conformance may be desirable. For the purposes of this document, the SAME support services are defined as the collection of some of the Ada package specifications listed in [2] with the exception of the contents of their private sections. The selected Ada package specifications are the following:

SQL_Standard (part of ANSI binding to Ada, [3])

SQL_System
SQL_Exceptions
SQL_Int_Pkg
SQL_Real_Pkg
SQL_Char_Pkg
SQL_Database_Error_Pkg
SQL_Communications_Pkg
SQL_Boolean_Pkg
SQL_Smallint_Pkg
SQL_Double_Precision_Pkg
SQL_Enumeration_Pkg

3. Conformance Criteria

An interface from an Ada application program to an ANSI SQL database is viewed as conforming to the SAME approach if it satisfies the criteria below:

1. A set of Ada packages, called the **Abstract Module** (AM), shall provide the exclusive means of access of the application program to the database. The **Abstract Interface** (AI) is the Ada specifications for those portions of the AM that are visible to the application program. The Abstract Module shall provide no function other than that necessary to provide the application program access to the database.

2. A one-to-one correspondence shall exist between the procedures of the SQL Module and the Ada procedures in the AI. (In some implementations, the SQL Module may not actually exist except as a formalism of the ANSI standard binding.)

3. **Parameters**. There are two cases that shall be termed "status parameters" and "data parameters":

    a. **Status parameter**: Each AI procedure may have a parameter of mode out used to return information regarding the value of the SQLCODE parameter in the corresponding procedure of the SQL Module. This parameter shall be of an enumeration type. In the absence of such a parameter, any non-zero SQLCODE value returned from the corresponding SQL Module procedure shall cause the raising of an exception in the execution of the corresponding AI procedure. If the status parameter is coded, a partial function shall exist from the possible values of SQLCODE to the possible values of the status parameter. The value of the status parameter shall be set in accordance with the partial function except that the exception shall be raised whenever SQLCODE is set to an unmapped non-zero value. The exception shall have the unique meaning that a non-zero, unmapped SQLCODE value has been returned by the SQL module.

    In the SAME support services, the exception described by the preceding paragraph is named SQL_COMMUNICATIONS_PKG.SQL_DATABASE_ERROR.

    b. **Data parameters**: All parameters of the SQL Module except SQLCODE are referred to as "data parameters." All data parameters of an SQL Module procedure shall be placed in correspondence with parameters of the corresponding AI procedure. In those cases where the SQL Module procedure is a fetch statement or an insert-values statement, all of the data parameters of the SQL procedure shall be placed in correspondence with the components of a *row record* parameter of the corresponding AI procedure. In the case where the SQL Module procedure is a select statement, the data parameters representing the table row shall likewise be placed in correspondence with the components of a row record parameter of the corresponding

AI procedure. In all other cases, each data parameter of the SQL Module procedure shall be placed in correspondence with an individual parameter of the AI procedure.

i. **Row records**: Each row record parameter appearing at the AI shall have the type of an Ada record. The components of the record shall be placed in correspondence with the data parameters of the SQL Module procedure. Except for the characteristic of being components of a record, the components shall satisfy the same requirements described below for individual parameters.

ii. **Individual parameters**: The individual data parameters of an AI procedure shall be placed in a correspondence with the data parameters of the corresponding SQL Module procedure. This correspondence shall be one-to-one except that an indicator parameter of an SQL module and its companion parameter shall be viewed as a single aggregate parameter for the purpose of making this correspondence. The Ada type of the data parameters of the AI procedures shall conform to the characteristics described in the following paragraphs:

1. **Possibly null types**: AI data parameters that correspond to an SQL Module Language aggregate parameter, i.e., an SQL indicator and its companion parameter, must be of a type that is capable of representing a null value in a manner that can be distinguished from all of the values of the companion parameter. The user shall select a data type that makes it impossible for the application program to use a null value as if it were non-null without the raising of an exception.

   The SAME support services provide appropriate types for the representation of possibly null values. The following table lists these types in correspondence with the types of the ANSI SQL binding that are declared in package SQL_Standard:

```
Char                SQL_Char_Pkg.SQL_Char
Smallint            SQL_Smallint_Pkg.SQL_Smallint
Int                 SQL_Int_Pkg.SQL_Int
Real                SQL_Real_Pkg.SQL_Real
Double_Precision    SQL_Double_Precision_Pkg.
                        SQL_Double_Precision
```

2. **Non-null types**: AI data parameters that are not in correspondence with an SQL indicator may have a data type that is not capable of representing a distinguishable null value.

   The SAME Support Services provide appropriate types

for use in this circumstance. The following table lists these types in correspondence with the type of the ANSI SQL binding that are declared in package SQL_Standard:

```
Char                SQL_Char_Pkg.SQL_Char_Not_Null
Smallint            SQL_Smallint_Pkg.
                        SQL_Smallint_Not_Null
Int                 SQL_Int_Pkg.SQL_Int_Not_Null
Real                SQL_Real_Pkg.SQL_Real_Not_Null
Double_Precision    SQL_Double_Precision_Pkg.
                     SQL_Double_Precision_Not_Null
```

Acknowledgements

This initial draft was based upon material formulated by the SAME Design Committee (SAME-DC) at its meeting of May 15-17, 1989, in Pittsburgh, Pennsylvania. This meeting was chaired by Marc Graham of the SEI and attended by:

Greg Zelesnik, SEI
Pat Timpanaro, Compass
Judith Richardson, U.S. Army
Rowan Maclaren, Compass
Kurt Wallnau, Unisys
Joan Krier, Intermetrics
Eugene Vasilescu, Grumman
Tom Wheeler, U.S. Army
Dudley Rodericks, U.S. Army
Jim Moore, IBM

Organizations are listed for identification purposes only. This document should not be taken as reflecting the opinions of any company or organization other than the SAME-DC.

References

[1] *Database Language SQL.*
American National Standards Institute, ANSI X3.135-1986.
also published as International Standards Organization ISO 9075-1987.

[2] Graham, Marc.
Guidelines for the Use of the SAME.
Technical Report SEI/CMU-89-TR-16, Software Engineering Institute, June, 1989.

[3] *Database Language Embedded SQL.*
ANS X3.168.1989.

Table of Contents