ISO/IEC JTC 1/SC 32 N 0322

Date: 1999-05-27

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)

Administered by Pacific Northwest National Laboratory on behalf of ANSI

DOCUMENT TYPE	Working Draft Text (for information or comment)
TITLE	WD Metadata Query Service An Object Technology Extension to the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes.
SOURCE	SC 32/WG 2
PROJECT NUMBER	
STATUS	For Submission to SC 32 for approval with the NP
REFERENCES	
ACTION ID.	COM
REQUESTED ACTION	For Submission to SC 32 for approval with the NP
DUE DATE	
Number of Pages	34
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 901 D Street, SW., Suite 900, Washington, DC, 20024-2115, United States of America

Telephone: +1 703 575 2114; Facsimile; +1 703 681 9180; E-mail: MannD@battelle.org

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

ISO/JTC 1/SC 32 N0322

1999-04-20

ISO/WD

ISO/IEC JTC 1/SC 32/WG 2

ANSI

Metadata Query Service

An Object Technology Extension

to the

ISO/IEC 11179 Specification and Standardization of Data Elements,

Part 3, Basic Attributes.

International standard
Draft Version
English

Contents

Foreword

ISO (the International Organization for Standardization) is a world-wide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.
Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.
ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for world-wide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.
In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Supporting organizations

Organzsation Represented	***Name of Representative***
3M Health Care	Thomas C. Culpepper
	Harold R. Solbrig (Alt.)
Australian Institute of Health & Welfare	Joe Christensen
Battelle Memorial Institute	Douglas D. Mann
Department of Defense - Health Affairs	Glenn Sperle
Health Care Financing Administration	Robert Mayes
Lawrence Berkeley National Laboratory	Frank Olken
	John McCarthy (Alt.)
Mitretek Systems	Andrew Schoka
National Institute of Standards and Technology	Judith J. Newton
Science Applications International Corporation	Phong Xuan Ngo
U.S. Bureau of the Census	Daniel W. Gillman
U.S. Environmental Protection Agency	Bruce E. Bargmeyer
	Larry Fitzwater (Alt.)
U.S. Geological Survey	Eliot Christian

Individual Acknowledgements

Henry Heffernan
William H. Kenworthey, Jr.

History

Initial Draft	**1999 April 20**
Informative	*Normative*
The initial draft of the Metadata Query Service to the *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes*. This draft is proposed for discussion at the SC 32/WG 2 meeting to be held 1999 May 24-28, in Matsua, Japan.	Five interfaces will be defined to correspond to the regions of the *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes*. In this initial draft the Stewardship interface has been completed. Classification, DataElement, DataElementConcept and ValueDomain interfaces will be designed next.

Initial Draft	**1999 May 11**
Informative	*Normative*
Grammatical changes. Clarification.	

Relationship to other standards

Because MQS is an object extension to the *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes* it is inherently tied to that standard. *ISO/IEC 11179* defines the attributes necessary for a metadata registry and the MQS defines the behaviors necessary for accessing information contained in an *ISO/IEC 11179* complaint metadata registry.

We anticipate that much of the work done in the Lexicon Query Service (LQS)[1] will be used as a pattern for some of the Metadata Query Service. The Lexicon "L" portion of the LQS deals with terminology and the underlining terminology model which will be reviewed and coordinated with *the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes* by those involved with designing and developing this specification. The Query Service "QS" portion of the LQS deals with access information in a well-defined way. We anticipate that the "QS" portion will be used as the pattern for accessing metadata from a data registry.

Normative/informative annexes

The following annexes are informative:

Annex A - Use Cases.

[1] Lexicon Query Service is a read-only specification for accessing the content of medical terminology systems. It is an adopted international standard by the Object Management Group (OMG). It was developed by the CORMAmed™ Domain Task Force under OMG.

Introduction

The purpose of this paper is to provide a back ground for the design and development of a Metadata Query Service and provide the necessary normative information required for accessing information from a Metadata Repository.

Over the years, government and industry have developed healthcare information systems. The healthcare industry has created a number of standards for the interchange of data. These organizations include Military Health System (MHS), Health Care Financing Administration (HCFA), industry-sponsored associations, and American National Standards Institute (ANSI) accredited Standards Development Organizations (SDOs). The National Committee on Vital and Health Statistics (NCVHS) is advising the Secretary of Health and Human Services (HHS) on directions to follow to meet the requirements of the Health Insurance Portability and Accountability Act (HIPAA) of 1996. HHS will mandate by regulation the standards for use in electronic healthcare data interchange.

The data element information within these emerging and existing standards is not readily available to standards developers or system developers. Regulators, information system developers, SDOs, and other standards users do not have access to complete, consistent, and easy-to-access metadata registries or dictionaries. Tools are needed to readily identify, compare, and extract relevant standards for their use.

This standard specifies the behavioral aspects of a data registry. The behavior is stated in the form of an interface specification better known as a Metadata Query Service. The Metadata Query Service provides a way to access the information in a data registry in a well-defined way.

This standard defines the access to a metadata registry utilizing the *ISO/IEC 14750:1999 Information Technology - Open Distributed Processing -Interface Definition Language* (IDL) which can be bound to a multitude of programming languages, platforms, and operating systems. The IDL is separated into eleven files.

The Access file is used to provide the necessary methods for accessing the metadata registry.

- MQSAccess.idl

The following are used to denote constants that are needed by many of the method calls:

- MQSAdministeredComponent.idl
- MQSAdministrationStatus.idl
- MQSClassificationScheme.idl
- MQSClassifiedComponent.idl
- MQSOrganization.idl
- MQSReferenceDocument.idl
- MQSRegistrationStatus.idl
- MQSVersion.idl

The MQSRegistrationAuthority.idl is used to provide the necessary structs required for the identification of a registration authority as well as an interface with accompanying methods for the transformation between qualified names and qualified code strings.

- MQSRegistrationAuthority.idl

The Value file denotes the structures of information that can be returned from a Metadata Repository

- MQSValue.idl

All of these files make up the normative section of this standard and are outlined in the Metadata Query Service - IDL.

Metadata Query Service

An Object Technology Extension

to the

ISO/IEC 11179 Specification and Standardization of Data Elements,

Part 3, Basic Attributes.

1 Scope

This is an extension to the *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes*. The *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attri*butes deals with the schema of a data registry where the descriptions of shareable data is stored. The Metadata Query Service is scoped at the level of accessing the information contained in a Metadata Registry.

The purpose of the Metadata Query Service is to provide Electronic Data Interchange (EDI) data element designers, software engineers, and end users a well-defined standard way of accessing the metadata from a data registry in order to curtail proprietary solutions and move to better solutions for Distributed Object Computing (DOC).

The *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes* defines the schema of a data registry; it can be used by a data registry management system in structuring a data registry.
The Metadata Query Service will provide the necessary run-time access to the data registry for the retrieval of information. This run-time access can be used to:

- build applications on the fly
- assist in data validation
- assist in mediating data across disparate systems.

2 Conformance

Because MQS is an object extension to the *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes* it must conform to that standard in terms of the classes, attributes and associations as defined by the normative section containing the Unified Modeling Language (UML) representation.

3 Normative reference(s)

The following standards contain provisions, which, through reference in the text, constitute provisions for this Part of the International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Part of the International Standard are encouraged to investigate the possibility of applying the most recent editions of standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

American National Standard for Information Technology – Metamodel for the Management of Shareable Data
ANSI X3.285 - 1998

4 Term(s) and definition(s)

For the purpose of this Part of the International Standard the following definitions apply:

Behavior
The observable effects of performing the requested service (including its results).

Const
Describes any object that cannot have its value changed through an expression involving the name being declared const.[2]

Exception
Any abstraction of an exceptional or error condition that is identified and raised by an operation.[3]

IDL
ISO/IEC 14750:1999 Information Technology - Open Distributed Processing -Interface Definition Language (IDL): is the language used to describe the interfaces that client objects call and object implementations provide. An interface definition written in IDL completely defines the interface and fully specifies each operation's parameters. An IDL interface provides the information needed to develop clients that use the interface's operations. Clients are not written in IDL, which is purely a descriptive language, but in languages for which mappings from IDL concepts have been defined. The mapping of an IDL concept to a client language construct will depend on the facilities available in the client language. For example, an IDL exception might be mapped to a structure in a language that has no notion of exception, or to an exception in a language that does.

Interface
A description of a set of possible uses of an object. Specifically, an interface describes a set of potential requests in which an object can participate meaningfully.

Implementation
A definition that provides the information needed to create an object, allowing the object to participate in providing an appropriate set of services. An implementation typically includes a desrption of the data structure used to represent the core state associated with an object, as well as definitions of the methods that access that data structure. It also typically includes information about the intended type of the object.

Method
The hidden implementation of an associated operation. An implementation of a responsibility. A discrete activity, action, or behavior that implements a functional or process abstraction, is performed by, belongs to, and is encapsulated in an object or class, and usually implements a service.[4] Code that may be executed to perform a requested service.

Operation
Any discrete activity, action, or behavior that is performed by an object or class.[5]

---

[2] Donald G. Firesmith & Edward M. Eykholt. *Dictionary of Object Technology*. SIGS Reference Library.1995.

[3] Donald G. Firesmith & Edward M. Eykholt. *Dictionary of Object Technology*. SIGS Reference Library.1995.

[4] Donald G. Firesmith & Edward M. Eykholt. *Dictionary of Object Technology*. SIGS Reference Library.1995.

[5] Donald G. Firesmith & Edward M. Eykholt. *Dictionary of Object Technology*. SIGS Reference Library.1995.

Query
An activity that invloves selecting objects from implicitly or explicitly identified collections based on a specified predicate.

Raises
To abandon normal execution and notify a calling operation or local exception handler of the occurrence of a develop-defined exception.[6]

Sequence
In IDL, you can declare a sequence of any IDL data type. An IDL sequence is similar to a one-dimensional array of elements. An IDL sequence does not have a fixed length. If the sequence has a fixed maximum length, then the sequence is bounded. Otherwise, the sequence is unbounded.

Service
A computation that may be performed in response to a request.

Struct
A struct data type allows you to package a set of named members of various types.

Typedef
The typedef keyword allows you to define a meaningful or more simple name for an IDL type.

Use Case
A sequence of transactions performed by a system in response to a triggering event initiated by an actor to the system. A full use case should provide a measurable value to an actor when the actor is performing a certain task. A use case contains a set of scenarios that explain various sequences of interaction within the transaction. A use case can also describe the behaviour of a set of objects, such as an organization. A use case may have a name, although it is typically not a simple name. It is often written as an informal text description of the external actors and the sequences of events between objects that make up the transaction. Use case names often start with a verb. For example, names of possible use cases for a bank include Identify Account, Prepare Statement, Audit.

4 Symbols

5 Metadata Query Service - IDL

The following describes the necessary IDL for allowing access to information contained in a metadata registry.

5.2 Access

The MQSAccess.idl is used to define the necessary interfaces and their respective methods for retrieving information from a metadata registry. The five interfaces are Stewardship, Classification, DataElement, DataElementConcept and ValueDomain. Each of which is used to provide a certain level of access into a metadata registry and defines the methods necessary for retrieving information from a metadata registry for that particular aspect of the registry. These interfaces correspond to the regions defined in the the *ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes* standard.

```
// File: MQSAccess.idl
#ifndef _MQS_ACCESS_IDL_
#define _MQS_ACCESS_IDL_
#include "MQSRegistrationAuthority.idl"
```

[6] Donald G. Firesmith & Edward M. Eykholt. *Dictionary of Object Technology*. SIGS Reference Library.1995.

```
module MQSAccess
{
	//////////////////////////////////////////////////////////////////
	// EXTERNAL TYPEDEFS
	//////////////////////////////////////////////////////////////////
	typedef MQSRegistrationAuthority::QualifiedCodeStr QualifiedCodeStr;
	typedef MQSRegistrationAuthority::QualifiedPersonId QualifiedPersonId;
	typedef MQSRegistrationAuthority::VersionIdentifier VersionIdentifier;
	typedef MQSRegistrationAuthority::InternationalRegistrationDataIdentifier InternationalRegistrationDataIdentifier;
	typedef MQSRegistrationAuthority::RegistrationAuthority RegistrationAuthority;
	typedef MQSRegistrationAuthority::RegistrationAuthorityIdentifier RegistrationAuthorityIdentifier;

	//////////////////////////////////////////////////////////////////
	// FORWARD DECLARATIONS
	//////////////////////////////////////////////////////////////////
	interface Stewardship;
	interface Classification;
	interface DataElement;
	interface DataElementConcept;
	interface ValueDomain;
	interface AdministeredComponentIterator;
	interface AbstractManagedObject;

	//////////////////////////////////////////////////////////////////
	// STRUCTS
	//////////////////////////////////////////////////////////////////

	// AccessComponent
	//////////////////////////////////////////////////////
	struct AccessComponentData
	{
		Stewardship		stewardship_access;
		Classification		classification_access;
		DataElement		data_element_access;
		DataElementConcept	data_element_concept_access;
		ValueDomain		value_domain_access;
	};
	struct PostalAddress
	{
		string			street;
		QualifiedCodeStr	city;
		QualifiedCodeStr	state_province;
		string			zip_postal_code;
		QualifiedCodeStr	country;
	};
	struct Phone
	{
		string	business;
		string	cellular;
		string	fax;
		string	pager;
	};
	struct ContactInformation
	{
		PostalAddress	postal_address;
		Phone		phone;
		string		eMail;
		string		webPage;
	};
```

```
struct ContactPerson
{
        QualifiedPersonId       name;
        QualifiedCodeStr        title;
        ContactInformation      contact_information;
};
struct Organization
{
        QualifiedCodeStr        label;
        string                  mail_address;
};
struct ReferenceDocument
{
        QualifiedCodeStr        type;
        string                  label;
};
struct ComponentRegistrationAuthority
{
        VersionIdentifier                       version_identifier;
        InternationalRegistrationDataIdentifier IRDI;
        QualifiedCodeStr                        administration_status;
        QualifiedCodeStr                        registration_status;
};
typedef string ComponentNameContext;
struct NameContext
{
        ComponentNameContext name;
        VersionIdentifier    version_identifier;
        string               description;
        string               label;
        QualifiedCodeStr     language;
};
typedef sequence <NameContext> NameContextSequence;
struct AdministeredComponent
{
        NameContextSequence             name_context;
        QualifiedCodeStr                administered_component_type;
        ComponentRegistrationAuthority  component_registration_authority;
        RegistrationAuthority           registration_authority;
        sequence <any,1>                value;
};

////////////////////////////////////////////////////////////////
// CONSTANTS
////////////////////////////////////////////////////////////////
const QualifiedCodeStr CONTAINS =       "1000/NCITSL8/MatchCriteria/0/CONTAINS";
const QualifiedCodeStr WHOLEWORD =      "1000/NCITSL8/MatchCriteria/0/WHOLEWORD";
const QualifiedCodeStr BEGINSWITH =     "1000/NCITSL8/MatchCriteria/0/BEGINSWITH";
const QualifiedCodeStr EXACT =          "1000/NCITSL8/MatchCriteria/0/EXACT";

////////////////////////////////////////////////////////////////
// TYPEDEFS
////////////////////////////////////////////////////////////////
typedef string Registrar;
typedef  sequence<QualifiedCodeStr,1> OptionalQualifiedCodeStrSequence;
typedef  sequence<string,1> OptionalStringSequence;

////////////////////////////////////////////////////////////////
// SEQUENCES
```

```
//////////////////////////////////////////////////////////////////
typedef sequence <Registrar> RegistrarSequence;
typedef sequence <Organization> OrganizationSequence;
typedef sequence <ContactPerson> ContactPersonSequence;
typedef sequence <ReferenceDocument> ReferenceDocumentSequence;
typedef sequence <RegistrationAuthority> RegistrationAuthoritySequence;
typedef sequence <QualifiedCodeStr> QualifiedCodeStrSequence;
typedef sequence <AdministeredComponent> AdministeredComponentSequence;

//////////////////////////////////////////////////////////////////
// EXCEPTIONS
//////////////////////////////////////////////////////////////////
exception NotImplemented
{
};
exception InvalidOrganization
{
        QualifiedCodeStr organization_label;
};
exception InvalidOrganizationType
{
        QualifiedCodeStr organization_type;
};
exception InvalidNumber
{
        long number;
};
exception InvalidMatchCriteria
{
        QualifiedCodeStr criteria;
};
exception InvalidOrganizationLabel
{
        QualifiedCodeStr organization_label;
};
exception InvalidRegistrationAuthorityIdentifier
{
        RegistrationAuthorityIdentifier identifier;
};
exception InvalidAdministeredComponentType
{
        QualifiedCodeStr administered_component_type;
};
exception InvalidVersionIdentifier
{
        QualifiedCodeStr version_identifier;
};
exception InvalidAdministrationStatus
{
        QualifiedCodeStr administration_status;
};
exception InvalidRegistrationStatus
{
        QualifiedCodeStr registration_status;
};
exception InvalidInternationalRegistrationDataIdentifier
{
        InternationalRegistrationDataIdentifier IRDI;
};
```

```
exception InvalidAdministeredComponentName
{
        string administered_component_name;
};
exception InvalidLabel
{
        string label;
};
exception InvalidLanguage
{
        QualifiedCodeStr language;
};
exception InvalidReferenceDocumentLabel
{
        string reference_document_label;
};
exception InvalidReferenceDocumentType
{
        QualifiedCodeStr reference_document_type;
};

//////////////////////////////////////////////////////////////////
// INTERFACES
//////////////////////////////////////////////////////////////////
interface AbstractManagedObject
{
        void done();
};
//////////////////////////////////////////////////////////
interface AccessComponent
{
        readonly attribute string mqs_verison;
        AccessComponentData get_components();
};
//////////////////////////////////////////////////////////
interface AdministeredComponentIterator : AbstractManagedObject
{
        unsigned long max_left();
        boolean next_n
        (
                in unsigned long n,
                out AdministeredComponentSequence administered_component_sequence
        );
};
//////////////////////////////////////////////////////////
interface Stewardship : AccessComponent
{
        ////////////////////////////////////////////////////
        // Organization
        ////////////////////////////////////////////////////
        Organization get_organization
        (
                in QualifiedCodeStr organization_label
        )
        raises
        (
                InvalidOrganization
        );
```

```
OrganizationSequence get_organizations
(
        in QualifiedCodeStr organization_type
)
raises
(
        InvalidOrganizationType
);

OrganizationSequence matchOrganizations
(
        in OptionalQualifiedCodeStrSequence     organization_type,
        in long                                 number,
        in QualifiedCodeStr                     criteria,
        in QualifiedCodeStr                     organization_label
)
raises
(
        InvalidOrganizationType,
        InvalidNumber,
        InvalidMatchCriteria
);

//////////////////////////////////////////////////////
// Contact persons
//////////////////////////////////////////////////////
ContactPersonSequence get_contact_persons
(
        in QualifiedCodeStr organization_label
)
raises
(
        InvalidOrganizationLabel
);
ContactPersonSequence match_contact_persons
(
        in QualifiedCodeStr     organization_label,
        in long                 number,
        in QualifiedCodeStr     criteria,
        in string               text
)
raises
(
        InvalidOrganizationLabel,
        InvalidNumber,
        InvalidMatchCriteria
);

//////////////////////////////////////////////////////
// Reference documents
//////////////////////////////////////////////////////
ReferenceDocumentSequence get_reference_documents
(
        in QualifiedCodeStr organization_label
)
raises
(
        InvalidOrganizationLabel
);
```

```
ReferenceDocumentSequence match_reference_documents
(
        in QualifiedCodeStr                     organization_label,
        in long                                 number,
        in QualifiedCodeStr                     criteria,
        in OptionalStringSequence                       reference_document_label,
        in OptionalQualifiedCodeStrSequence     reference_document_type
)
raises
(
        InvalidOrganizationLabel,
        InvalidNumber,
        InvalidMatchCriteria,
        InvalidReferenceDocumentLabel,
        InvalidReferenceDocumentType
);

////////////////////////////////////////////////////
// Registration authority
////////////////////////////////////////////////////
RegistrationAuthoritySequence get_registration_authoritys();
RegistrarSequence getRegistrars
(
        in RegistrationAuthorityIdentifier identifier
)
raises
(
        InvalidRegistrationAuthorityIdentifier
);

////////////////////////////////////////////////////
// AdministeredComponent
////////////////////////////////////////////////////
AdministeredComponent get_administered_component
(
        in InternationalRegistrationDataIdentifier IRDI
)
raises
(
        InvalidInternationalRegistrationDataIdentifier
);

// Organization type
////////////////////////////////////////////////////
AdministeredComponentSequence get_administered_components_by_submitting_organization
(
        in QualifiedCodeStr                     organization_label,
        in QualifiedCodeStrSequence             administered_component_type,
        in unsigned long                        max_sequence,
        out AdministeredComponentIterator               the_rest
)
raises
(
        InvalidOrganizationLabel,
        InvalidAdministeredComponentType
);

AdministeredComponentSequence get_administered_components_by_responsible_organization
(
```

```
        in QualifiedCodeStr                 organization_label,
        in QualifiedCodeStrSequence         administered_component_type,
        in unsigned long                    max_sequence,
        out AdministeredComponentIterator           the_rest
)
raises
(
        InvalidOrganizationLabel,
        InvalidAdministeredComponentType
);

// Status
////////////////////////////////////////////////////
AdministeredComponentSequence get_administered_components_by_administration_status
(
        in RegistrationAuthorityIdentifier  identifier,
        in QualifiedCodeStr                 version_identifier,
        in QualifiedCodeStrSequence         administered_component_type,
        in QualifiedCodeStrSequence         administration_status,
        in unsigned long                    max_sequence,
        out AdministeredComponentIterator           the_rest
)
raises
(
        InvalidRegistrationAuthorityIdentifier,
        InvalidVersionIdentifier,
        InvalidAdministeredComponentType,
        InvalidAdministrationStatus
);

AdministeredComponentSequence get_administered_components_by_registration_status
(
        in RegistrationAuthorityIdentifier  identifier,
        in QualifiedCodeStr                 version_identifier,
        in QualifiedCodeStrSequence         administered_component_type,
        in QualifiedCodeStrSequence         registration_status,
        in unsigned long                    max_sequence,
        out AdministeredComponentIterator           the_rest
)
raises
(
        InvalidRegistrationAuthorityIdentifier,
        InvalidVersionIdentifier,
        InvalidAdministeredComponentType,
        InvalidRegistrationStatus
);

// Name and name context
////////////////////////////////////////////////////
AdministeredComponent get_administered_component_by_component_name_context
(
        in string administered_component_name
)
raises
(
        InvalidAdministeredComponentName
);

AdministeredComponentSequence get_administered_components_by_name_context
```

```
                (
                        in OptionalQualifiedCodeStrSequence     version_identifier,
                        in OptionalStringSequence                       description,
                        in OptionalStringSequence                       label,
                        in OptionalQualifiedCodeStrSequence     language
                )
                raises
                (
                        InvalidVersionIdentifier,
                        InvalidLabel,
                        InvalidLanguage
                );

                // Matching
                ////////////////////////////////////////////////////
                AdministeredComponentSequence match_administered_components
                (
                        in RegistrationAuthorityIdentifier      identifier,
                        in QualifiedCodeStr                     criteria,
                        in string                               text_to_match_on,
                        in QualifiedCodeStrSequence             administration_status,
                        in QualifiedCodeStrSequence             registration_status,
                        in QualifiedCodeStr                     version_identifier,
                        in QualifiedCodeStrSequence             administered_component_type,
                        in unsigned long                        max_sequence,
                        out AdministeredComponentIterator               the_rest
                )
                raises
                (
                        InvalidRegistrationAuthorityIdentifier,
                        InvalidVersionIdentifier,
                        InvalidAdministeredComponentType,
                        InvalidAdministrationStatus
                );
        };
        ////////////////////////////////////////////////////////
        interface Classification : AccessComponent
        {
        };
        ////////////////////////////////////////////////////////
        interface DataElement : AccessComponent
        {
        };
        ////////////////////////////////////////////////////////
        interface DataElementConcept : AccessComponent
        {
        };
        ////////////////////////////////////////////////////////
        interface ValueDomain : AccessComponent
        {
        };
};
#endif  // _MQS_ACCESS_IDL_
```

5.3 Administered Component

The MQSAdministeredComponent.idl is used to provide the necessary constants required for the identification of administered components as defined in the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes.

```
// file MQSAdministeredComponent.idl
#ifndef _MQS_ADMINISTERED_COMPONENT_IDL_
#define _MQS_ADMINISTERED_COMPONENT_IDL_

#include "MQSAccess.idl"
module MQSVersion {
	typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
	const QualifiedCodeStr ALLADMINISTEREDCOMPONENTS =
		"1000/NCITSL8/AdministeredComponent/0/AllAdministeredComponents";
	const QualifiedCodeStr CLASSIFICATIONSCHEME =
		"1000/NCITSL8/AdministeredComponent/0/ClassificationScheme";
	const QualifiedCodeStr CLASSIFIEDCOMPONENT =
		"1000/NCITSL8/AdministeredComponent/0/ClassifiedComponent";
	const QualifiedCodeStr DATAELEMENT =
		"1000/NCITSL8/AdministeredComponent/0/DataElement";
	const QualifiedCodeStr RULE =
		"1000/NCITSL8/AdministeredComponent/0/Rule";
	const QualifiedCodeStr VALUEDOMAIN =
		"1000/NCITSL8/AdministeredComponent/0/ValueDomain";
};
#endif // _MQS_ADMINISTERED_COMPONENT_IDL_
```

6.3 Administration Status

The MQSAdministrationStatus.idl is used to provide the necessary constants required for the identification of administration status as defined in the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes.

```
// file MQSAdministrationStatus.idl
#ifndef _MQS_ADMINISTRATION_STATUS_IDL_
#define _MQS_ADMINISTRATION_STATUS_IDL_

#include "MQSAccess.idl"
module MQSAdministeredStatus {
	typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
	const QualifiedCodeStr ALLADMINISTRATIONSTATUS =
		"1000/NCITSL8/AdministrationStatus/0/AllAdministrationStatus";
	const QualifiedCodeStr CERTIFIED =
		"1000/NCITSL8/AdministrationStatus/0/Certified";
	const QualifiedCodeStr DRAFT =
		"1000/NCITSL8/AdministrationStatus/0/Draft";
	const QualifiedCodeStr RECEIVED =
		"1000/NCITSL8/AdministrationStatus/0/Received";
	const QualifiedCodeStr REJECTED =
		"1000/NCITSL8/AdministrationStatus/0/Rejected";
};
#endif // _MQS_ADMINISTRATION_STATUS_IDL_
```

6.4 Classification Scheme

The MQSClassificationScheme.idl is used to provide the necessary constants required for the identification of classification schemes as defined in the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes.

```
// file MQSClassificationScheme.idl
#ifndef _MQS_CLASSIFICATION_SCHEME_IDL_
#define _MQS_CLASSIFICATION_SCHEME_IDL_

#include "MQSAccess.idl"
module MQSClassificationScheme {
        typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
        const QualifiedCodeStr LIST =
                "1000/NCITSL8/ClassificationScheme/0/List";
        const QualifiedCodeStr TAXONOMY =
                "1000/NCITSL8/ClassificationScheme/0/Taxonomy";
        const QualifiedCodeStr NETWORK =
                "1000/NCITSL8/ClassificationScheme/0/Network";
        const QualifiedCodeStr ONTOLOGY =
                "1000/NCITSL8/ClassificationScheme/0/Ontology";
};
#endif // _MQS_CLASSIFICATION_SCHEME_IDL_
```

6.5 Classified Component

The MQSClassifiedComponent.idl is used to provide the necessary constants required for the identification of classified components as defined in the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes.

```
// file MQSClassifiedComponent.idl
#ifndef _MQS_CLASSIFIED_COMPONENT_IDL_
#define _MQS_CLASSIFIED_COMPONENT_IDL_

#include "MQSAccess.idl"
module MQSClassifiedComponent {
        typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
        const QualifiedCodeStr CONCEPTUALDOMAIN =
                "1000/NCITSL8/ClassifiedComponent/0/ConceptualDomain";
        const QualifiedCodeStr DATAELEMENTCONCEPT =
                "1000/NCITSL8/ClassifiedComponent/0/DataElementConcept";
        const QualifiedCodeStr OBJECTCLASS =
                "1000/NCITSL8/ClassifiedComponent/0/ObjectClass";
        const QualifiedCodeStr REPRESENTATIONCLASS =
                "1000/NCITSL8/ClassifiedComponent/0/RepresentationClass";
};
#endif // _MQS_CLASSIFIED_COMPONENT_IDL_
```

6.6 Organization

The MQSOrganization.idl is used to provide the necessary constants required for the identification of the types of organizations as defined in the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes.

```
// file MQSOrganization.idl
#ifndef _MQS_ORGANIZATION_IDL_
#define _MQS_ORGANIZATION_IDL_
```

```
#include "MQSAccess.idl"
module MQSOrganization {
	typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
	const QualifiedCodeStr ALLORGANIZATIONS =
		"1000/NCITSL8/Organization/0/AllOrganizations";
	const QualifiedCodeStr SUBMITTING =
		"1000/NCITSL8/Organization/0/Submitting";
	const QualifiedCodeStr RESPONSIBLE =
		"1000/NCITSL8/Organization/0/Responsible";
};
#endif // _MQS_ORGANIZATION_IDL_
```

6.7 Reference Document

The MQSReferenceDocument.idl is used to provide the necessary constants required for the identification of the types of reference documents that may be used.

```
// file MQSReferenceDocument.idl
#ifndef _MQS_REFERENCE_DOCUMENT_IDL_
#define _MQS_REFERENCE_DOCUMENT_IDL_

#include "MQSAccess.idl"
module MQSReferenceDocument {
	typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
	const QualifiedCodeStr ALLREFERENCEDOCUMENTS =
		"1000/NCITSL8/ReferenceDocument/0/AllReferenceDocuments";
	const QualifiedCodeStr DICTIONARY =
		"1000/NCITSL8/ReferenceDocument/0/Dictionary";
	const QualifiedCodeStr ENCYCLOPEDIA =
		"1000/NCITSL8/ReferenceDocument/0/Encyclopedia";
	const QualifiedCodeStr MAGAZINE =
		"1000/NCITSL8/ReferenceDocument/0/Magazine";
};
#endif // _MQS_REFERENCE_DOCUMENT_IDL_
```

6.8 Registration Authority

The MQSRegistrationAuthority.idl is used to provide the necessary structs required for the identification of a registration authority as well as an interface with accompanying methods for the transformation between qualified names and qualified code strings.

```
// MQSRegistrationAuthority.idl
#ifndef _MQS_REGISTRATION_AUTHORITY_IDL_
#define _MQS_REGISTRATION_AUTHORITY_IDL_

module MQSRegistrationAuthority
{
	typedef string QualifiedCodeStr;
	typedef string DataIdentifier;
	typedef string VersionIdentifier;

	struct RegistrationAuthorityIdentifier
	{
		long	international_code_designator;
		string	organization_identifier;
		string	organization_part_identifier;
		long	OPI_source_indicator;
	};
```

```
        struct InternationalRegistrationDataIdentifier
        {
                RegistrationAuthorityIdentifier     RAI;
                DataIdentifier                      DI;
                VersionIdentifier                   VI;
        };

        struct QualifiedDataElement
        {
                RegistrationAuthorityIdentifier     RAI;
                string                              local_name;
        };

        struct QualifiedPersonId
        {
                RegistrationAuthorityIdentifier     authority_id;
                string                              local_name;
        };

        struct RegistrationAuthority
        {
                QualifiedCodeStr                    documentation_language;
                RegistrationAuthorityIdentifier     identifier;
        };

        interface Translation
        {
                QualifiedCodeStr qualified_code_to_str
                (
                        in QualifiedDataElement qualified_data_element
                );

                QualifiedDataElement str_to_qualified_name
                (
                        in QualifiedCodeStr qualified_code_str
                );
        };
};

#endif // _MQS_REGISTRATION_AUTHORITY_IDL_
```

6.9 Registration Status

The MQSRegistrationStatus.idl is used to provide the necessary constants required for the identification of registration status as defined in the ISO/IEC 11179 Specification and Standardization of Data Elements, Part 3, Basic Attributes.

```
// file MQSRegistrationStatus.idl
#ifndef _MQS_REGISTRATION_STATUS_IDL_
#define _MQS_REGISTRATION_STATUS_IDL_

#include "MQSAccess.idl"
module MQSRegistrationStatus {
        typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
        const QualifiedCodeStr ALLREGISTRATIONSTATUS =
                "1000/NCITSL8/RegistrationStatus/0/AllRegistrationStatus";
        const QualifiedCodeStr CERTIFIED =
                "1000/NCITSL8/RegistrationStatus/0/Certified";
```

```
        const QualifiedCodeStr INCOMPLETE =
                "1000/NCITSL8/RegistrationStatus/0/Incomplete";
        const QualifiedCodeStr RECORDED =
                "1000/NCITSL8/RegistrationStatus/0/Recorded";
        const QualifiedCodeStr RETIRED =
                "1000/NCITSL8/RegistrationStatus/0/Retired";
        const QualifiedCodeStr STANDARDIZED =
                "1000/NCITSL8/RegistrationStatus/0/Standardized";
};
#endif // _MQS_REGISTRATION_STATUS_IDL_
```

6.10 Value

The MQSValue.idl is used to provide the necessary structs for the representation of information that can be used as a return value from an interface's methods.

```
// File: MQSValue.idl
#ifndef _MQS_VALUE_IDL_
#define _MQS_VALUE_IDL_

#include "MQSAccess.idl"
module MQSValue
{
        typedef MQSRegistrationAuthority::QualifiedCodeStr QualifiedCodeStr;
        //////////////////////////////////////////////////////////////////
        // ClassificationScheme
        //////////////////////////////////////////////////////////////////
        typedef QualifiedCodeStr ClassificationSchemeType;

        //////////////////////////////////////////////////////////////////
        // ClassifiedComponent
        //////////////////////////////////////////////////////////////////
        struct Definition
        {
                string     context_identifier;
                string     text;
        };
        typedef sequence <Definition> DefinitionSequence;

        struct ClassifiedComponent
        {
                DefinitionSequence          definition;
        };
        typedef sequence <ClassifiedComponent> ClassifiedComponentSequence;

        struct ClassificationScheme
        {
                QualifiedCodeStr                ClassificationSchemeType;
                ClassifiedComponentSequence     parent;
                QualifiedCodeStr                relation;
                ClassifiedComponentSequence     child;
        };
        //////////////////////////////////////////////////////////////////
        // Rule
        //////////////////////////////////////////////////////////////////
        typedef string Rule;

        //////////////////////////////////////////////////////////////////
```

```
        // ValueDomain
        ////////////////////////////////////////////////////////////////
        struct ValueDomain
        {
                string    label;
                long      minimum_character_quantity;
                long      maximum_character_quantity;
                string    format;
                string    data_type;
                string    character_type;
                string    unit_of_quantity;
        };

        typedef sequence <string> ExampleSequence;
        ////////////////////////////////////////////////////////////////
        // DataElement
        ////////////////////////////////////////////////////////////////
        struct DataElement
        {
                Rule                    rule;
                ClassifiedComponent     representation_class;
                string                  representation_class_qualifier;
                ExampleSequence         example;
        };
};
#endif // _MQS_VALUE_IDL_
```

6.11 Version

The MQSVersion.idl is used to provide the necessary constants required for the identification of versions.

```
// file MQSVersion.idl
#ifndef _MQS_VERSION_IDL_
#define _MQS_VERSION_IDL_

#include "MQSAccess.idl"
module MQSVersion {
        typedef MQSAccess::QualifiedCodeStr QualifiedCodeStr;
        const QualifiedCodeStr ALLVERSIONS =
                "1000/NCITSL8/Version/0/AllVersions";
        const QualifiedCodeStr EARLIEST =
                "1000/NCITSL8/Version/0/Earliest";
        const QualifiedCodeStr LATEST =
                "1000/NCITSL8/Version/0/Latest";
};
#endif // _MQS_VERSION_IDL_
```

6 MQS Conformance

The following taxonomy is defined for specific conformance classes of MQS implementations. An implementation claiming conformance to any of these classes must conform to all of the interfaces specified for that class. An implementation may claim conformance to multiple conformance classes as long as it is conformant to each one it claims. Each row in the following table includes the specification for a different conformance class. The columns represent the interfaces AccessComponent. A star '*' in a column indicates the conformance class in that row includes the interface of that column.

Conformance Class	**Stewardship Interface**	**Classification Interface**	**DataElement Interface**	**DataElementConcept Interface**	**ValueDomain Interface**
Simple MQS	*				

Annex A
(informative)

Use Cases

Use cases are used in the Analysis and Design phase of software engineering and play a critical role in understanding how a user will want to use a system. The following use cases were instrumental in assisting in creating the IDL necessary to handle requests from users for information coming from a metadata registry.

A.1 View Information Based on Criteria

Table 1 — View Information Based on Criteria

Name	View Information Based on Criteria
Summary	User selects data element search. The system displays a form where the user specifies the criteria to be used in the search. This criteria can be a word used in a name or definition and further limited by such criteria as the registration authority, data element type, administrative and registration status, submitting organization, or application system. Based on the criteria, the system will display a list of names for the data element type from which the user may drill-down to more detail.
Actor(s)	Standards User, System Developer, Standards Developer
Pre-Conditions	None.
Begins When	User selects data element search.
Description	System displays a search query form screen. (See Attachment 1.) User fills in criteria or accepts the defaults provided by the system. System displays a list of names meeting the criteria. User selects a name. System displays information in standard format.
Ends When	System displays detailed information in standard information.
Exceptions	No information found meeting the criteria. System will display error message and pointer to query the help screens.
Post-Conditions	User may request that detail information for previous name or next name in the query result list be displayed.
Traceability	II.3 Information can be viewed based on criteria. II.7 Queries should be available by data element. II.11 Queries should be available by registry.

A.2 Data Element Usage

Table 2 — Show Data Element Usage

Name	Show Data Element Usage
Summary	User selects a data element, and the system displays a list of data elements linked directly or indirectly to the data element. The user then both selects and requests the display of information for an individual data element on the list or selects several data elements on the list and requests the generation of a comparison matrix report.
Goal	Allow a user to identify possible sources of additional information on a data element's meaning and usage by reviewing other information linked directly or indirectly to it.
Actor(s)	Standards User, Standards Developer.
Pre-Conditions	User enters Services Area of data registry.
Begins When	User selects Show Data Element Usage report.
Description	System displays a data element search screen that allows a user to make a selection based on a data element name or letter of alphabet. User requests a data element name or a letter of the alphabet. System displays a list of data elements beginning with the data element name closest to the word entered or beginning with the selected letter of the alphabet. User selects a data element. System displays a list of data elements that are linked directly or indirectly to the data element. (See Attachment 1.)
Ends When	System displays data usage information.
Exceptions	None.
Post-Conditions	User can select a data element, and the system displays its information in the standard format. User can select several data elements and request the generation of a comparison matrix. User can select a data element and request a show data element usage be generated.
Traceability	II.2 Data registry should be able to identify, compare, and extract relevant standards.

A.3 Download Information

Table 3 — Download Information

Name	Download Information
Summary	While viewing a detailed data element information report, the user requests the download of the data element's metadata. The system requests some user identification information and then downloads the information.
Goal	Ability to download the metadata for a data element including permissible values for a coded data element.
Actor(s)	Systems Developer.
Pre-Conditions	None.
Begins When	User selects 'Download Data Element Information' from list provided when the detailed data element information report has been provided to the user.
Description	The system requests the user's name, organization name, user's telephone number, and email address and questions whether all the metadata or just the domain values are to be downloaded. User provides requested information and indicates whether all the metadata or just the domain values are to be downloaded. The system downloads the information.
Ends When	The system downloads the information.
Exceptions	None.
Post-Conditions	None.
Traceability	II.13 The user will be able to select data elements and request that the metadata for the data element be downloaded at any time during the session.

A.4 Generate Comparison Matrix

Table 4 — Generate Comparison Matrix

Name	Generate Comparison Matrix
Summary	User selects an area of interest, various registration authorities, and then data elements that are displayed in matrix format to facilitate comparison of the metadata.
Goal	Allow a user to utilize data element mappings to a high level model that were done independently and separately to group, select, and compare data elements which might have the same meaning.
Actor(s)	Standards User, System Developer, Standards Developer.
Pre-Conditions	User enters Services area of data registry. Data elements have previously been mapped to high level model.
Begins When	User selects Generate Comparison Matrix report.
Description	System displays High Level Model. User selects area of interest. System displays a list of registration authorities that have data elements mapped to the selected area of interest. User selects registration authorities to be included. System displays a list of data elements that are mapped to the selected area of interest. The list also shows previously linked data elements in a matrix format. See Attachment 1 for example of format. User selects data elements to include in the comparison matrix. (Note: During the process of selecting data elements, a user can display detailed information on an individual data element to determine whether to include it or not.) System displays comparison matrix with rows for registration authority, data element ID, data element name, definition context, and data domain information. The columns include the row descriptions in column one and the data element information in the other columns. (See Attachment 2 for example.)
Ends When	System displays comparison matrix.
Exceptions	None.
Post-Conditions	User can select a data element, and the system displays its detailed information.
Traceability	II.1 Queries should be available by high level AIHW model. II.2 Data registry should be able to identify, compare, and extract relevant standards. II.4 Comparisons of same or similar data concepts will show names, definitions, allowable values (code sets), format, and size. II.11 Queries should be available by registry.

A.5 View Information Based on Keyword / Synonyms

Table 5 — View Information Based on Keyword / Synonyms

Name	View Information Based on Keyword / Synonyms
Summary	User enters keyword search area of data registry, locates the keyword of interest, and the system then displays data elements previously linked to that keyword.
Goal	Allow a user to locate data elements based on previously established keywords, which might not be used in the data element name or its definition, but which have been linked to the data element.
Actor(s)	Standards User.
Pre-Conditions	Keywords have been established. Data elements have previously been linked to keywords.
Begins When	User selects keywords from the main menu or the menu at the bottom of the screen.
Description	System displays keyword search screen that allows a user to make a selection based on a word or a letter of alphabet. User requests a word or a letter of the alphabet. System displays a list of keywords beginning with the keyword closest to the word entered or beginning with the selected letter of the alphabet. User selects a keyword. System displays a list of data elements that are linked to the keyword. User selects a data element. System displays the data element information in standard format.
Ends When	System displays data element information.
Exceptions	None.
Post-Conditions	None.
Traceability	II.10 Queries should be available by keywords.

A.6 Linking to Authoritative Source

Table 6 — Linking to Authoritative Source

Name	Linking to Authoritative Source
Summary	User researches an external data registry and finds a data element pertinent to a local data element. User enters information in the local data registry that links the data elements together. The link allows the external data element's information to be reviewed at a later time.
Goal	Ability to link data elements to information in other data registries and view the current information in that other data registry.
Actor(s)	Standards Developer.
Pre-Conditions	The external data registry information and minimal set of information about data elements has been registered in the general registry. See Table 1 for information required. User enters services area of data registry.
Begins When	User selects establish data element linkage.
Description	System displays list of registration authorities that are available. User selects two registration authorities (or indicates the data elements are both in the local data registry). System requests the user to enter a data element name or the beginning letter of the data element name for each of the two data elements to be linked. System display list of data element names, which may be linked. User indicates which two data elements to link and their type of relationship. System establishes linkage and informs user linkage established.
Ends When	System informs user linkage has been established.
Exceptions	None.
Post-Conditions	None.
Traceability	II.5 Data registry can link to other authoritative sources.

A.7 View Information Based on Lists

Table 7 — View Information Based on Lists

Name	View Information Based on Lists
Summary	User selects an area of interest from the following: data agreements, data collections, initiatives, or organizations. The system shows a list of topics within that area of interest. The user selects a topic. The system provides a general description of the topic and a list of the data elements linked to topic.
Actor(s)	Standards User, System Developer, Standards Developer
Pre-Conditions	None.
Begins When	User selects data agreements from the main menu or menu at the bottom of the screen.
Description	System displays a list of data agreements. User selects a data agreement of interest. System displays descriptive information for the selected data agreement, including a list of the data elements linked to that data agreement. User selects data element. System displays data element information in standard format (See Attachment 1.)
Ends When	System displays data element information.
Exceptions	None
Post-Conditions	User may select for display an Information Model Entity linked to this Data Element. User may select for display a Computer Aided System Engineering (CASE) Model linked to this Data Element. User may select for display a Data Agreement that includes this Data Element. User may select for display a Data Collection that includes this Data Element. User may select for display an Initiative that references this Data Element. User may select for display a Keyword that is related to this Data Element. User may download this Data Element's information.
Traceability	I.1 Keep facts about characteristics of data so they can be used to describe, inventory, analyze, and classify data. I.2 Require information about a data element that will provide a common understanding of its meaning, representation, and identification. I.3 Base structure of the data registry on the ISO/IEC 11179 and ANSI X3.285 standards. I.4 Implement a registry derived from AIHW Knowledgebase Registry and EPA Environmental Data Registry applications. I.4 Support version control for elements in the data registry. I.5 Use data elements and concepts that show names, definitions, allowable values (code sets), format, and size. II.8 Queries should be available by data agreement.

A.8 Generate Version Comparison Matrix

Table 8 — Generate Version Comparison Matrix

Name	Generate Version Comparison Matrix
Summary	User selects a data element, and versions of the data element metadata are displayed in matrix format to facilitate comparison.
Goal	Allow a user to determine what has changed between the various versions of a data element.
Actor(s)	Standards Developer.
Pre-Conditions	User enters Services area of data registry. System restricts data elements to those included in previously selected registration authorities.
Begins When	User selects Generate Version Comparison Matrix report.
Description	System displays Data Elements Search area. User selects a data element. System displays version comparison matrix with rows for the metadata specifications. The columns include the row descriptions in column one and the data element information in the other columns. The system will highlight what metadata has changed between the various versions. (See Attachment 1 for example.)
Ends When	System displays version comparison matrix.
Exceptions	None.
Post-Conditions	None.
Traceability	II.2 Data registry should be able to identify, compare, and extract relevant standards. II.4 Comparisons of same or similar data concepts will show names, definitions, allowable values (code sets), format, and size.

Bibliography

[1] Grady Booch, *Object-Oriented Design*. The Benjamin/Cummings Publishing Company, Inc. 1991.
[2] Jon Siegel, *CORBA Fundamental and Programming*. John Wiley & Sons, Inc. 1996.
[3] Martin Fowler , *Analysis Patterns - Reusable Object Models.* Addison-Wesley. 1997.
[4] Adele Goldberg & Kenneth S. Rubin, *Succeeding with Objects.* Addison-Wesley Publishing Company, Inc.. 1995