Gene Expression Specification

OMG Document dtc/~~2002~~2003-0~~9~~5-02

COSS, and IIOP are trademarks of the Object Management Group, Inc. X/Open is a trademark of X/Open Company Ltd.

ISSUE REPORTING

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.or*g, under Documents & Specifications, Report a Bug/Issue.

Table of Contents

List of Figures

Preface

About the Object Management Group

The Object Management Group, Inc. (OMG) is an international organization supported by over 600 members, including information system vendors, software developers and users. Founded in 1989, the OMG promotes the theory and practice of object-oriented technology in software development. The organization's charter includes the establishment of industry guidelines and object management specifications to provide a common frame-work for application development. Primary goals are the reusability, portability, and interoperability of object-based software in distributed, heterogeneous environments. Con-formance to these specifications will make it possible to develop a heterogeneous applications environment across all major hardware platforms and operating systems.

OMG's objectives are to foster the growth of object technology and influence its direction by establishing the Object Management Architecture (OMA). The OMA provides the conceptual infrastructure upon which all OMG specifications are based.

What is CORBA?

The Common Object Request Broker Architecture (CORBA), is the Object Management Group's answer to the need for interoperability among the rapidly proliferating number of hardware and software products available today. Simply stated, CORBA allows applications to communicate with one another no matter where they are located or who has designed them. CORBA 1.1 was introduced in 1991 by Object Management Group (OMG) and defined the Interface Definition Language (IDL) and the Application Pro-gramming Interfaces (API) that enable client/server object interaction within a specific implementation of an Object Request Broker (ORB). CORBA 2.0, adopted in December of 1994, defines true interoperability by specifying how ORBs from different vendors can

OMG Documents

The OMG documentation is organized as follows:

OMG Modeling

- ***Unified Modeling Language (UML) Specification*** defines a graphical language for visualizing, specifying, constructing, and documenting the artifacts of distributed object systems.

- ***Meta-Object Facility (MOF) Specification*** defines a set of CORBA IDL interfaces that can be used to define and manipulate a set of interoperable metamodels and their corresponding models.

• ***OMG XML Metadata Interchange (XMI) Specification*** supports the interchange of any kind of metadata that can be expressed using the MOF specification, including both model and metamodel information.

Object Management Architecture Guide

This document defines the OMG's technical objectives and terminology and describes the conceptual models upon which OMG standards are based. It defines the umbrella architecture for the OMG standards. It also provides information about the policies and procedures of OMG, such as how standards are proposed, evaluated, and accepted.

CORBA: Common Object Request Broker Architecture and Specification

Contains the architecture and specifications for the Object Request Broker.

OMG Interface Definition Language (IDL) Mapping Specifications

These documents provide a standardized way to define the interfaces to CORBA objects. The IDL definition is the contract between the implementor of an object and the client. IDL is a strongly typed declarative language that is programming language-independent. Language mappings enable objects to be implemented and sent requests in the developer's programming language of choice in a style that is natural to that language. The OMG has an expanding set of language mappings, including Ada, C, C++, COBOL, IDL to Java, Java to IDL, Lisp, and Smalltalk.

CORBAservices

Object Services are general purpose services that are either fundamental for developing useful CORBA-based applications composed of distributed objects, or that provide a universal-application domain-independent basis for application interoperability.

These services are the basic building blocks for distributed object applications. Compliant objects can be combined in many different ways and put to many different uses in applications. They can be used to construct higher level facilities and object frameworks that can interoperate across multiple platform environments.

Adopted OMG Object Services are collectively called CORBAservices and include specifications such as *Collection*, *Concurrency*, *Event*, *Externalization*, *Naming*, *Licensing*, *Life Cycle*, *Notification*, *Persistent Object*, *Property*, *Query*, *Relationship*, *Security*, *Time*, *Trader*, and *Transaction*.

CORBAfacilities

Common Facilities are interfaces for horizontal end-user-oriented facilities applicable to most domains. Adopted OMG Common Facilities are collectively called CORBAfacilities and include specifications such as *Internationalization and Time*, and *Mobile Agent Facility*.

Object Frameworks and Domain Interfaces

Unlike the interfaces to individual parts of the OMA "plumbing" infrastructure, Object Frameworks are complete higher level components that provide functionality of direct interest to end-users in particular application or technology domains.

Domain Task Forces concentrate on Object Framework specifications that include Domain Interfaces for application domains such as Finance, Healthcare, Manufacturing, Telecoms, E-Commerce, and Transportation.

Currently, specifications are available in the following domains:

- *CORBA Business*: Comprised of specifications that relate to the OMG-compliant interfaces for business systems.
- *CORBA Finance*: Targets a vitally important vertical market: financial services and accounting. These important application areas are present in virtually all organizations: including all forms of monetary transactions, payroll, billing, and so forth.
- *CORBA Healthcare*: Comprised of specifications that relate to the healthcare industry and represents vendors, healthcare providers, payers, and end users.
- *CORBA Manufacturing*: Contains specifications that relate to the manufacturing industry. This group of specifications defines standardized object-oriented interfaces between related services and functions.
- *CORBA Telecoms*: Comprised of specifications that relate to the OMG-compliant interfaces for telecommunication systems.
- *CORBA Transportation*: Comprised of specifications that relate to the OMG-compliant interfaces for transportation systems.

Obtaining OMG Documents

The OMG collects information for each book in the documentation set by issuing Requests for Information, Requests for Proposals, and Requests for Comment and, with its membership, evaluating the responses. Specifications are adopted as standards only when representatives of the OMG membership accept them as such by vote. (The policies and procedures of the OMG are described in detail in the *Object Management Architecture Guide*.)

OMG formal documents are available from our web site in PostScript and PDF format. To obtain print-on-demand books in the documentation set or other OMG publications, contact the Object Management Group, Inc. at:

OMG Headquarters

250 First Avenue

Needham, MA 02494

USA

Tel: +1-781-444-0404

Fax: +1-781-444-0320

pubs@omg.org

http://www.omg.org

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Helvetica bold - OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier bold` - Programming language elements.

Helvetica - Exceptions

Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Acknowledgements

Supporting Organizations

The following organizations have been involved in the process of developing, prototyping, and/or reviewing this submission. The submitters of this response thank the organizations for participating and giving their valuable input.

Affymetrix, Inc., Santa Clara, California

Agilent Technologies, Inc., Palo Alto, California

The Microarray Gene Expression Database (MGED) group

National Center for Genome Resources (NCGR), Santa Fe, New Mexico

NetGenics, Inc., Cleveland, Ohio

Acknowledgements

The editor of this document wishes to express his appreciation to those listed below (in alphabetic order) for their contributions of ideas and experience. Ultimately, the ideas expressed in this document are those of the authors and do not necessarily reflect the views or ideas of these individuals, nor does the inclusion of their names imply an endorsement of the final product.

Doug Bassett	Rosetta Inpharmatics
Derek Bernhart	Affymetrix
Alvis Brazma	EBI
Steve Chervitz	Affymetrix
Francisco De La Vega	Applied Biosystems
Michael Dickson	NetGenics
David Frankel	IONA
Ken Griffiths	NetGenics

Scott Markel	NetGenics
Alan Robinson	EBI
Ugis Sarkans	EBI
Martin Senger	EBI
Paul Spellman	UC Berkeley
Jason Stewart	Open Informatics
Charles Troup	Agilent

The editor would also like to express his appreciation to those who participated in the MGED programming efforts, the first sponsored by Iobion and Affymetrix, that took place in Toronto September 14-18, 2001 and the second sponsored by EBI, that took place in Hinxton December 6-11, 2001, and also to those who contributed ideas.

Bill Andreopoulos	University of Toronto
Eric Deutsch	Institute for Systems Biology
Jason Goncalves	Iobion
Robert Hubley	Institute for Systems Biology
Daniel Iordan	Iobion
Marc Lepage	Molecular Mining
W.L. Marks	Iobion
Todd Peterson	NCGR
Angel Pizarro	University of Pennsylvania
Marcin Swiatek	Imaging Research
Joe White	The Institute for Genomic Research
John Yost	National Cancer Institute

Proof of Concept

EBI Proof of Concept

European Bioinformatics Institute has been working on developing a public repository for gene expression data (ArrayExpress) since 1999 (see http://www.ebi.ac.uk/arrayexpress/). ArrayExpress development has been centered on ArrayExpress Object Model (AEOM), and design of MAGE-OM has been influenced by this experience.

AEOM was mapped to relational tables and implemented as a Oracle 8i database. A data loader from MAML (XML format that was a part of the initial Gene Expression RFP Response from EBI on behalf of MGED) to ArrayExpress was implemented, and some data sets were loaded into the repository. A prototype web interface is being developed. We do not foresee any obstacles in adopting MAGE-ML format as a data input format for ArrayExpress. We will also use MAGE-OM as a basis for ArrayExpress API development.

NCGR Proof of Concept

The National Center for Genome Resources has been developing an OpenSource gene expression database resource, GeneX, since 1999 (see http://genex.ncgr.org/). The GeneX project has focused on the development of a relational data model and a corresponding XML data-transmission model, GeneXML (see http://www.ncgr.org/genex/genexml.html). NCGR participated in the MGED MAML submission and many of the features of GeneXML were included in MAML.

GeneX comes with fully functional WWW interface to the DB, a direct export function to transmit data in GeneXML, and an importer to load GeneXML into the DB. We do not foresee any obstacles in adopting MAGE-ML format as a data input format for the GeneX project. The GeneX project intends to switch to the use of MAGE-OM as its high level data model, and to the use of MAGE-ML as its data transport format for the next major release (GeneX-1.2).

Rosetta Inpharmatics Proof of Concept

Rosetta Inpharmatics and Agilent Technologies have been using the GEML™ 1.0 format as part of internal pipelines for the past year and a half. The experience gained has influenced the development of MAGE-OM and MAGE-ML.

Rosetta has been continuously loading XML files on the order of thirteen megabytes into the Rosetta Resolver™ system, an enterprise expression data analysis product. We used internal tools to export the more than one thousand profiles, assigned annotations, and supporting patterns that constituted the data for the article, *Functional Discovery via a Compendium of Expression Profiles*, that appeared in the July 7, 2000 issue of *Cell*. The total size of the export, when compressed, was a little over a half of gigabyte of data. That data was then imported by Harvard into their Rosetta Resolver system.

In support of the publication of *Experimental Annotation of the Human Genome Using Microarray Technology*, that appeared in *Nature*, vol. 409, the results for Chromosome 22 were published in GEML 1.0 format and for the rest in GEML 2.0 format at http://download.rii.com/tech/pubs/nature.

Rosetta has developed the freeware GEML 1.0 Conductor™ tools for visualization of GEML 1.0 formatted data and for conversion of gene expression data in other formats into GEML.

With the release of Rosetta Resolver 2.0, support for GEML 2.0 (based on the initial Rosetta Inpharmatics response to the Gene Expression RFP) was implemented for the conversion of Intensity-based microarray hybridizations.

We have not, as of yet, implemented the data model contained in this proposal but we see no technical problems in migrating from the GEML format to MAGE.

MGED Open Source Efforts

In order to facilitate the acceptance of the MAGE format, MGED is driving an effort to develop open source tools that will make it substantially easier for organizations to adopt MAGE. There have been two get togethers dedicated to developing code, the first in Toronto in September and the next in December at the EBI site in Cambridge,

UK. At the Toronto get together the ground work was laid for using the UML model to generate much of the support code from the XMI representation of that model. The initial code to generate the data objects from the model in Java, Perl and C++ was completed and, shortly after the meeting, code was completed to generate the DTD from the model.

The ultimate goals of this effort will be to generate the data objects from the UML model along with code that can parse from XML based on the MAGE format to and from the other platforms. This was essentially completed for the Perl code at EBI and was partially completed for the Java code. It is also desired that Web-based forms can be generated to aid in entering the annotation for a given Gene Expression experiment and to associate the data with those annotations. A reference implementation for generating a schema based on the model is also planned.

The current state of the generating code can be obtained from: http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/mged/. It is distributed under the MIT Open Source License.

1 Introduction

1.1 General Remarks

This document contains a proposal for a standard that addresses the representation of gene expression data and relevant annotations, as well as mechanisms for exchanging these data.

The field of gene expression experiments has several distinct technologies that a standard must include. These include single vs. dual channel experiments, cDNA vs. oligonucleotides. Because of these different technologies and different types of gene expression experiments, it is not expected that all aspects of the standard will be used by all organizations.

With the acceptance of XML Metadata Interchange as an OMG standard it is possible to specify a normative UML model using a tool such as Rational Rose that describes the data structures for Gene Expression.

In this document we describe this normative model, Microarray and Gene Experiment Object Model (MAGE-OM). It also describes the general mapping rules to a DTD that best captures the syntax and semantics from the UML model in a way that will ease its understanding and use in the Gene expression community. One area of the DTD is further transformed to provide efficient and flexible formats for the actual data. The DTD is generated from the model with the addition of the transformed representation of the gene expression data in the DTD. Because of the primary importance of the DTD as the useful output of this proposal, it is considered normative, rather than the specialized generating algorithm.

Given the massive amount of data associated with a single set of experiments, we feel that Extensible Markup Language (XML) is the best way to describe the data. The use of a Document Type Definition (DTD) allows a well-defined tag set, a vocabulary, to describe the domain of gene expression experiments. It also has the virtue of compressing very well so that files in an XML format compress to ten percent of their original size. XML is now widely accepted as a data exchange format across multiple platforms.

Organizations that request these XML streams can use freely available implementations of either of the W3C recommended DOM or the XML-DEV SAX parsing interfaces to create import and export applications. These import and export applications can be tailored for the specific needs of the organization without the need to burden the vocabulary of the XML with specifics of any organization's schema requirements.

This submittal includes the MAGE.xmi, dtc/2002-02-03 produced from Rational Rose Enterprise Edition, v2000.02.10 using the Unisys Rose UML tool, Version 1.3.2. The export uses the XMI 1.0 option. The MAGE-ML.dtd is included as dtc/2002-09-04. The Java generating code and support files, including examples, is included as dtc/2002-09-5. The Rational Rose mdl file is also included for reference as dtc/2002-09-06.

Although there are good, standard ways to specify queries both in terms of the object model (OQL) and the XML (XQuery, XPath), the submitters have decided to limit

the scope of this proposal to the underlying format of the data for interchange between organizations. An emphasis in the design, however, was placed on making the XML modular so that the results of a single experiment could be split into different, easily managed files and that there were appropriate attributes and associations to facilitate queries.

Although not described in the Object Management Architecture Guide at the time of the issuance of the Gene expression RFP, we feel that this representation using the UML model to derive the DTD from is appropriate for this proposal and is in the spirit of Section 4.9.1 of the Gene Expression RFP and the recent emphasis on UML for modeling,

> "The models and terminology in the *Object Management Architecture Guide* and *CORBA* should be used in your submission. Where you believe this is not appropriate, describe and provide a rationale for the models and terminology you believe OMG should use."

The preceding paragraphs are our rational.

A few design principles and patterns that we have used are outlined first.

1.1.1 Representing Relationships in XML

There is a need in the XML vocabulary that describes the gene expression data to link two elements together, for instance, a Reporter to its BioSequence or a Hybridization to its Labeled Prep. There are three standard ways that are generally used in XML.

1. Using ID and IDREF or IDREFS attributes, where the IDREF or IDREFS values of an element 'point' to the ID value of one or more related elements within the document.
2. Using an XPointer to refer to an element or attribute in the current document or in another.
3. Using defined relationships that are implemented by applications.

We have chosen the third method for two reasons:

1. In general, both the physical items of gene expression data (the chip, the labeled extracts, solvents, etc.) and the virtual (array pattern, biosequence cluster, etc.) are assigned unique identifiers within an organization or department of an organization. These identifiers can be used as the reference value and can be further disambiguated using syntax similar to the CosNaming::StringName or Uniform Resource Names.
2. By defining reference elements in the vocabulary, the reference, itself, can serve as a place holder for what it references and the actual data that is referenced does not need to be in the document. This allows the most freedom of inter- and intra-document references.

Where the first isn't true, the element and its reference can be provided in the same document and their reference resolved within the document. An implementation is free to take as much advantage of the second point to provide efficiency on download as desired.

The following naming convention is used:

The element **foo** with unambiguous identifier attribute **identifier** would have a reference element named **foo_ref** with an attribute **identifier**. In the case of multiple attributes defining an unambiguous identifier, the **identifier** would be formed using the rules below (Section 2.1.3) for creating a string from multiple components.

1.1.2 Representing Extended Annotations and Relationships in the Object Model

There are three abstract classes in MAGE-OM from which all the classes in the model derive from, **Extendable**, **Describable**, and **Identifiable**. Although there are many places where explicit annotation is provided for in the model (many classes have an association to **OntologyEntity** to associate a **type** with them, **BioSorce** has a **characteristics** association to **OntologyEntry** to describe genotype, phenotype, sex, etc.), not all annotations can be modeled directly. **Extendable** and **Describable** provide two levels for extended annotation. **Identifiable**, on the other hand, provides attributes **identifier** and **name** for those objects that can be referenced.

Identifiable derives from **Describable** and **Describable** derives from **Extendable**, so that **Identifiable** has the associations of **Describable** and **Describable** has the associations of **Extendable**.

Extendable

The **Extendable** abstract class has an association to **NameValueType** that allows Property Sets and other miscellaneous information to be attached to an instance of a class. This is of primary use within an organization or a collaboration to facilitate internal pipelines or information that is not central to gene expression data but is of use internally.

These **NameValueTypes** specifically can not be considered to contain compliant information and the document should be equally useful for information germane to gene expression data if these values are ignored.

Describable

Describable allows associations to classes for annotation that may be useful in understanding the gene expression data and to provide rudimentary tracking and security permission associations. The association to **Description** provides further associations to specify external sources, database and bibliographic references, and ontology entries.

The association to **Audit** allows tracking changes and the association to **Security** allows indicating what groups should have rights to view or change the data. **Security** is not meant to provide security for the document itself, it is assumed that the organizations exchanging information in the MAGE-ML format use as secure a method as desired. The attributes and elements of **Security** are intended simply to be used by a repository or in an organization's internal data pipeline to provide the permissions for the data once it is saved, e.g. in a database.

Identifiable

Identifiable objects have attributes **identifier** and **name**. The criteria for being **Identifiable** is that in some association the class is contained by reference. Because of the inheritance from Identifiable, these classes are then transformed to the DTD with an associated reference element (as in Section 1.1.1) with the **identifier** being composed of the alternative key. The **name** attribute is for a common name which may not be unambiguous.

The intended scope of the **identifier** is important and will determine the disambiguating rules, an in-house application may require nothing more than common conventions for naming where as to store the data in a repository may take disambiguating rules between identifiers of two organizations.

*1.1.3 Use of **CosNaming::StringName** or **Uniform Resource Names***

The **Identifiable.identifier** attribute in the MAGE-OM, that is inherited by many classes, can be represented as a qualified string. We have found that using a convention similar to that described in the Interoperable Naming service, with the additional constraints below, provides a good naming system for the above. The following is from the Bibliographic Query Service (formal/02-05-03) which we will suggest to form identifiers.

> "To allow strings, yet mitigate their potential for abuse, this standard (and indeed some other LSR standards [BSA], [MAPS]) uses the syntax convention of **CosNaming::StringName** as described in the Interoperable Naming service [INS]. This convention is mainly a syntactical one; in no way is the use of a naming service implementation required or implied (but it is not precluded either).
>
> A brief description of **CosNaming::StringName** is as follows. **CosNaming::Name** is a list of **struct NameComponent**s. For the purpose of illustration, a **NameComponent** can be likened to a directory or filename, whereas **CosNaming::Name** constitutes a full path-name. The **struct NameComponent** has string members **id** and **kind**. To transform a **CosNaming::Name** into a string, all its **NameComponent**s are represented as strings " id.kind". If the **kind**-field is empty, this becomes simply " id"; if the **id**-field is empty, this becomes ".kind"; finally, the Naming service also allows both the **id**- and **kind**-fields to be empty, which is represented as ".". The full *stringified* **CosNaming::Name** is then obtained by concatenating all the **NameComponent**s using "/" as a separator character. The character "\" is designated as an escape character; if it precedes any of the special characters ".", "/" and "\", these special characters are taken as literal characters."

Typically the first name component can be the organization name then any inner name components can have as **ids** the departments within the organization, if needed, and the final name component **ids** would be the identifier (which may consist of multiple fields) within the organization.

A similar, alternative strategy may be the use of **Uniform Resource Names** as specified by The Internet Engineering Task Force in a variety of RFCs beginning with RFC 1737 which defines the requirements and the latest Working Group which intends to "...define the framework for URNs, at least one resolution registry system,

and at least one namespace. RFCs describing additional material will also be developed...".

1.1.4 OntologyEntry

The **OntologyEntry** class and the associations to it are to allow entries in the XML from ontologies or, the simplest case of an ontology, controlled vocabularies. A good example is the **Characteristics** association between **BioSource** and **OntologyEntry**. Different ontologies have been developed and are being developed, for instance, that are species dependent.

MAGE does not mandate the use of any particular Ontologies or Controlled vocabularies (except where an enumeration is specified as the datatype of an attribute). Instead it allows terms from an Ontology to be used with the ability to specify the name of the Ontology and, through the database reference association, where information and documentation for the Ontology can be located. This will allow new Ontologies to be used in MAGE compliant documents as they become available.

As the ontologies associated with gene expression experiments mature, it is expected that future versions of this specification will mandate or recommend that specific ontologies be used for particular associations.

1.1.5 Number Formats

In order to foster interoperability, it is necessary to agree on a common lexicographical representation of numbers in valid MAGE-ML XML documents.

Since the W3C recommendation, "XML Schema Part 2: Datatypes", contains definitions for such representations, they are the recommended form. Below are the definitions from that specification:"3.2.2 boolean

...

3.2.2.1 Lexical representation

An instance of a datatype that is defined as · boolean· can have the following legal literals {true, false, 1, 0}."

"3.2.3 decimal

...

NOTE: All · minimally conforming· processors · must· support decimal numbers with a minimum of 18 decimal digits (i.e., with a · totalDigits· of 18). However, minimally conforming· processors · may· set an application-defined limit on the maximum number of decimal digits they are prepared to support, in which case that application-defined maximum number · must· be clearly documented.

3.2.3.1 Lexical representation

decimal has a lexical representation consisting of a finite-length sequence of decimal digits (#x30-#x39) separated by a period as a decimal indicator. If · totalDigits· is specified, the number of digits must be less than or equal to · totalDigits· . If · fractionDigits· is specified, the number of digits following the decimal point must be

less than or equal to the · fractionDigits· . An optional leading sign is allowed. If the sign is omitted, "+" is assumed. Leading and trailing zeroes are optional. If the fractional part is zero, the period and following zero(es) can be omitted. For example: -1.23, 12678967.543233, +100000.00, 210."

(· totalDigits· would be documented for an attribute by the exporting organization of the XML)

"3.3.13 integer

...

3.3.13.1 Lexical representation

integer has a lexical representation consisting of a finite-length sequence of decimal digits (#x30-#x39) with an optional leading sign. If the sign is omitted, "+" is assumed. For example: -1, 0, 12678967543233, +100000."

"3.2.4 float

...

3.2.4.1 Lexical representation

float values have a lexical representation consisting of a mantissa followed, optionally, by the character "E" or "e", followed by an exponent. The exponent · must· be an integer. The mantissa must be a decimal number. The representations for exponent and mantissa must follow the lexical rules for integer and decimal. If the "E" or "e" and the following exponent are omitted, an exponent value of 0 is assumed.

The special values positive and negative zero, positive and negative infinity and not-a-number have lexical representations 0, -0, INF, -INF and NaN, respectively.

For example, -1E4, 1267.43233E12, 12.78e-2, 12 and INF are all legal literals for float."

"3.2.5 double

...

3.2.5.1 Lexical representation

double values have a lexical representation consisting of a mantissa followed, optionally, by the character "E" or "e", followed by an exponent. The exponent · must· be an integer. The mantissa must be a decimal number. The representations for exponent and mantissa must follow the lexical rules for integer and decimal. If the "E" or "e" and the following exponent are omitted, an exponent value of 0 is assumed.

The special values positive and negative zero, positive and negative infinity and not-a-number have lexical representations 0, -0, INF, -INF and NaN, respectively.

For example, -1E4, 1267.43233E12, 12.78e-2, 12 and INF are all legal literals for double."

1.1.6 Dates

There are several attributes that represent dates. The suggested encoding is carried over from the Bibliographic Query Service and is as defined in a W3C NOTE “Date and Time Formats”. This NOTE defines a profile of ISO8601 standard. The NOTE reduces the scope and restricts the supported formats to a small number. The profile offers a number of options from which this specification permits the following ones:

- Year

  YYYY (e.g. 2000)

- Year and month

  YYYY-MM (e.g. 2000-12)

- Complete date

  YYYY-MM-DD (e.g. 2000-12-31)

- Complete date plus hours, minutes and seconds

  YYYY-MM-DDThh:mm:ssZ (e.g. 2000-12-31T23:59:59Z)

Where

YYYY	four-digit year
MM	two-digit month (01=January, etc.)
DD	two-digit day of month (01 through 31)
hh	two digits of hour (00 through 23)
mm	two digits of minute (00 through 59)
ss	two digits of second (00 through 59)

Exactly the components shown here must be present, with exactly this punctuation. Note that the "T" appears literally in the string, to indicate the beginning of the time element, as specified in ISO 8601.

Times are expressed in UTC (Coordinated Universal Time), with a special UTC designator ("Z"), again as specified in ISO 8601.

1.1.7 Conventions

For the Platform Independent Model, MAGE-OM, classes, associations, operations and attributes appear using this font when used inline: **ClassName**. Individual class descriptions use the following pattern.

ClassName

Class comment.

Derived from ParentClass

Methods:

methodOne() : returnType

method comment
Parameters:
paramOne : paramType
paramTwo : paramType

methodTwo() : returnType

method comment
Parameters:
paramOne : paramType

Associations:

associationOne : endClass (0..n)
association comment

associationTwo : endClass (1..1)
association comment

Attributes:

attributeOne : attributeType (optional)
attribute comment

attributeTwo : attributeType (required)
enumeration {*choice1* | *choice2* | *choice3* }
attribute comment

attributeThree : attributeType (default: *choice2*)
enumeration {*choice1* | *choice2* | *choice3* }
attribute comment

1.1.8 Documentation of MAGE-OM

The documentation for MAGE-OM in Section 2 is generated from the UML model. It has been reformatted for this document but is otherwise the same.

1.2 Overview

The structure of this submission has been driven by the definition of a UML model to represent the data structures for Gene Expression data interchange. The UML model is presented in order to best capture the domain of Gene Expression Data independent of any implementation. The model, then, is mapped to a DTD by a set of rules, which include, in the representation of the gene expression data in the XML, a further transformation. These rules have been encoded to generate the MAGE-ML.dtd from the XMI 1.0 representation of the model.

1.2.1 Domain Model MicroArray and Gene Expression Object Model (MAGE-OM)

MAGE-OM specifically attempts to define the objects of Gene Expression data independent of any implementation. Further, the submitters felt it important to try to abstract the ideas so that the model might be applicable to a broader set of array style

experiments. Rather than use **Hybridization,** the general class is **BioAssay** of which **Hybridization** is a subclass of **BioAssayCreation**.

MAGE-OM can be used to map to data structures in different platforms, such as Java, Perl or C++. The final form of the **BioSequence** package will be dependent on the *Biomolecular Sequence Analysis Entities (BSANE)* (lifesci/01-03-04) which is currently in the revision process, and the **BibliographicReference** class is intended to provide criteria for queries to implementations of the *Bibliographic Query Service* (formal/02-05-03) or other bibliographic resources.

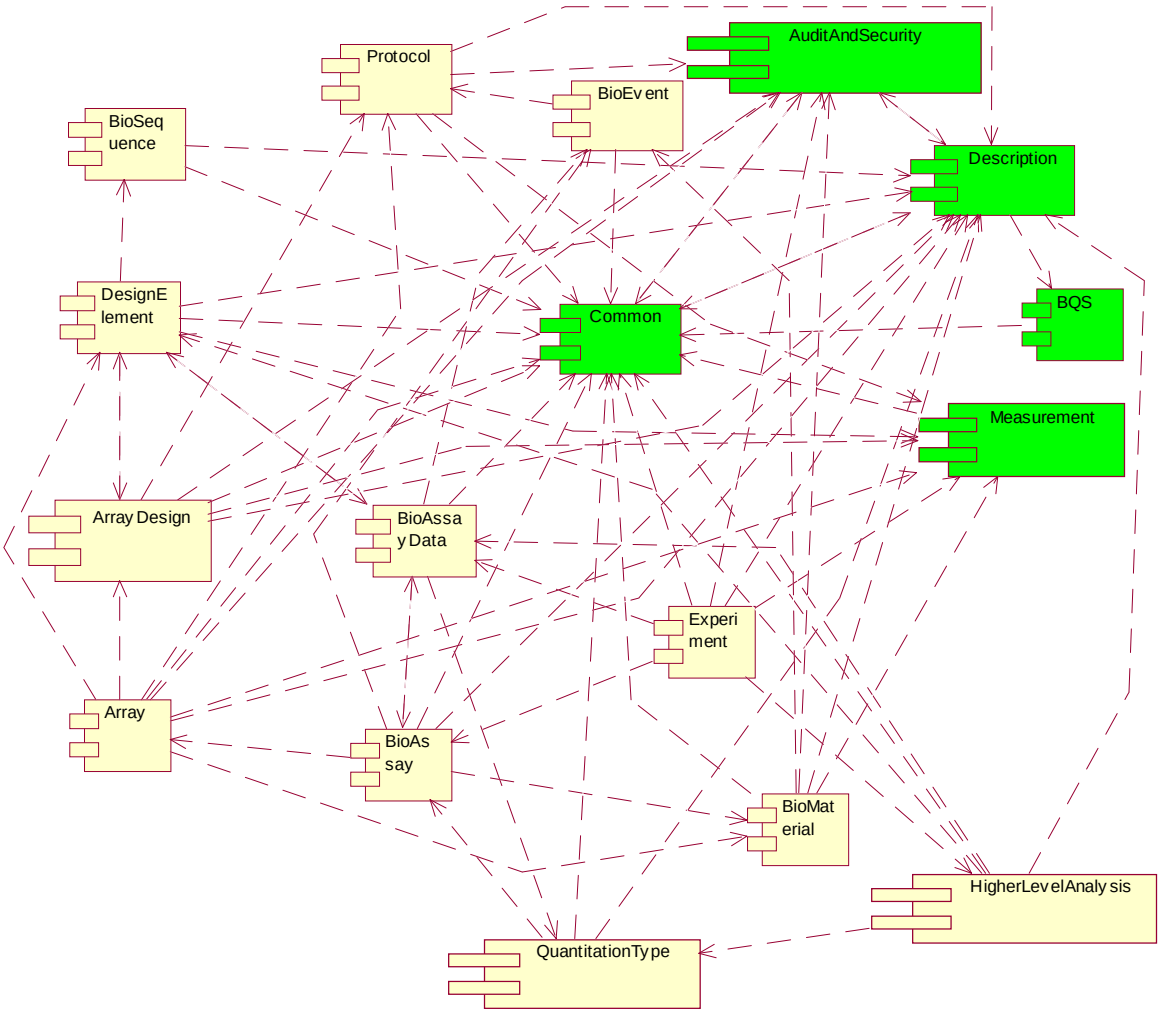

Figure 1: Relationship of MAGE-OM Packages

Figure 1 shows the relationships of the different packages in MAGE-OM. The packages correspond to the natural separation of events and objects of Gene Expression data along with other more generic packages, such as **Common** and **Description**, which include classes for annotation.

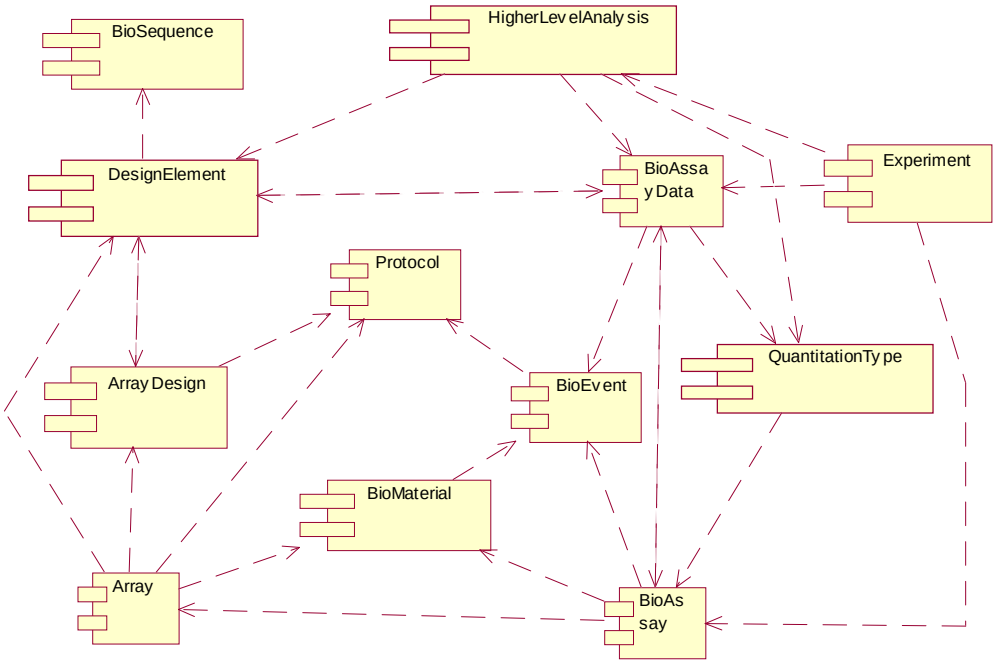

Figure 2: Main packages

Figure 2 gives a clearer view of the relationship between the main packages, leaving out the packages specific for annotation.

Packages:

- **BioSequence**--Specifies classes that describe the sequence information for a **BioSequence**.
- **QuantitationType**--This package defines the classes for quantitations, such as measured and derived signal, error, and pvalue.
- **ArrayDesign**--Describes a microarray design that can be printed and then, in the case of gene expression, hybridized.
- **DesignElement**--The classes of this package are the contained and referenced classes of the **ArrayDesign** and describe through the **DesignElements** what is intended to be at each location of the **Array**.
- **Array**--Describes the process of creating arrays from array designs.
- **BioMaterial**-- Specifies classes that describe how a **BioSource** is treated to obtain the **BioMaterial** (typically a **LabeledExtract**) used to create a **BioAssay**.
- **BioAssay**--Specifies classes that contain information and annotation on the event of joining an **Array** with a **BioMaterial** preparation, the acquisition of images and the extraction of data on a per feature basis from those images.
- **BioAssayData**—Specifies classes that describe the data and information and annotation on the derivation of that data.
- **Experiment**--Represents the container for a hierarchical grouping of **BioAssays**.

- **HigherLevelAnalysis**--Describes the results of performing analysis on the result of the **BioAssayData** from an **Experiment**.
- **Protocol**--Provides a relatively immutable class, **Protocol**, that can describe a generic laboratory procedure or analysis algorithm, for example, and an instance class, **ProtocolApplication**, which can describe the actual application of a protocol.
- **Description**--The classes in this package allow a variety of references to third party annotation and direct annotation by the experimenter.
- **AuditAndSecurity**--Specifies classes that allow tracking of changes and information on user permissions.
- **Measurement**--The classes of this package provide utility information on the quantities of other classes to each other.
- **BioEvent**--An abstract class representing an event that takes sources of some type to produce a target(s) of some type.

1.2.2 Domain Model MAGE-ML

A Document Type Definition (DTD) is generated from the MAGE-OM by an implementation of a well-described set of mapping rules (See Section 3). This DTD file, MAGE-ML.dtd will have one additional, specified transformation for the representation of the gene expression data that maps the classes representing the gene expression data in MAGE-OM into a transformed set of elements that should allow more efficiency and alternative representation for these data. Because the transformation for the DTD is well-formed mapping, for example, a Java implementation to the DTD poses no problems.

Through the mapping rules, the documentation for elements of the DTD are equivalent to their source in the Object Model and are not further specified in this document. The DTD that is generated includes that documentation as comments.

1.3 Relationship to OMG Specifications

1.3.1 Model Driven Architecture

While the model provides only the data structures needed for the interchange of Gene Expression data and specifies no services, it does share the MDA approach of using the UML model as the basis of generating from the XMI the DTD. Although not part of this proposal, there is also an effort underway that has used the model to generate Java, Perl, and C++ versions of the data structures.

1.3.2 XML Metadata Interchange

The MAGE-OM model will be submitted in XMI format, v1.0, generated from Ration Rose Enterprise edition using the Unisys JCRUML 1.3.2 addin. Although XMI specifies a mapping from a model to a DTD, the state of such implementations is still immature. In addition, the simplicity of the model, in that it is data-centric, and the desire to use reference elements, convinced the submitters to specify a set of

mapping rules that tailored the DTD for the target user community. A more complete discussion can be found in Section 3.1.2

1.3.3 Query Service

The Query Service can be implemented to provide access to XML documents in MAGE-ML format though the submitters considered it out of the scope of this specification to submit a specialized Query Service for Gene Expression Data.

1.3.4 Biomolecular Sequence Analysis and BSANE

Although the BSA and BSANE emerging standard is not directly incorporated, attributes and annotation from associations of the **BioSequence** can be used in queries to data sources that support the interfaces for these specifications.

1.3.5 Bibliographic Query Service

Although the BQS specification is not directly incorporated, the attributes and annotation from associations of the **BibliographicReference** can be used in queries to data sources that support the interfaces in the BQS specification.

1.3.6 Collection Service

The Collection Service is not used. Instead, the W3C DOM and XML-DEV SAX parsers provide similar capabilities for XML data as the Collection Service for IDL data. The DOM parser is ideal for smaller XML data and provides full navigation backwards and forwards. The SAX parser provides a forward only traversal of the data, which is ideal for parsing large XML documents.

1.4 Compliance Points

For a document to be compliant with the MAGE-ML.dtd it must be valid as defined by the W3C Recommendation: *Extensible Markup Language (XML) 1.0*. That is, a parser that conforms to those rules will find the document valid

1.5 Intended Audience and Use

There is a desire that organizations can exchange Gene Expression experiments. One very important exchange is between an organization and a journal for purposes of publication and review. There is also a desire for creating public repositories to share Gene Expression experiments. There has been considerable work done in stating what information should be included in a set of documents that would be considered a complete set of documents for reporting a Gene Expression experiment. The RFP, itself, specified several criteria and MGED has gone even further in their document, *Minimum Information About a Microarray Experiment* (MIAME). The design of MAGE-OM and also as reflected in MAGE-ML was predicated on the fact that the model and DTD must allow that information to be captured.

As a guide to users, Section 2.3, **Response to RFP Requirements**, in previous submittal, lifesci/02-01-01 details how a set of documents can fulfil the information considered germane by the RFP document and Appendix B, **Minimum Information About a Microarray Experiment**, similarly details how a set of documents can be considered MIAME compliant for reporting a Gene Expression experiment.

It is also expected that organizations can use portions of the MAGE-ML format to facilitate in-house pipelines. For instance, as array designs are created, they can be stored then exported as XML to the arrayer which can automatically use the XML as a guide to creating the arrays. It allows the hybridization of an array with biomaterial and then its automated scanning (again, based on the array design) to be recorded in separate XML documents then joined later in the organization's data store. The process of creating and hybridizing the 1000+ arrays that constitute a Gene Expression experiment can take days and even months but once that process is completed and the high level annotation of the experiment done through either web interfaces or software tools, then the experiment can be exported as a whole.

Collaborations between organizations can also benefit by sharing incomplete pieces of an experiment as they become available. A company that specializes in producing arrays, for instance, can ship the arrays with the array design to its customers and the customers can carry on without further need of communication with the array company. An organization can divide an experiment between several different laboratories and then gather the work together at the end without the labs needing to interact with each other.

2 Platform Independent Model

2.1 MicroArray and GeneExpression Object Model (MAGE-OM)

MAGE-OM was developed with the purpose of capturing the objects relevant for MicroArray experiments, specifically for capturing the data and annotation of Gene

Figure 3: Workflow

Expression Experiments. There is also an inherent workflow to these experiments which drove the development of the separate packages.

MAGE-OM is a framework for describing experiments done on all types of DNA-arrays, including spotted and synthesized arrays, and oligo-nucleotide and cDNA arrays. It is independent of the particular image analysis and data normalization methods, and allows representation of both raw and processed microarray data. Above the representation of expression measurements, it allows for comprehensive annotation of experimental results.

The decision was made, however, to make the model general enough so that other technologies, such as proteomics, could potentially re-use this model. So, for instance, instead of referring directly to Hybridization, the general class is specified as **BioAssayCreation**.

2.1.1 Logical View

Classes directly in the logical view are generic abstract classes intended to be subclassed for generic annotation and for attributes to be used as identifiers for instances of a class.

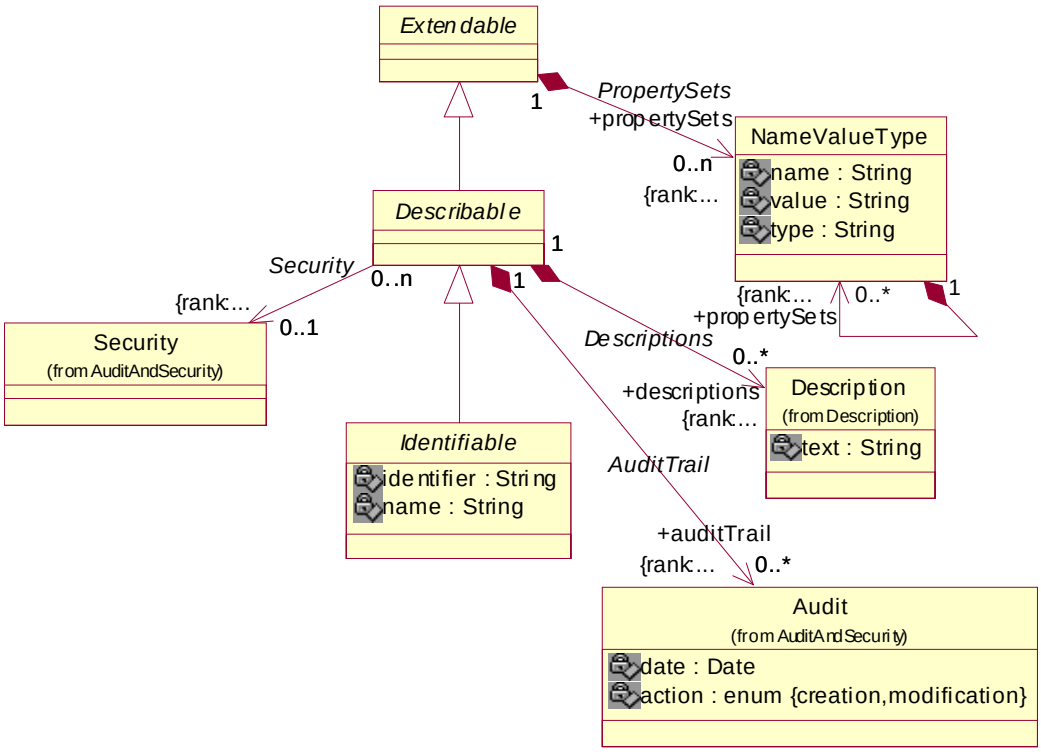

Figure 4: Logical View

Extendable

Abstract class that specifies for subclasses an association to **NameValueTypes**. These can be used, for instance, to specify proprietary properties and in-house processing hints.

Associations:

propertySets : NameValueType (0..n)
Allows specification of name/value pairs. Meant to primarily help in-house, pipeline processing of instances by providing a place for values that aren't part of the specification proper.

Describable

Abstract class that allows subclasses to inherit the association to **Description**, for detailed annotations such as **OntologyEntries** and **Database** references, the association to **Audit**, for tracking changes, and the association to **Security** for indicating permissions.

Derived from Extendable

Associations:

security : Security (0..1)
Information on the security for the instance of the class.

auditTrail : Audit (0..n)
A list of **Audit** instances that track changes to the instance of **Describable**.

descriptions : Description (0..n)
Free hand text descriptions. Makes available the associations of **Description** to an instance of **Describable**.

Identifiable

An **Identifiable** class is one that has an unambiguous reference within the scope. It also has a potentially ambiguous name.

Derived from Describable

Attributes:

identifier : String (required)
An identifier is an unambiguous string that is unique within the scope (i.e. a document, a set of related documents, or a repository) of its use.

name : String (optional)
The potentially ambiguous common identifier.

NameValueType

A tuple designed to store data, keyed by a name and type.

Associations:

propertySets : NameValueType (0..n)

Allows nested specification of name/value pairs

Attributes:

name : String (optional)
The name of the key.

value : String (optional)
The value of the name.

type : String (optional)
The type of the key.

2.1.2 AuditAndSecurity

Specifies classes that allow tracking of changes and information on user permissions to view the data and annotation.

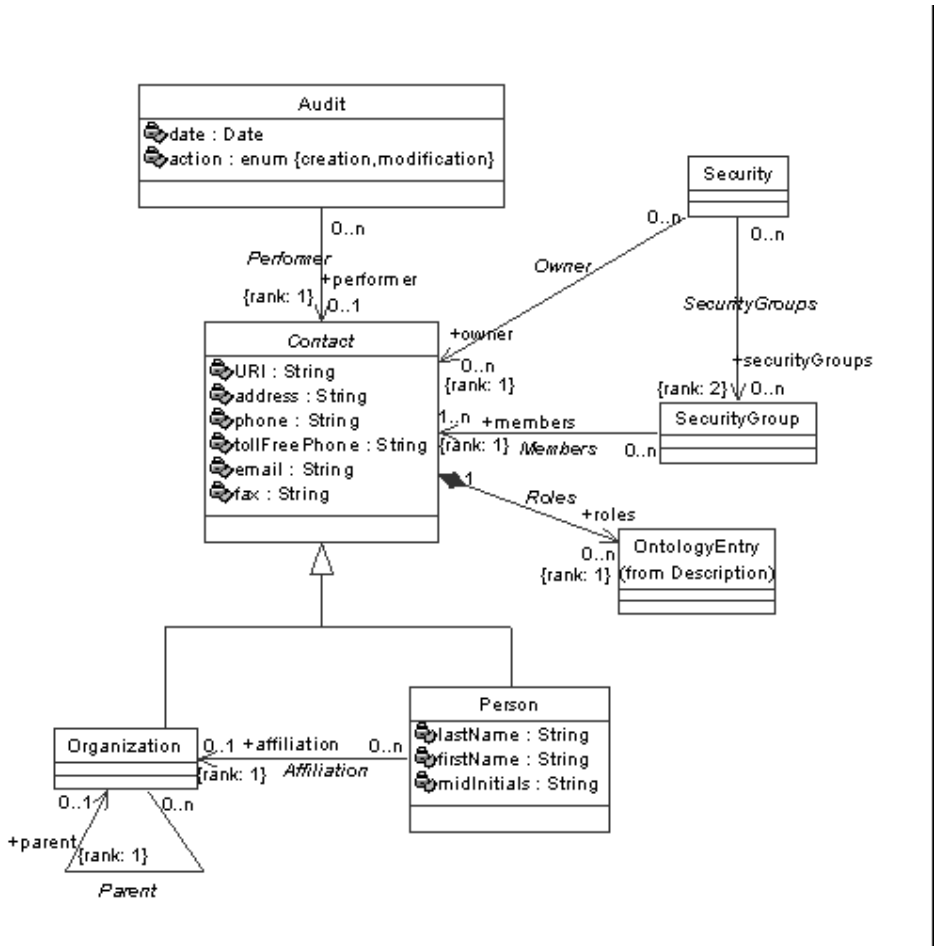

Figure 5: Audit and Security

Audit

Tracks information on the contact that creates or modifies an object.

Derived from Describable

Associations:

performer : Contact (0..1)
The contact for creating or changing the instance referred to by the **Audit**.

Attributes:

date : Date (required)
The date of a change.

action : Action (required)
enumeration {*creation* | *modification*}
Indicates whether an action is a creation or a modification.

Contact

A contact is either a person or an organization.

Derived from Identifiable

Associations:

roles : OntologyEntry (0..n)
The roles (lab equipment sales, contractor, etc.) the contact fills.

Attributes:

URI : String (optional)

address : String (optional)

phone : String (optional)

tollFreePhone : String (optional)

email : String (optional)

fax : String (optional)

Person

A person for which the attributes are self describing.

Derived from Contact

Associations:

affiliation : Organization (0..1)
The organization a person belongs to.

Attributes:

lastName : String (optional)

firstName : String (optional)

midInitials : String (optional)

Organization

Organizations are entities like companies, universities, government agencies for which the attributes are self describing.

Derived from Contact

Associations:

parent : Organization (0..1)
The containing organization (the university or business which a lab belongs to, etc.)

Security

Permission information for an object as to ownership, write and read permissions.

Derived from Identifiable

Associations:

securityGroups : SecurityGroup (0..n)
Specifies which security groups have permission to view the associated object.

owner : Contact (0..n)
The owner of the security rights.

SecurityGroup

Groups contacts together based on their security privileges.

Derived from Identifiable

Associations:

members : Contact (1..n)
The members of the Security Group.

2.1.3 Description

The classes in this package allow a variety of references to third party annotation and direct annotation by the experimenter.

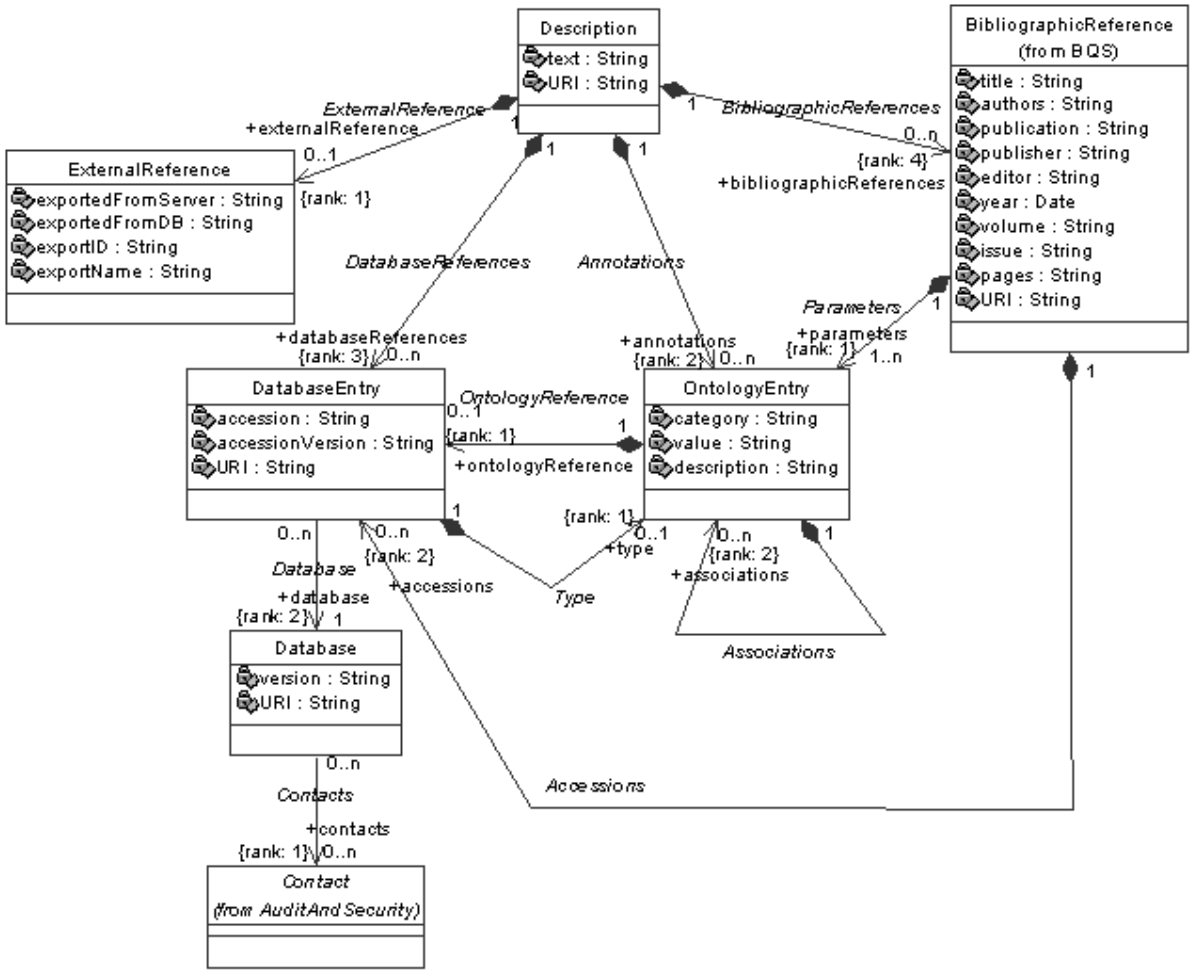

Figure 6: Description

Description

A free text description of an object.

Derived from Describable

Associations:

databaseReferences : DatabaseEntry (0..n)
References to entries in databases.

bibliographicReferences : BibliographicReference (0..n)
References to existing literature.

externalReference : ExternalReference (0..1)
Specifies where the described instance was originally obtained from.

annotations : OntologyEntry (0..n)
Allows specification of ontology entries related to the instance being described.

Attributes:

text : String (optional)
The description.

URI : String (optional)
A reference to the location and type of an outside resource.

Database

An address to a repository.

Derived from Identifiable

Associations:

contacts : Contact (0..n)
Information on the contacts for the database

Attributes:

version : String (optional)
The version for which a **DatabaseReference** applies.

URI : String (optional)
The location of the **Database**.

DatabaseEntry

A reference to a record in a database.

Derived from Extendable

Associations:

database : Database (1..1)
Reference to the database where the **DatabaseEntry** instance can be found.

type : OntologyEntry (0..1)
The type of record (e.g. a protein in SwissProt, or a yeast strain in SGD).

Attributes:

accession : String (required)
The identifier used to look up the record.

accessionVersion : String (optional)
The appropriate version of the accession (if applicable).

URI : String (optional)
The location of the record.

ExternalReference

A reference to the originating source for the object.

Derived from Extendable

Attributes:

exportedFromServer : String (optional)
The originating server for the object, a network address or commo n name.

exportedFromDB : String (optional)
Name of the database, if applicable, that the object was exported from.

exportID : String (optional)
The identifier of the object at the originating source.

exportName : String (optional)
The name of the object at the originating source.

OntologyEntry

A single entry from an ontology or a controlled vocabulary. For instance, category could be 'species name', value could be 'homo sapiens' and ontology would be taxonomy database, NCBI.

Derived from Extendable

Associations:

ontologyReference : DatabaseEntry (0..1)
Many ontology entries will not yet have formalized ontologies. In those cases, they will not have a database reference to the ontology.

In the future it is highly encouraged that these ontologies be developed and **OntologyEntry** be subclassed from **DatabaseReference**.

associations : OntologyEntry (0..n)
Allows an instance of an OntologyEntry to be further qualified.

Attributes:

category : String (required)
The category to which this entry belongs.

value : String (required)
The value for this entry in this category.

description : String (optional)
The description of the meaning for this entry.

2.1.4 BQS

Allows a reference to an article, book or other publication to be specified for searching repositories.

BibliographicReference

Attributes for the most common criteria and association with **OntologyEntry** allows criteria to be specified for searching for a Bibliographic reference.

Derived from Describable

Associations:

parameters : OntologyEntry (1..n)
Criteria that can be used to look up the reference in a repository.

accessions : DatabaseEntry (0..n)
References in publications, eg Medline and PubMed, for this **BibliographicReference**.

Attributes:

title : String (optional)

authors : String (optional)

publication : String (optional)

publisher : String (optional)

editor : String (optional)

year : Date (optional)

volume : String (optional)

issue : String (optional)

pages : String (optional)

URI : String (optional)

2.1.5 BioSequence

Describes a known gene or sequence. **BioAssays** typically seek to identify what **BioSequences** are expressed in a **BioMaterial** after treatments, the expression level measured from the association between the **BioMaterial** and the **Array**. The **Array's Features** typically provide known locations for this association to occur. Most often, the **Reporter** and **CompositeSequence** are known and the presence or absence of a particular **BioSequence** in the **BioMaterial** is based on whether there as been an association to the **DesignElement** targeted for it. Some other experiments may not know the **DesignElement's** target but can discover it with known properties of the **BioSequences** in the **BioMaterial**.

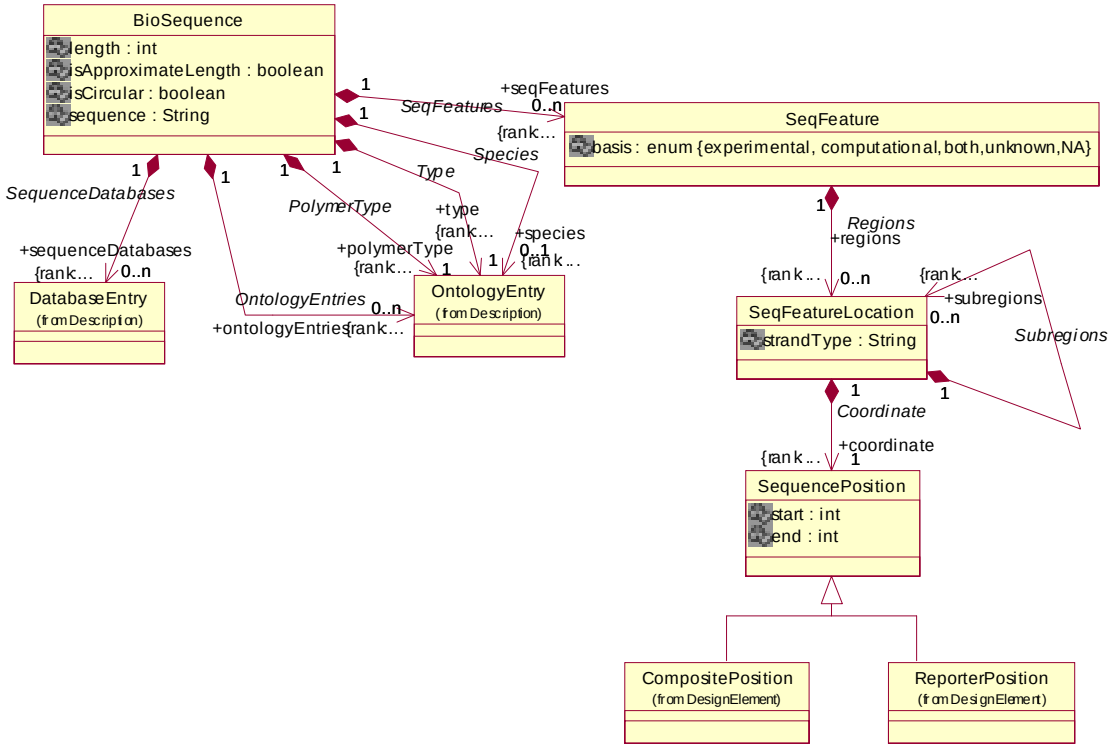

Figure 7: BioSequence

BioSequence

A **BioSequence** is a representation of a DNA, RNA, or protein sequence. It can be represented by a Clone, Gene, or the sequence.

Derived from Identifiable

Associations:

sequenceDatabases : DatabaseEntry (0..n)
References an entry in a species database, like GenBank, UniGene, etc.

ontologyEntries : OntologyEntry (0..n)
Ontology entries referring to common values associated with **BioSequences**, such as gene names, go ids, etc.

polymerType : OntologyEntry (1..1)
A choice of protein, RNA, or DNA.

type : OntologyEntry (1..1)
The type of biosequence, i.e. gene, exon, UniGene cluster, fragment, BAC, EST, etc.

species : OntologyEntry (0..1)
The organism from which this sequence was obtained.

seqFeatures : SeqFeature (0..n)
Association to annotations for subsequences. Corresponds to the GenBank Feature Table.

Attributes:

length : int (optional)
The number of residues in the biosequence.

isApproximateLength : boolean (optional)
If length not positively known will be true

isCircular : boolean (optional)
Indicates if the BioSequence is circular in nature.

sequence : String (optional)
The actual components of the sequence, for instance, for DNA a string consisting of A,T,C and G.

The attribute is optional and instead of specified here, can be found through the DatabaseEntry.

SeqFeature

Represents, in general, what would be a GenBank Feature Table annotation for a sequence.

Derived from Describable

Associations:

regions : SeqFeatureLocation (0..n)
Association to classes that describe the location with the sequence of the **SeqFeature**.

Attributes:

basis : Basis (required)
enumeration {*experimental* | *computational* | *both* | *unknown* | *NA*}
How the evidence for a **SeqFeature** was determined.

SeqFeatureLocation

The location of the **SeqFeature** annotation.

Derived from Extendable

Associations:

subregions : SeqFeatureLocation (0..n)
Regions within the **SeqFeature**.

coordinate : SequencePosition (1..1)
At which base pairs or amino acid this **SeqFeature** begins and ends.

Attributes:

strandType : String (required)

Indicates the direction and/or type of the **SeqFeature**, i.e. whether it is in the 5' or 3' direction, is double stranded, etc.

SequencePosition

Designates the position of the **Feature** in its **BioSequence**.

Derived from Extendable

Attributes:

start : int (optional)
The location of the base, for nucleotides, that the **SeqFeature** starts.

end : int (optional)
The location of the base, for nucleotides, that the **SeqFeature** ends.

2.1.6 ArrayDesign

Describes a microarray design that can be printed and then, in the case of gene expression, hybridized. An array design consists of several features (also called spots) in which reporter sequences are placed. Many features may have the same reporter replicated and a reporter may be specified in one or more array designs.

The nature of the reporter's biosequence placed on a spot will depend on the technology. Two well-known technologies differ significantly-spotter arrays draw material from a well and place a spot on the array whereas in situ oligo arrays are created through the synthesis of many, short (~20-100mer) nucleotide sequences onto the features.

Reporters can be grouped together into **CompositeSequences**, typically representing a gene or one or more splice variants in gene expression experiments.

There are then two distinct ways that **DesignElements** are grouped. The one described in the **ArrayDesign** package by **FeatureGroup**, **ReporterGroup** and **CompositeGroup** is by technology type, that is, one might want to segregate the controls to a Group and all the non-controls to another. Or if PCR Product and Oligos are both used on an array they would likely be in different groups. The grouping described in the **DesignElement** package by the mappings relates the **Features** to the **Reporter**, the **Reporters** to **CompositeSequence**, and at higher levels, **CompositeSequences** to **CompositeSequence**.

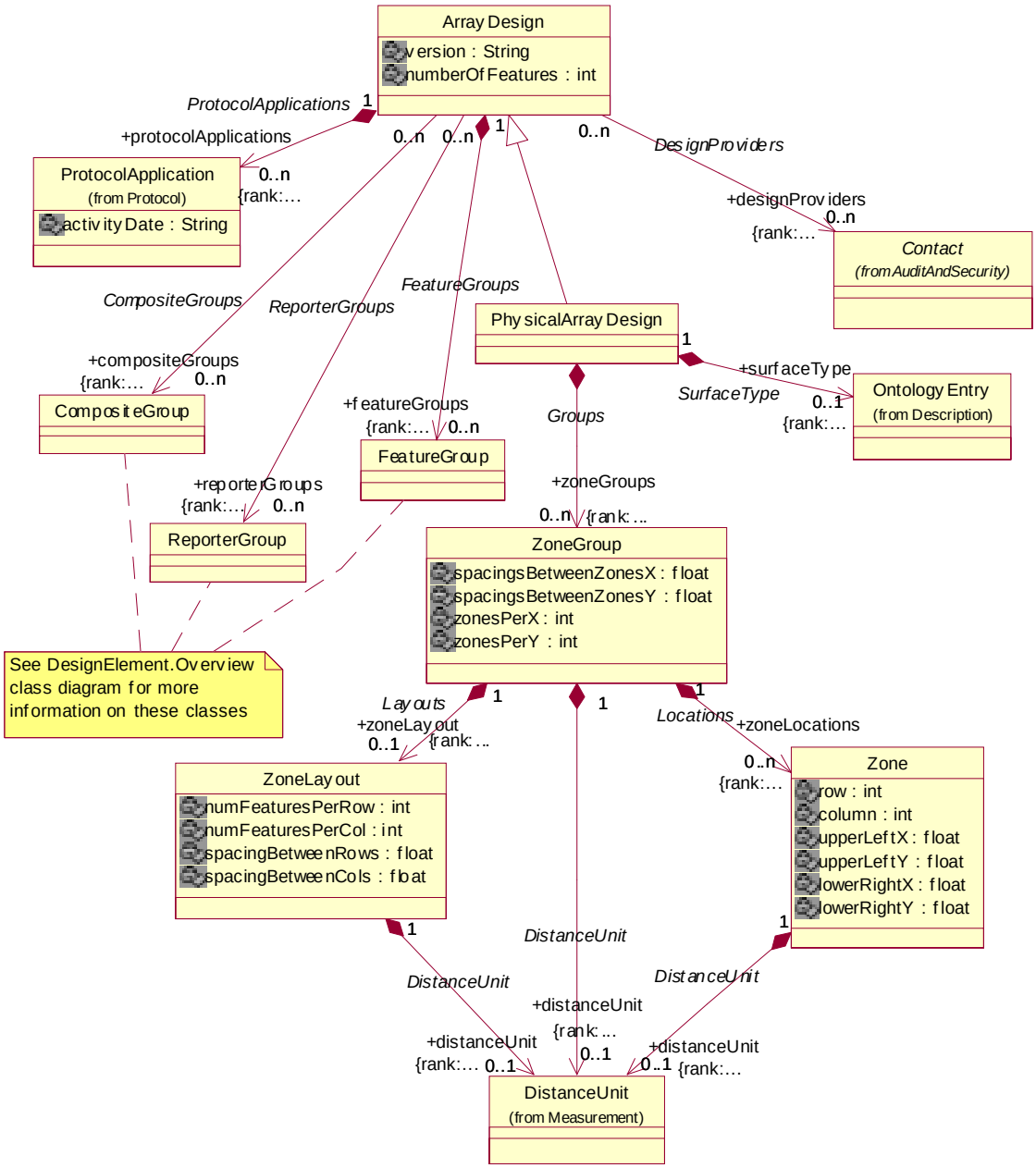

Figure 8: ArrayDesign

ArrayDesign

Describes the design of an gene expression layout. In some cases this might be virtual and, for instance, represent the output from analysis software at the composite level without reporters or features.

Derived from Identifiable

Associations:

protocolApplications : ProtocolApplication (0..n)
Describes the application of any protocols, such as the methodology used to pick oligos, in the design of the array.

compositeGroups : CompositeGroup (0..n)
The grouping of like **CompositeSequence** together. If more than one technology type occurs on the array, such as the mixing of Cloned BioMaterial and Oligos, then there would be multiple **CompositeGroups** to segregate the technology types.

designProviders : Contact (0..n)
The primary contact for information on the array design

reporterGroups : ReporterGroup (0..n)
The grouping of like **Reporter** together. If more than one technology type occurs on the array, such as the mixing of Cloned BioMaterial and Oligos, then there would be multiple **ReporterGroups** to segregate the technology types.

featureGroups : FeatureGroup (0..n)
The grouping of like **Features** together. Typically for a physical array design, this will be a single grouping of features whose type might be PCR Product or Oligo. If more than one technology type occurs on the array, such as the mixing of Cloned BioMaterial and Oligos, then there would be multiple **FeatureGroups** to segregate the technology types.

Attributes:

version : String (optional)
The version of this design.

numberOfFeatures : int (optional)
The number of features for this array

PhysicalArrayDesign

A design that is expected to be used to manufacture physical arrays.

Derived from ArrayDesign

Associations:

zoneGroups : ZoneGroup (0..n)
In the case where the array design is specified by one or more zones, allows specifying where those zones are located.

surfaceType : OntologyEntry (0..1)
The type of surface from a controlled vocabulary that would include terms such as non-absorptive, absorptive, etc.

DesignElementGroup

The **DesignElementGroup** holds information on either features, reporters, or compositeSequences, particularly that information that is common between all of the DesignElements contained.

Derived from Identifiable

Associations:

species : OntologyEntry (0..1)
The organism from which the biosequences of this group are from.

types : OntologyEntry (0..n)
The specific type of a feature, reporter, or composite. A composite type might be a gene while a reporter type might be a cDNA clone or an oligo.

CompositeGroup

Allows specification of the type of Composite Design Element.

Derived from DesignElementGroup

Associations:

compositeSequences : CompositeSequence (1..n)
The compositeSequences that belong to this group.

ReporterGroup

Allows specification of the type of Reporter Design Element.

Derived from DesignElementGroup

Associations:

reporters : Reporter (1..n)
The reporters that belong to this group.

FeatureGroup

A collection of like features.

Derived from DesignElementGroup

Associations:

distanceUnit : DistanceUnit (0..1)
The unit for the feature measures.

features : Feature (1..n)
The features that belong to this group.

technologyType : OntologyEntry (0..1)
The technology type of this design. By specifying a technology type, higher level analysis can use appropriate algorithms to compare the results from

multiple arrays. The technology type may be spotted cDNA or in situ photolithography.

featureShape : OntologyEntry (0..1)
The expected shape of the feature on the array: circular, oval, square, etc.

Attributes:

featureWidth : float (optional)
The width of the feature.

featureLength : float (optional)
The length of the feature.

featureHeight : float (optional)
The height of the feature.

ZoneGroup

Specifies a repeating area on an array. This is useful for printing when the same pattern is repeated in a regular fashion.

Derived from Extendable

Associations:

zoneLayout : ZoneLayout (0..1)
Describes the rectangular layout of features in the array design.

zoneLocations : Zone (0..n)
Describes the location of different zones within the array design.

distanceUnit : DistanceUnit (0..1)
Unit for the ZoneGroup attributes.

Attributes:

spacingsBetweenZonesX : float (optional)
Spacing between zones, if applicable.

spacingsBetweenZonesY : float (optional)
Spacing between zones, if applicable.

zonesPerX : int (optional)
The number of zones on the x-axis.

zonesPerY : int (optional)
The number of zones on the y-axis.

Zone

Specifies the location of a zone on an array.

Derived from Identifiable

Associations:

distanceUnit : DistanceUnit (0..1)
Unit for the Zone attributes.

Attributes:

row : int (optional)
row position in the **ZoneGroup**

column : int (optional)
column position in the **ZoneGroup**.

upperLeftX : float (optional)
Boundary vertical upper left position relative to (0,0).

upperLeftY : float (optional)
Boundary horizontal upper left position relative to (0,0).

lowerRightX : float (optional)
Boundary vertical lower right position relative to (0,0).

lowerRightY : float (optional)
Boundary horizontal lower right position relative to (0,0).

ZoneLayout

Specifies the layout of features in a rectangular grid.

Derived from Extendable

Associations:

distanceUnit : DistanceUnit (0..1)
Unit of the **ZoneLayout** attributes.

Attributes:

numFeaturesPerRow : int (optional)
The number of features from left to right.

numFeaturesPerCol : int (optional)
The number of features from top to bottom of the grid.

spacingBetweenRows : float (optional)
Spacing between the rows.

spacingBetweenCols : float (optional)
Spacing between the columns.

2.1.7 DesignElement

The classes of this package are the contained classes of the **ArrayDesign** and describe through the **DesignElements** what is intended to be at each location of the **Array**. The **Feature** describes an intended location on the **Array**, the **Reporter** the Oligo, Clone, PCR Product that is on a **Feature** and the **CompositeSequence** which combines **Reporters** or **CompositeSequences** into what the child **DesignElements** are meant to represent biologically, e.g. a Gene, Exon, Splice Variant, etc.

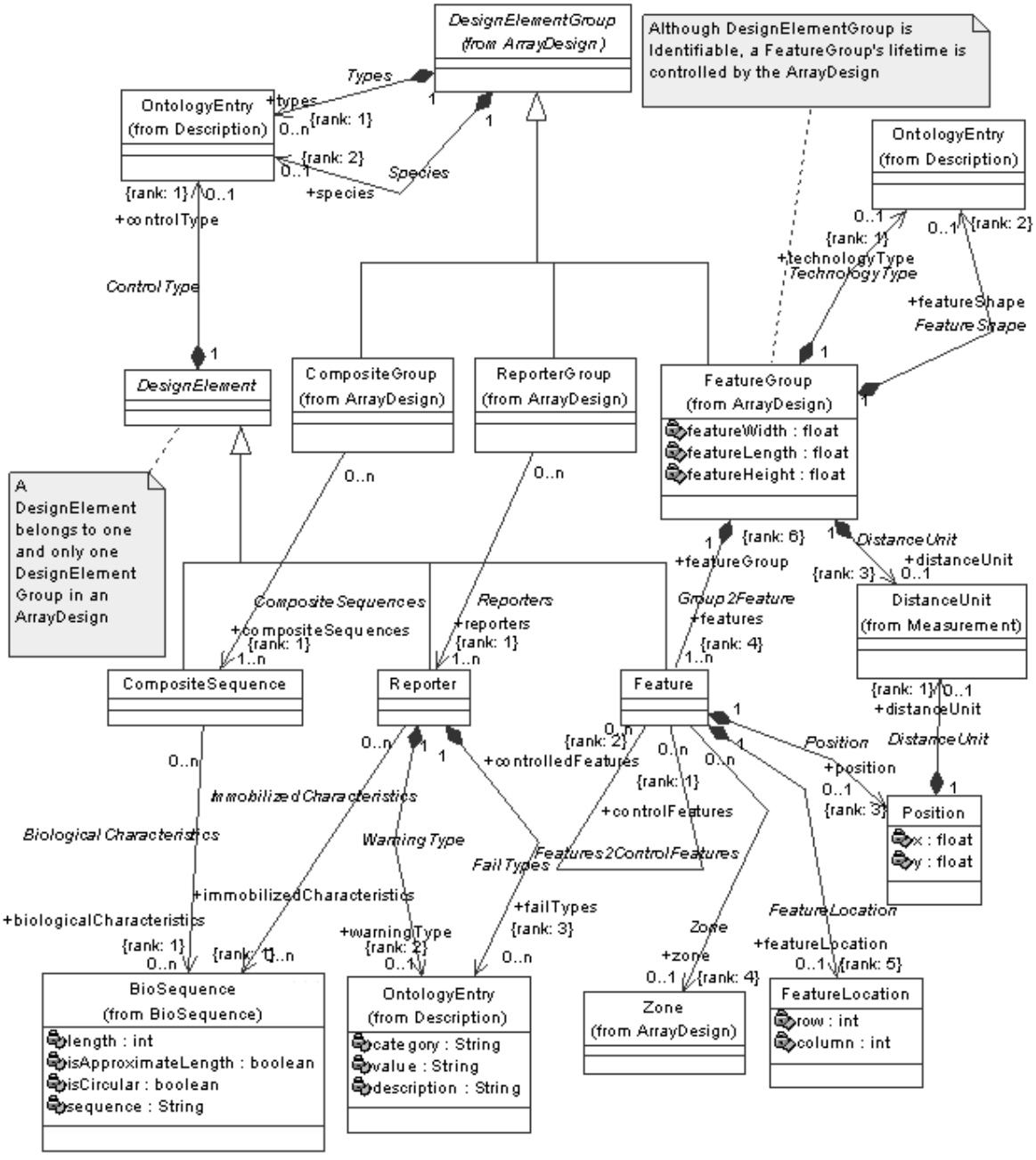

Figure 9: DesignElement

DesignElement

An element of an array. This is generally of type feature but can be specified as reporters or compositeSequence for arrays that are abstracted from a physical array.

Derived from Identifiable

Associations:

controlType : OntologyEntry (0..1)
If the design element represents a control, the type of control it is (normalization, deletion, negative, positive, etc.)

CompositeSequence

A collection of **Reporter** or **CompositeSequence** Design Elements, annotated through the association to **BioSequence**.

Derived from DesignElement

Associations:

biologicalCharacteristics : BioSequence (0..n)
The annotation on the **BioSequence** this **CompositeSequence** represents. Typically the sequences will be a Genes, Exons, or SpliceVariants.

reporterCompositeMaps : ReporterCompositeMap (0..n)
A map to the reporters that compose this **CompositeSequence**.

compositeCompositeMaps : CompositeCompositeMap (0..n)
A map to the compositeSequences that compose this **CompositeSequence**.

Reporter

A Design Element that represents some biological material (clone, oligo, etc.) on an array which will report on some biosequence or biosequences. The derived data from the measured data of its **Features** represents the presence or absence of the biosequence or biosequences it is reporting on in the **BioAssay**.

Reporters are Identifiable and several **Features** on the same array can be mapped to the same reporter as can **Features** from a different **ArrayDesign**. The granularity of the **Reporters** independence is dependent on the technology and the intent of the **ArrayDesign**. Oligos using mature technologies can in general be assumed to be safely replicated on many features where as with PCR Products there might be the desire for quality assurance to make reporters one to one with features and use the mappings to **CompositeSequences** for replication purposes.

Derived from DesignElement

Associations:

failTypes : OntologyEntry (0..n)
If at some time the reporter is determined to be failed this indicts the failure (doesn't report on what it was intended to report on, etc.)

warningType : OntologyEntry (0..1)
Similar to failType but indicates a warning rather than a failure.

immobilizedCharacteristics : BioSequence (0..n)
The sequence annotation on the **BioMaterial** this reporter represents. Typically the sequences will be an Oligo Sequence, Clone or PCR Primer.

featureReporterMaps : FeatureReporterMap (0..n)
Associates features with their reporter.

Feature

An intended position on an array.

Derived from DesignElement

Associations:

zone : Zone (0..1)
A reference to the zone this feature is in.

featureGroup : FeatureGroup (1..1)
The features that belong to this group.

position : Position (0..1)
The position of the feature on the array, relative to the top, left corner.

controlledFeatures : Feature (0..n)
Associates features with their control features.

controlFeatures : Feature (0..n)
Associates features with their control features.

featureLocation : FeatureLocation (0..1)
Location of this feature relative to a grid.

Position

Specifies a position on an array.

Derived from Extendable

Associations:

distanceUnit : DistanceUnit (0..1)
The units of the x, y positions.

Attributes:

x : float (required)
The horizontal distance from the upper left corner of the array.

y : float (required)
The vertical distance from the upper left corner of the array.

FeatureLocation

Specifies where a feature is located relative to a grid.

Derived from Extendable

Attributes:

row : int (required)
row position in the Zone

column : int (required)
column position in the Zone.

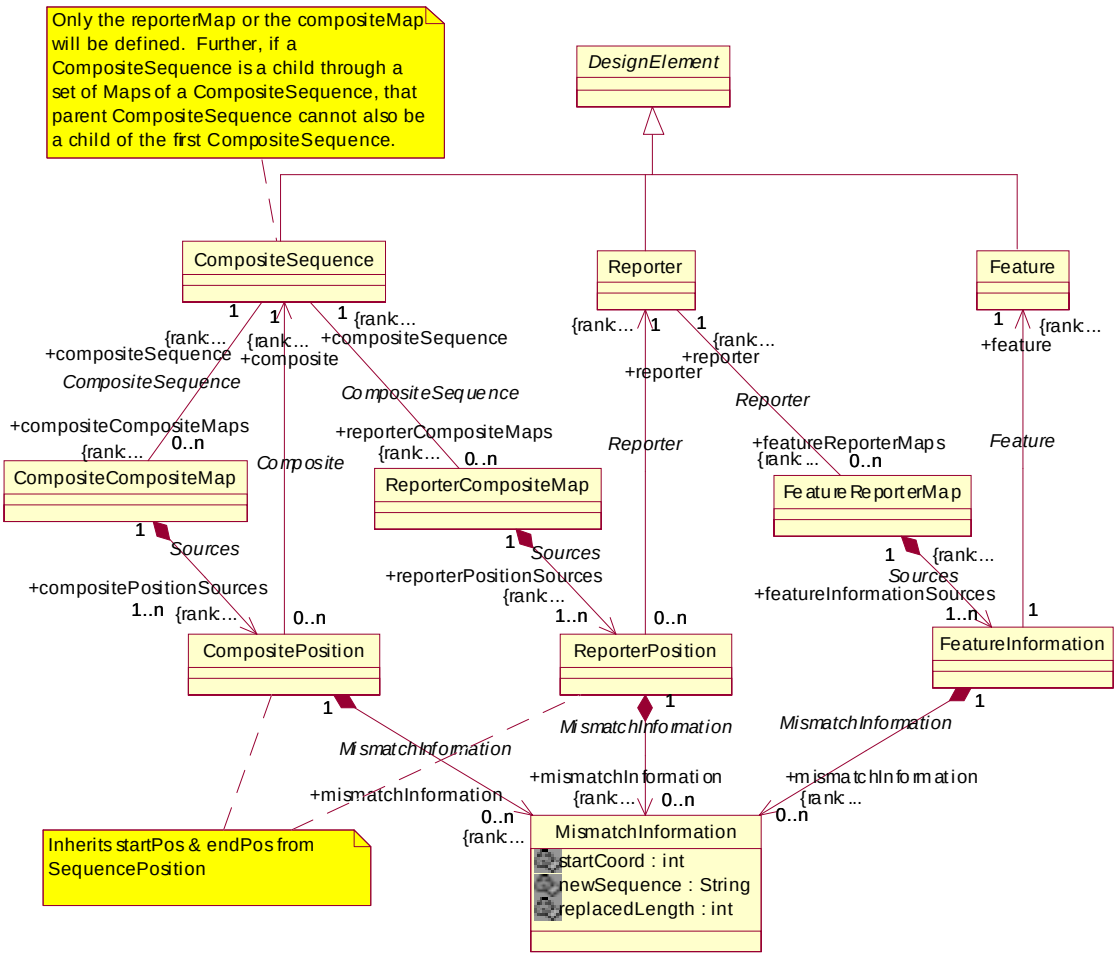

Figure 10: DesignElement Maps

CompositeCompositeMap

A **CompositeCompositeMap** is the description of how source **CompositeSequences** are transformed into a target **CompositeSequence**. For instance, several **CompositeSequences** could represent different sequence regions for a Gene and

could be mapped to different **CompositeSequences**, each representing a different splice variant for that Gene.

Derived from DesignElementMap

Associations:

compositePositionSources : CompositePosition (1..n)
Association to the **CompositeSequences** that compose this **CompositeSequence** and where those **CompositeSequences** occur.

compositeSequence : CompositeSequence (1..1)
A map to the compositeSequences that compose this **CompositeSequence**.

CompositePosition

The location in the **compositeSequence** target's sequence to which a source compositeSequence maps. The association to **MismatchInformation** allows the specification, usually for control purposes, of deviations from the **CompositeSequence's BioMaterial**.

Derived from SequencePosition

Associations:

composite : CompositeSequence (1..1)
A source **CompositeSequence** that is part of a target **CompositeSequence**

mismatchInformation : MismatchInformation (0..n)
Differences in how the contained compositeSequence matches its target compositeSequence's sequence.

ReporterCompositeMap

A **ReporterCompositeMap** is the description of how source **Reporters** are transformed into a target **CompositeSequences**. For instance, several reporters that tile across a section of a chromosome could be mapped to a **CompositeSequence**.

Derived from DesignElementMap

Associations:

reporterPositionSources : ReporterPosition (1..n)
Association to the reporters that compose this **CompositeSequence** and where those reporters occur.

compositeSequence : CompositeSequence (1..1)
A map to the reporters that compose this **CompositeSequence**.

ReporterPosition

The location in the composite target's sequence to which a source reporter maps. The association to **MismatchInformation** allows the specification, usually for control purposes, of deviations from the **CompositeSequence's BioMaterial**.

Derived from SequencePosition

Associations:

reporter : Reporter (1..1)
A reporter that comprises part of a **CompositeSequence**.

mismatchInformation : MismatchInformation (0..n)
Differences in how the reporter matches its compositeSequence's sequence.

FeatureReporterMap

A **FeatureReporterMap** is the description of how source features are transformed into a target reporter. These would map replicate features for a reporter to the reporter.

Derived from DesignElementMap

Associations:

reporter : Reporter (1..1)
Associates features with their reporter.

featureInformationSources : FeatureInformation (1..n)
Typically, the features on an array that are manufactured with this reporter's **BioSequence**.

FeatureInformation

As part of the map information, allows the association of one or more differences in the **BioMaterial** on a feature from the **BioMaterial** of the **Reporter**. Useful for control purposes such as in Affymetrix probe pairs.

Derived from Extendable

Associations:

feature : Feature (1..1)
The feature the **FeatureInformation** is supplying information for.

mismatchInformation : MismatchInformation (0..n)
Differences in how the feature matches the reporter's sequence, typical examples is the Affymetrix probe pair where one of the features is printed with a mismatch to the other feature's perfect match.

MismatchInformation

Describes how a reporter varies from its **ReporterCharacteristics** sequence(s) or how a **Feature** varies from its **Reporter** sequence.

Derived from Extendable

Attributes:

startCoord : int (required)
Offset into the sequence that the mismatch occurs.

newSequence : String (optional)
The sequence that replaces the specified sequence starting at start_coord.

replacedLength : int (required)

Length of the original sequence that is replaced. A deletion is specified when the length of the newSequence is less than the replacedLength.

2.1.8 Array

Describes the process of creating arrays from array designs. Includes information on how arrays are grouped together, if relevant, how an array deviates from its array design both in layout and per feature and potentially contains references to LIMS data that might contain more detail on the **BioMaterial** used to create the reporters.

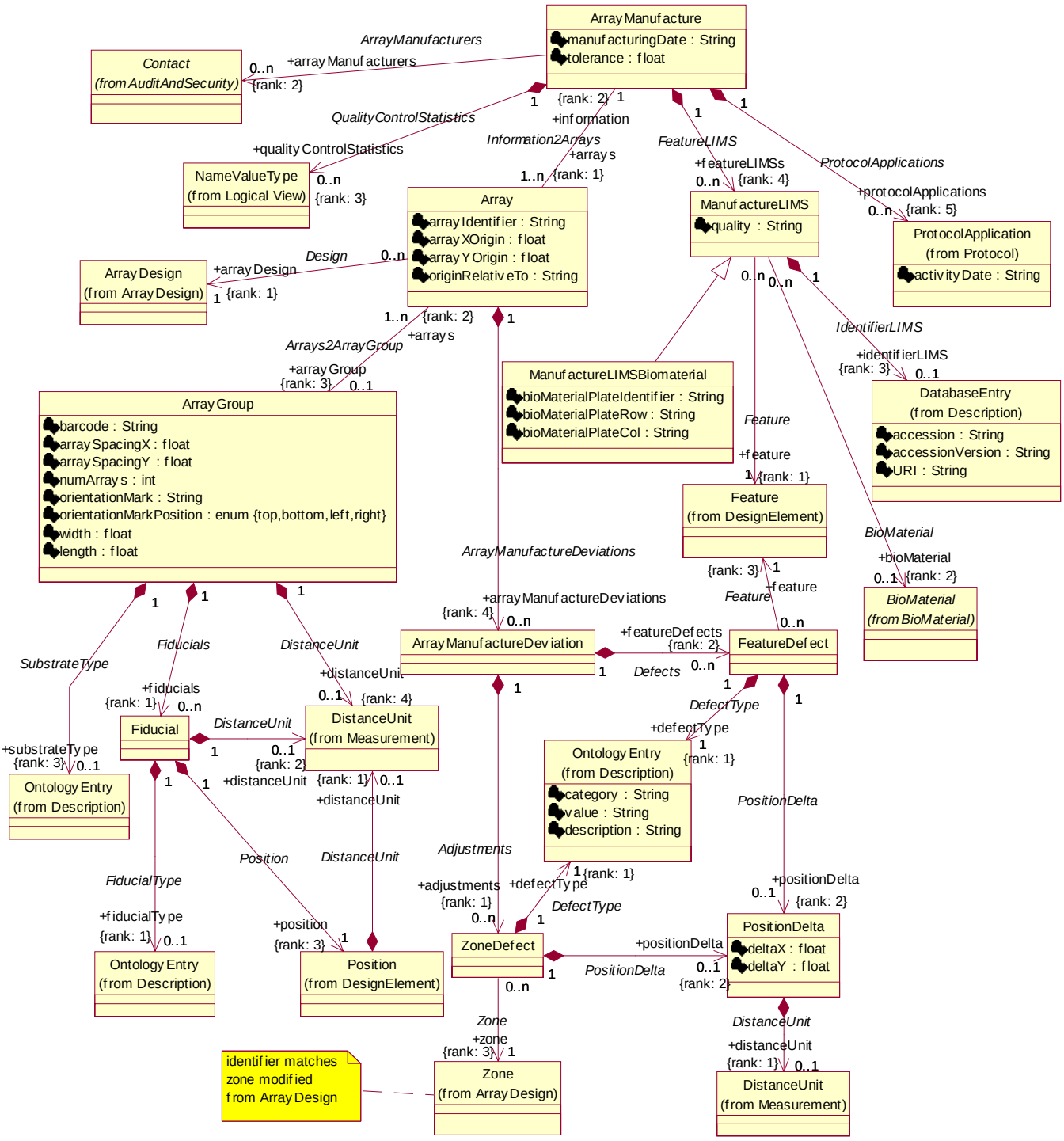

Figure 11: Array

Array

The physical substrate along with its features and their annotation

Derived from Identifiable

Associations:

arrayDesign : ArrayDesign (1..1)
The association of a physical array with its array design.

arrayGroup : ArrayGroup (0..1)
Association between an **ArrayGroup** and its **Arrays**, typically the **ArrayGroup** will represent a slide and the **Arrays** will be the manufactured so that they may be hybridized separately on that slide.

information : ArrayManufacture (1..1)
Association between the manufactured array and the information on that manufacture.

arrayManufactureDeviations : ArrayManufactureDeviation (0..n)
Association to classes to describe deviations from the **ArrayDesign**.

Attributes:

arrayIdentifier : String (optional)
An identifying string, e.g. a barcode.

arrayXOrigin : float (optional)
This can indicate the x position on a slide, chip, etc. of the first **Feature** and is usually specified relative to the fiducial.

arrayYOrigin : float (optional)
This can indicate the y position on a slide, chip, etc. of the first **Feature** and is usually specified relative to the fiducial.

originRelativeTo : String (optional)
What the array origin is relative to, e.g. upper left corner, fiducial, etc.

ArrayGroup

An array package is a physical platform that contains one or more arrays that are separately addressable (e.g. several arrays that can be hybridized on a single microscope slide) or a virtual grouping together of arrays.

The array package that has been manufactured has information about where certain artifacts about the array are located for scanning and feature extraction purposes.

Derived from Identifiable

Associations:

arrays : Array (1..n)
Association between an **ArrayGroup** and its **Arrays**, typically the **ArrayGroup** will represent a slide and the **Arrays** will be the manufactured so that they may be hybridized separately on that slide.

fiducials : Fiducial (0..n)

Association to the marks on the **Array** for alignment for the scanner.

distanceUnit : DistanceUnit (0..1)
Association to the marks on the **Array** for alignment for the scanner.

substrateType : OntologyEntry (0..1)
Commonly, arrays will be spotted on 1x3 glass microscope slides but there is nothing that says this must be the case. This association is for scanners to inform them on the possible different formats of slides that can contain arrays.

Attributes:

barcode : String (optional)
Identifier for the **ArrayGroup**.

arraySpacingX : float (optional)
If there exist more than one array on a slide or a chip, then the spacing between the arrays is useful so that scanning / feature extraction software can crop images representing 1 unique bioassay.

arraySpacingY : float (optional)
If there exist more than one array on a slide or a chip, then the spacing between the arrays is useful so that scanning / feature extraction software can crop images representing 1 unique bioassay.

numArrays : int (optional)
This attribute defines the number of arrays on a chip or a slide.

orientationMark : String (optional)
For a human to determine where the top left side of the array is, such as a barcode or frosted side of the glass, etc.

orientationMarkPosition : OrientationMarkPosition (optional)
enumeration {*top* | *bottom* | *left* | *right*}
One of top, bottom, left or right.

width : float (optional)
The width of the platform

length : float (optional)
The length of the platform.

ArrayManufacture

Describes the process by which arrays are produced.

Derived from Identifiable

Associations:

arrays : Array (1..n)
Association between the manufactured array and the information on that manufacture.

protocolApplications : ProtocolApplication (0..n)
The protocols followed in the manufacturing of the arrays.

featureLIMSs : ManufactureLIMS (0..n)
Information on the manufacture of the features.

arrayManufacturers : Contact (0..n)
The person or organization to contact for information concerning the **ArrayManufacture**.

qualityControlStatistics : NameValueType (0..n)
Information on the quality of the **ArrayManufacture**.

Attributes:

manufacturingDate : String (optional)
The date the arrays were manufactured

tolerance : float (optional)
The allowable error of a feature printed to its intended position.

ArrayManufactureDeviation

Stores information of the potential difference between an array design and arrays that have been manufactured using that design (e.g. a tip failed to print several spots).

Derived from Extendable

Associations:

featureDefects : FeatureDefect (0..n)
Description on features who are manufactured in a different location than specified in the **ArrayDesign**.

adjustments : ZoneDefect (0..n)
Descriptions of how a **Zone** has been printed differently than specified in the **ArrayDesign**.

FeatureDefect

Stores the defect information for a feature.

Derived from Extendable

Associations:

positionDelta : PositionDelta (0..1)
How the feature deviates in position from the **ArrayDesign**.

feature : Feature (1..1)
The feature that was manufactured defectively.

defectType : OntologyEntry (1..1)
Indicates the type of defect (e.g. a missing feature or a moved feature).

Fiducial

A marking on the surface of the array that can be used to identify the array's origin, the coordinates of which are the fiducial's centroid.

Derived from Describable

Associations:

distanceUnit : DistanceUnit (0..1)
The units the fiducial is measured in.

fiducialType : OntologyEntry (0..1)
A descriptive string that indicates the type of a fiducial (e.g. the chrome border on an Affymetrix array, a laser ablation mark).

position : Position (1..1)
The position, relative to the upper left corner, of the fiducial

ManufactureLIMS

Information on the physical production of arrays within the laboratory.

Derived from Describable

Associations:

feature : Feature (1..1)
The feature whose LIMS information is being described.

identifierLIMS : DatabaseEntry (0..1)
Association to a LIMS data source for further information on the manufacturing process.

bioMaterial : BioMaterial (0..1)
The **BioMaterial** used for the feature.

Attributes:

quality : String (optional)
A brief description of the quality of the array manufacture process.

ManufactureLIMSBiomaterial

Stores the location from which a biomaterial was obtained.

Derived from ManufactureLIMS

Attributes:

bioMaterialPlateIdentifier : String (optional)
The plate from which a biomaterial was obtained.

bioMaterialPlateRow : String (optional)
The plate row from which a biomaterial was obtained. Specified by a letter.

bioMaterialPlateCol : String (optional)
The plate column from which a biomaterial was obtained. Specified by a number.

PositionDelta

The delta the feature was actually printed on the array from the position specified for the feature in the array design.

Derived from Extendable

Associations:

distanceUnit : DistanceUnit (0..1)
The unit for the attributes.

Attributes:

deltaX : float (required)
Deviation from the y coordinate of this feature's position.

deltaY : float (required)
Deviation from the y coordinate of this feature's position.

ZoneDefect

Stores the defect information for a zone.

Derived from Extendable

Associations:

zone : Zone (1..1)
Reference to the **Zone** that was misprinted.

positionDelta : PositionDelta (0..1)
How the zone deviates in position from the **ArrayDesign**.

defectType : OntologyEntry (1..1)
Indicates the type of defect (e.g. a missing zone or a moved zone).

2.1.9 BioEvent

An abstract class representing an event that takes sources of some type to produce a target of some type. Each of the realized subclasses determines the type of the sources and the target. The association to a protocol application allows specification of how the event was performed.

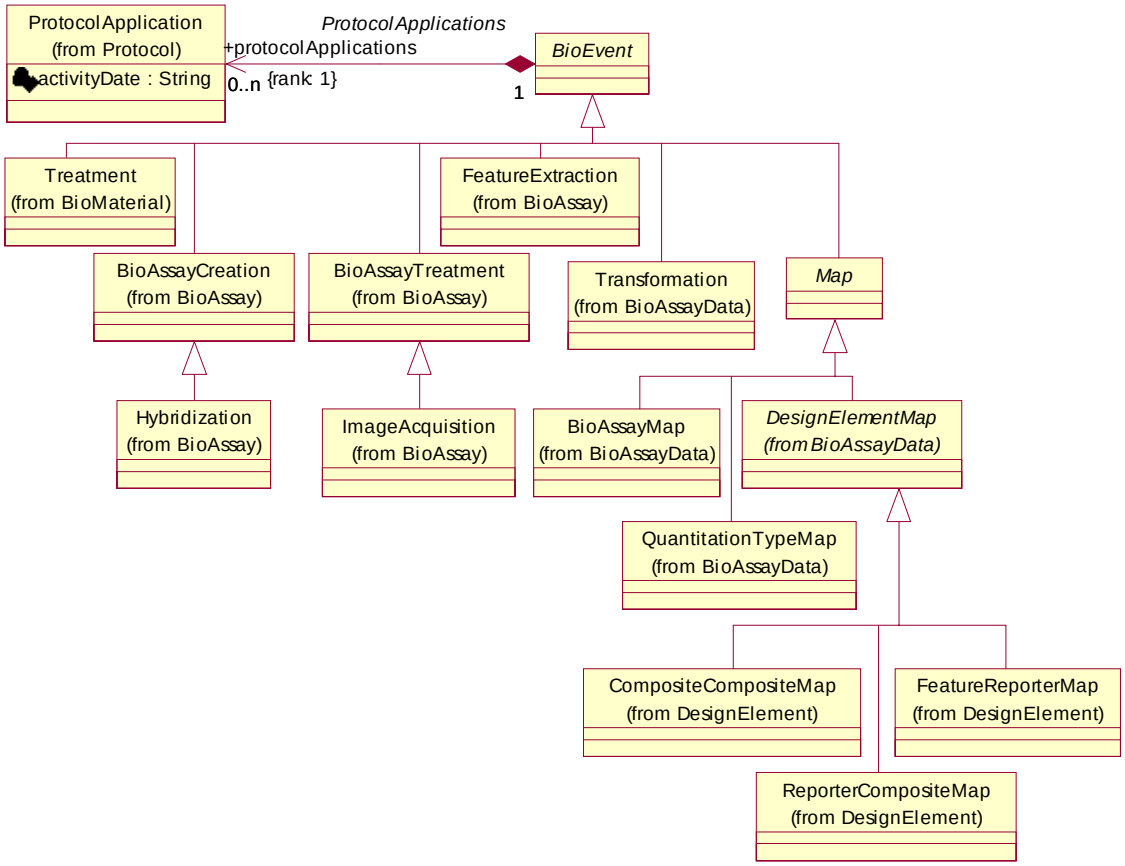

Figure 12: BioEvent

BioEvent

An abstract class to capture the concept of an event (either in the laboratory or a computational analysis).

Derived from Identifiable

Associations:

protocolApplications : ProtocolApplication (0..n)
The applied protocols to the BioEvent.

Map

A **Map** is the description of how sources are transformed into a target. Provides an abstract base class that separates the mapping **BioEvents** from the transforming.

Derived from BioEvent

2.1.10 BioMaterial

The classes in this package describe how a **BioSource** is treated to obtain the **BioMaterial** (typically a **LabeledExtract**) that is used by a **BioAssayCreation** in combination with an **Array** to produce a **PhysicalBioAssay**. A set of treatments are typically linear in time but can form a Directed Acyclic Graph.

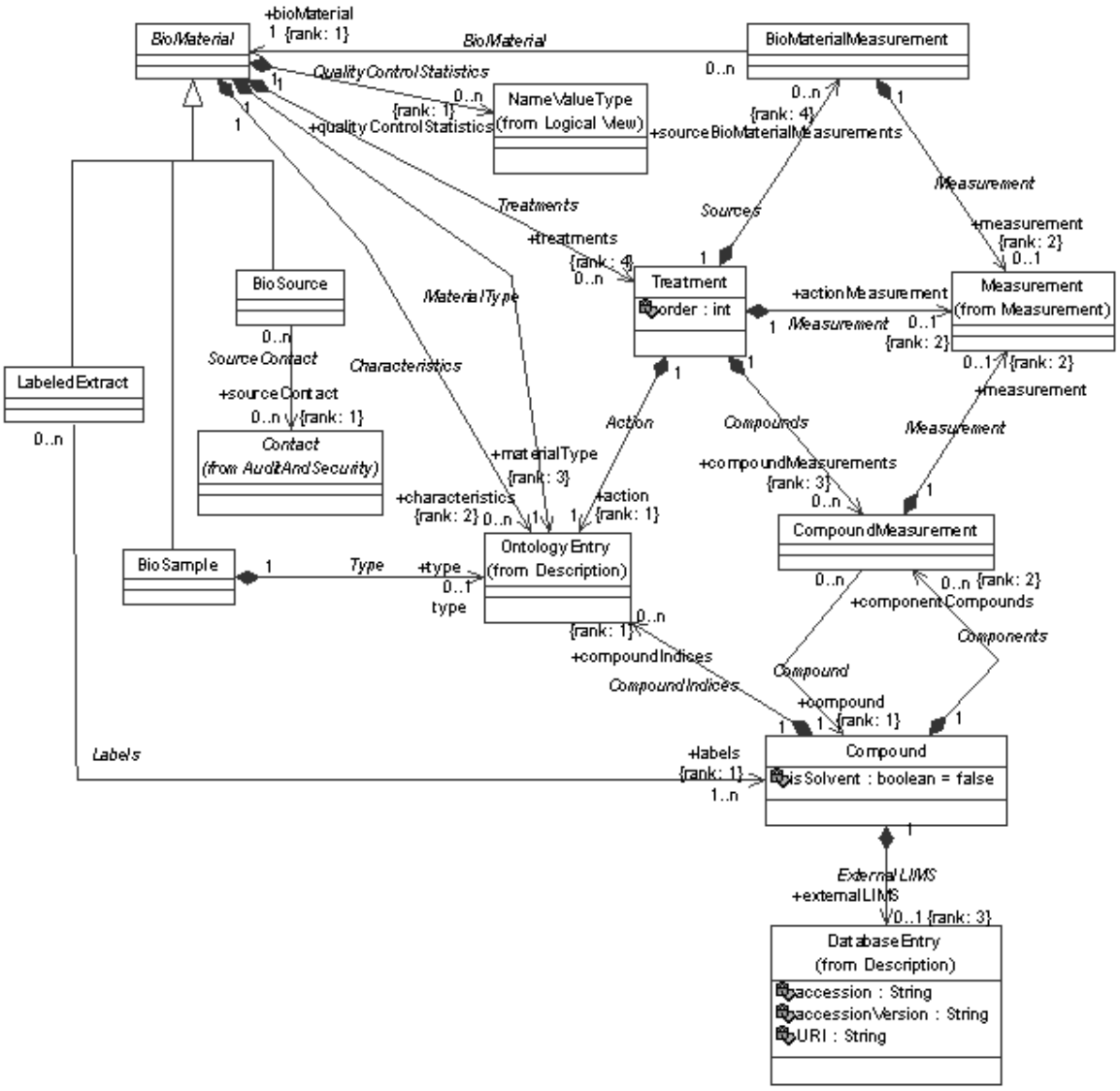

Figure 13: BioMaterial

BioMaterial

BioMaterial is an abstract class that represents the important substances such as cells, tissues, DNA, proteins, etc... Biomaterials can be related to other biomaterial through a directed acyclic graph (represented by treatment(s)).

Derived from Identifiable

Associations:

treatments : Treatment (0..n)
This association is one way from **BioMaterial** to **Treatment**. From this a **BioMaterial** can discover the amount and type of **BioMaterial** that was part of the treatment that produced it.

materialType : OntologyEntry (1..1)
The type of material used, i.e. rna, dna, lipid, phosphoprotein, etc.

characteristics : OntologyEntry (0..n)
Innate properties of the biosource, such as genotype, cultivar, tissue type, cell type, ploidy, etc.

qualityControlStatistics : NameValueType (0..n)
Measures of the quality of the **BioMaterial**.

BioSample

BioSamples are products of treatments that are of interest. **BioSamples** are often used as the sources for other biosamples. The **type** attribute describes the role the **BioSample** holds in the treatment hierarchy. This **type** can be an extract.

Derived from BioMaterial

Associations:

type : OntologyEntry (~~1~~0..1)
The **type** attribute describes the role the **BioSample** holds in the treatment hierarchy. This **type** can be an extract.

BioSource

The **BioSource** is the original source material before any treatment events. It is also a top node of the directed acyclic graph generated by treatments. The association to **OntologyEntry** allows enumeration of a **BioSource's** inherent properties.

Derived from BioMaterial

Associations:

sourceContact : Contact (0..n)
The **BioSource's** source is the provider of the biological material (a cell line, strain, etc...). This could be the ATTC (American Tissue Type Collection).

LabeledExtract

LabeledExtracts are special **BioSamples** that have **Compounds** which are detectable (these are often fluorescent or reactive moieties).

Derived from BioMaterial

Associations:

labels : Compound (1..n)
Compound used to label the extract.

Treatment

The process by which a biomaterial is created (from source biomaterials). Treatments have an order and an action.

Derived from BioEvent

Associations:

compoundMeasurements : CompoundMeasurement (0..n)
The compounds and their amounts used in the treatment.

sourceBioMaterialMeasurements : BioMaterialMeasurement (0..n)
The **BioMaterials** and the amounts used in the treatment

actionMeasurement : Measurement (0..1)
Measures events like duration, centrifuge speed, etc.

action : OntologyEntry (1..1)
The event that occurred (e.g. grow, wait, add, etc...). The actions should be a recommended vocabulary

Attributes:

order : int (optional)
The chronological order in which a treatment occurred (in relation to other treatments). More than one treatment can have the same chronological order indicating that they happened (or were caused to happen) simultaneously.

BioMaterialMeasurement

A **BioMaterialMeasurement** is a pairing of a source **BioMaterial** and an amount (**Measurement**) of that **BioMaterial**.

Derived from Extendable

Associations:

bioMaterial : BioMaterial (1..1)
A source **BioMaterial** for a treatment.

measurement : Measurement (0..1)
The amount of the **BioMaterial**.

Compound

A **Compound** can be a simple compound such as SDS (sodium dodecyl sulfate). It may also be made of other **Compounds** in proportions using **CompoundMeasurements** to enumerate the **Compounds** and their amounts such as LB (Luria Broth) Media.

Derived from Identifiable

Associations:

componentCompounds : CompoundMeasurement (0..n)
The **Compounds** and their amounts used to create this **Compound**.

compoundIndices~~merckIndex~~ : OntologyEntry (0..~~1~~n)
Indices into common Compound Indices, such as the Merck Index, for this **Compound**.~~The Merck Index of this **Compound**~~.

externalLIMS : DatabaseEntry (0..1)
Reference to an entry in an external LIMS data source.

Attributes:

isSolvent : boolean (default: *false*)
A **Compound** may be a special case solvent.

CompoundMeasurement

A **CompoundMeasurement** is a pairing of a source **Compound** and an amount (**Measurement**) of that **Compound**.

Derived from Extendable

Associations:

compound : Compound (1..1)
A **Compound** to be used to create another **Compound**.

measurement : Measurement (0..1)
The amount of the **Compound**.

2.1.11 BioAssay

Provides classes that contain information and annotation on the event of joining an **Array** with a **BioMaterial** preparation, the acquisition of images and the extraction of data on a per feature basis from those images. The derived classes of **BioAssay** represent the base **PhysicalBioAssays** which lead to the production of **Images**, the **MeasuredBioAssay** which is associated with the set of quantitations produced by **FeatureExtraction**, and **DerivedBioAssay** (see **BioAssayData** package) which groups together **BioAssays** that have been analyzed together to produce further refinement of the quantitations.

The design of this package and the related **BioAssayData** package was driven by the following query considerations and the desire to return as little data as necessary to satisfy a query. Often, the first set of queries for experiments below the **Experiment** level will want to discover the why of an experiment and this is captured in the

BioAssay class through its **FactorValue**, **BioEvent** and **Description** associations. This separates it from the data but allows an overview of the experiment hierarchy. The **BioAssayData** class association to **BioDataValues** is optional only to allow queries on them to discover the how of the experiment through the association to the transformation and mappings of the three **BioAssayData** dimensions and the protocols used. Once a researcher, for instance, has narrowed down the experiments of interest then the actual data, represented by the **BioDataValues**, can be downloaded. Because these data can be in the hundreds of megabytes to gigabytes range, it was considered desirable to be able to return information and annotation on the experiment without the data.

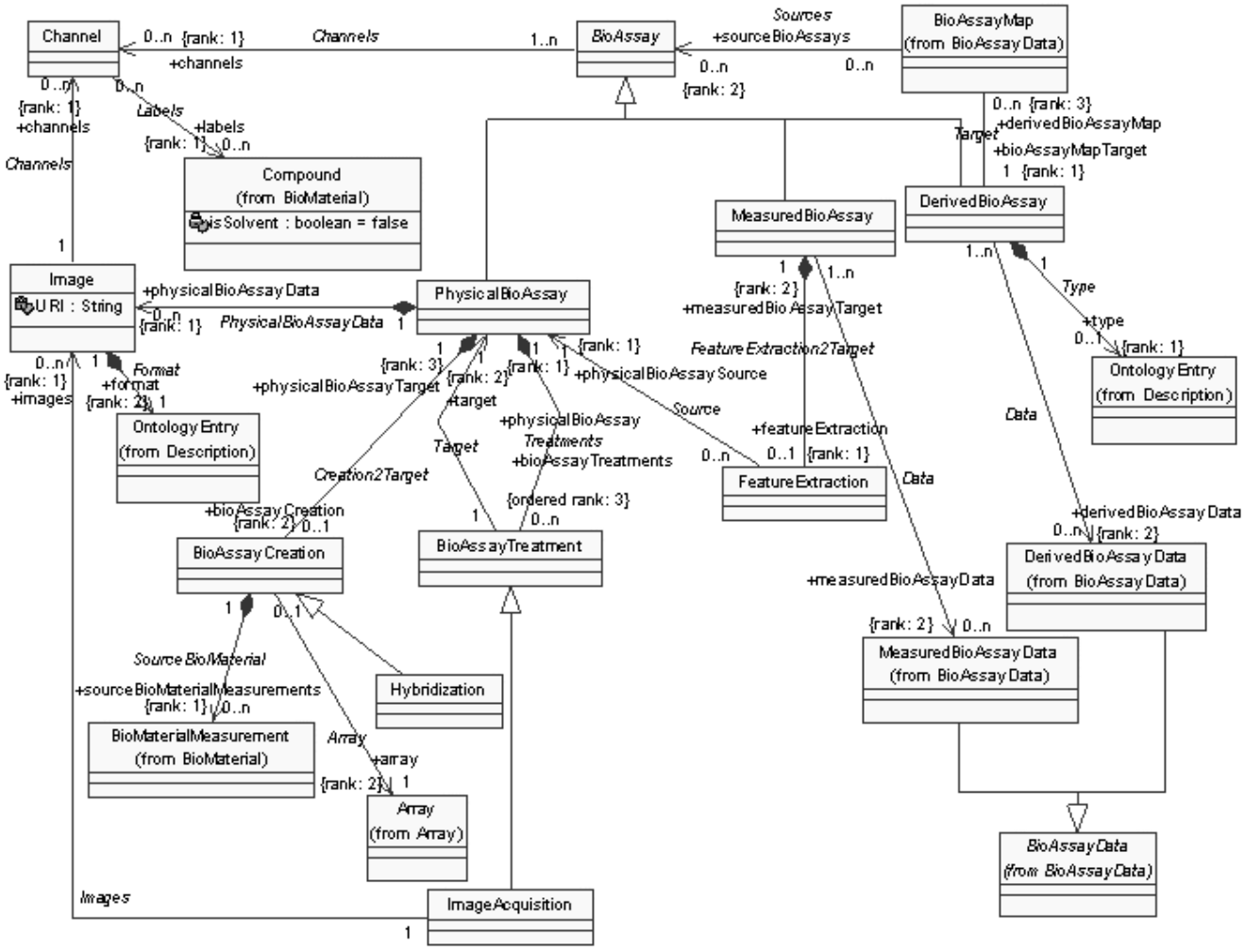

Figure 14: BioAssay

BioAssay

An abstract class which represents both physical and computational groupings of arrays and biomaterials.

Derived from Identifiable

Associations:

channels : Channel (0..n)
Channels can be non-null for all subclasses. For instance, collapsing across replicate features will create a **DerivedBioAssay** that will potentially reference channels.

bioAssayFactorValues : FactorValue (0..n)
The values that this **BioAssay** is associated with for the experiment.

PhysicalBioAssay

A bioAssay created by the bioAssayCreation event (e.g. in gene expression analysis this event is represented by the hybridization event).

Derived from BioAssay

Associations:

physicalBioAssayData : Image (0..n)
The **Images** associated with this **PhysicalBioAssay** by **ImageAcquisition**.

bioAssayCreation : BioAssayCreation (0..1)
The association between the **BioAssayCreation** event (typically **Hybridization**) and the **PhysicalBioAssay** and its annotation of this event.

bioAssayTreatments : BioAssayTreatment (0..n)
The set of treatments undergone by this **PhysicalBioAssay**.

MeasuredBioAssay

A measured bioAssay is the direct processing of information in a physical bioAssay by the featureExtraction event. Often uses images which are referenced through the physical bioAssay.

Derived from BioAssay

Associations:

featureExtraction : FeatureExtraction (0..1)
The association between the **MeasuredBioAssay** and the **FeatureExtraction** Event.

measuredBioAssayData : MeasuredBioAssayData (0..n)
The data associated with the **MeasuredBioAssay**.

DerivedBioAssay

A **BioAssay** that is created by the **Transformation BioEvent** from one or more **MeasuredBioAssays** or **DerivedBioAssays**.

Derived from BioAssay

Associations:

derivedBioAssayData : DerivedBioAssayData (0..n)
The data associated with the **DerivedBioAssay**.

derivedBioAssayMap : BioAssayMap(0..n)
The DerivedBioAssay that is produced by the sources of the BioAssayMap.

type : OntologyEntry (0..1)
The derivation type, for instance collapsed spot replicate, ratio, averaged intensity, bioassay replicates, etc.

BioAssayCreation

The process by which an array and one or more biomaterials are combined to create a bioAssayCreation.

Derived from BioEvent

Associations:

array : Array (1..1)
The array used in the **BioAssayCreation** event.

sourceBioMaterialMeasurements : BioMaterialMeasurement (0..n)
The **BioSample** and its amount used in the **BioAssayCreation** event.

physicalBioAssayTarget : PhysicalBioAssay (1..1)
The association between the **BioAssayCreation** event (typically **Hybridization**) and the **PhysicalBioAssay** and its annotation of this event.

Hybridization

The archetypal bioAssayCreation event, whereby biomaterials are hybridized to an array.

Derived from BioAssayCreation

BioAssayTreatment

The event which records the process by which **PhysicalBioAssays** are processed (typically washing, blocking, etc...).

Derived from BioEvent

Associations:

physicalBioAssay : PhysicalBioAssay (1..1)
The set of treatments undergone by this **PhysicalBioAssay**.

target : PhysicalBioAssay (1..1)
The **PhysicalBioAssay** that was treated.

FeatureExtraction

The process by which data is extracted from an image producing a measuredBioAssayData and a measuredBioAssay.

Derived from BioEvent

Associations:

physicalBioAssaySource : PhysicalBioAssay (1..1)
The **PhysicalBioAssay** used in the **FeatureExtraction** event.

measuredBioAssayTarget : MeasuredBioAssay (1..1)
The association between the **MeasuredBioAssay** and the **FeatureExtraction** Event.

Image

An image is created by an imageAcquisition event, typically by scanning the hybridized array (the **PhysicalBioAssay**).

Derived from Identifiable

Associations:

channels : Channel (0..n)
The channels captured in this image.

format : OntologyEntry (1..1)
The file format of the image typically a TIF or a JPEG.

Attributes:

URI : String (optional)
The file location in which an image may be found.

ImageAcquisition

The process by which an image is generated (typically scanning).

Derived from BioAssayTreatment

Associations:

images : Image (0..n)
The images produced by the **ImageAcquisition** event.

Channel

A channel represents an independent acquisition scheme for the **ImageAcquisition** event, typically a wavelength.

Derived from Identifiable

Associations:

labels : Compound (0..n)
The compound used to label the extract.

2.1.12 BioAssayData

The classes defined here provide data and the information and annotation on the derivation of that data. Some of the scenarios that might occur are the following.

FeatureExtraction of a single **PhysicalBioAssay** produces **MeasuredBioAssayData** that has a single **BioAssay** on the **BioAssayDimension**, typically the **Features** described in the **ArrayDesign** on the **DesignElementDimension**, and the **Quantitations** associated with the application of a **FeatureExtraction** protocol on the **QuantitationDimension**.

An error model transformation might be applied that doesn't change the **BioAssayDimension** or the **DesignElementDimension** but likely changes the **QuantitationDimension**. A transformation on replicate **Reporters** or **CompositeSequences** might be applied on the single **BioAssay** that would change the **DesignElementDimension** and the **QuantitationDimension** both. Replicate and Control **BioAssays** might be added to the **BioAssayDimension** and a transformation could change the **BioAssayDimension** and the **QuantitationDimension** but not change the **DesignElementDimension** to produce a new **DerivedBioAssayData**. Or some combination of the above transformations could be performed at once to change all three dimensions.

Because the classes derive from **Describable**, the Experimenter can provide as much detail at each level of the class hierarchy as desired.

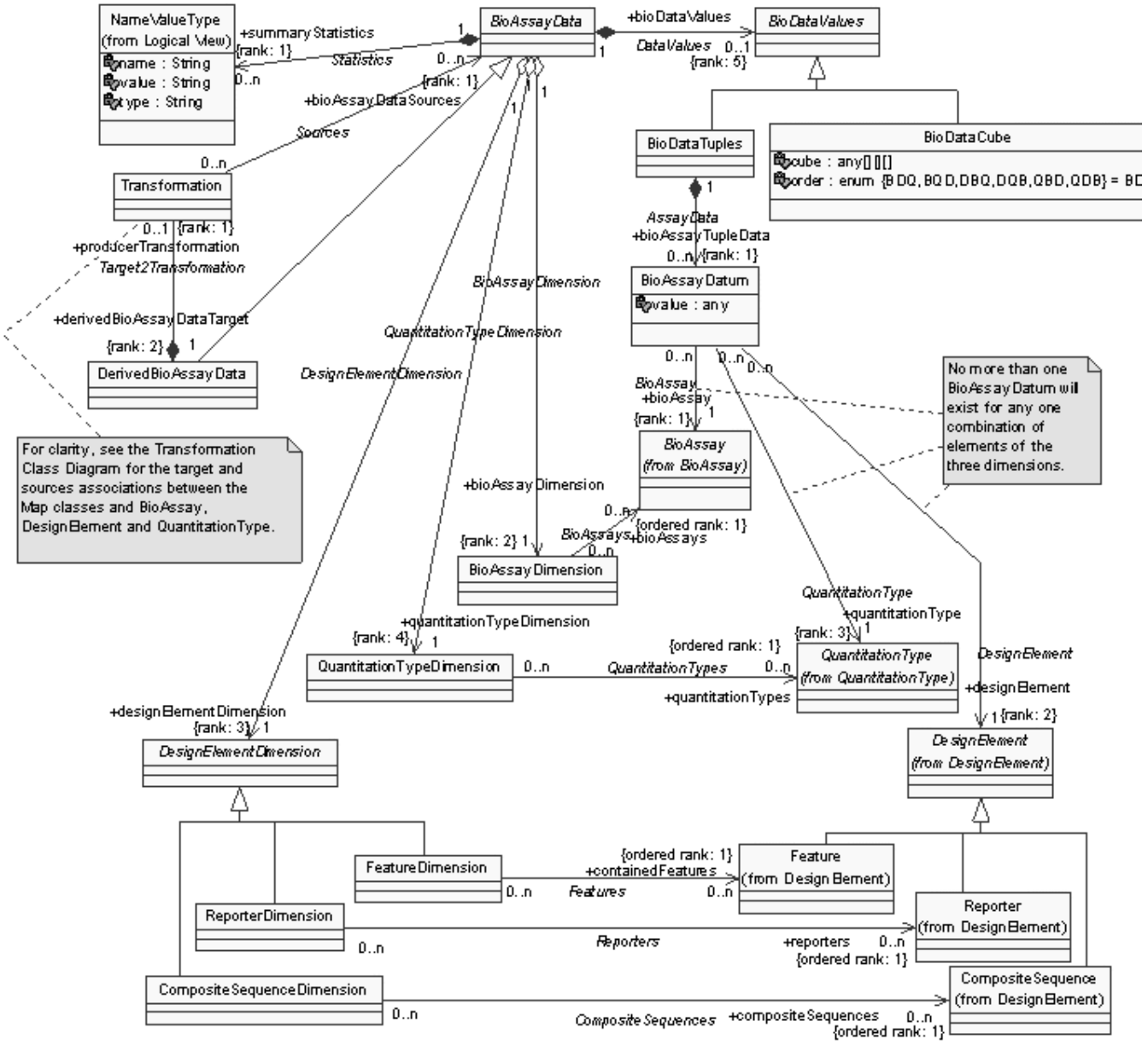

Figure 15: BioAssayData

BioAssayData

Represents the dataset created when the **BioAssays** are created. **BioAssayData** is the entry point to the values. Because the actual values are represented by a different object, **BioDataValues**, which can be memory intensive, the annotation of the transformation can be gotten separate from the data.

Derived from Identifiable

Associations:

bioAssayDimension : BioAssayDimension (1..1)
The **BioAssays** of the **BioAssayData**.

designElementDimension : DesignElementDimension (1..1)
The **DesignElements** of the **BioAssayData**.

quantitationTypeDimension : QuantitationTypeDimension (1..1)
The **QuantitationTypes** of the **BioAssayData**.

summaryStatistics : NameValueType (0..n)
Statistics on the Quality of the **BioAssayData**.

bioDataValues : BioDataValues (0..1)
The data values of the **BioAssayData**.

MeasuredBioAssayData

The data associated with the **MeasuredBioAssay** produced by **FeatureExtraction**.

Derived from BioAssayData

DerivedBioAssayData

The output of a transformation event.

Derived from BioAssayData

Associations:

producerTransformation : Transformation (0..1)
The association between the **DerivedBioAssayData** and the **Transformation** event that produced it.

BioDataValues

The actual values for the **BioAssayCube**.

Derived from Extendable

BioDataCube

A three-dimensional cube representation of the data.

Derived from BioDataValues

Attributes:

cube : any[][][] (required)
Three dimension array, indexed by the three dimensions to provide the data for the **BioAssayData**.

order : Order (default: *BDQ*)
enumeration {*BDQ* | *BQD* | *DBQ* | *DQB* | *QBD* | *QDB*}
The order to expect the dimension. The enumeration uses the first letter of the three dimensions to represent the six possible orderings.

BioDataTuples

A relational, tuple representation of the data.

Derived from BioDataValues

Associations:

bioAssayTupleData : BioAssayDatum (0..n)
The collection of **BioAssayData** tuples.

BioAssayDatum

A single cell of the quantitation, bioAssay, designElement matrix.

Derived from Extendable

Associations:

bioAssay : BioAssay (1..1)
The **BioAssay** associated with the value of the **BioAssayDatum**.

designElement : DesignElement (1..1)
The **DesignElement** associated with the value of the **BioAssayDatum**.

quantitationType : QuantitationType (1..1)
The **QuantitationType** associated with the value of the **BioAssayDatum**.

Attributes:

value : any (required)
The datum value.

BioAssayDimension

An ordered list of bioAssays.

Derived from Identifiable

Associations:

bioAssays : BioAssay (0..n)
The **BioAssays** for this Dimension

DesignElementDimension

An ordered list of designElements. It will be realized as one of its three subclasses.

Derived from Identifiable

CompositeSequenceDimension

Specialized **DesignElementDimension** to hold **CompositeSequences**.

Derived from DesignElementDimension

Associations:

compositeSequences : CompositeSequence (0..n)
The **CompositeSequences** for this Dimension.

ReporterDimension

Specialized **DesignElementDimension** to hold **Reporters**.

Derived from DesignElementDimension

Associations:

reporters : Reporter (0..n)
The reporters for this dimension.

FeatureDimension

Specialized **DesignElementDimension** to hold **Features**.

Derived from DesignElementDimension

Associations:

containedFeatures : Feature (0..n)
The features for this dimension.

QuantitationTypeDimension

An ordered list of quantitationTypes.

Derived from Identifiable

Associations:

quantitationTypes : QuantitationType (0..n)
The **QuantitationTypes** for this **Dimension**.

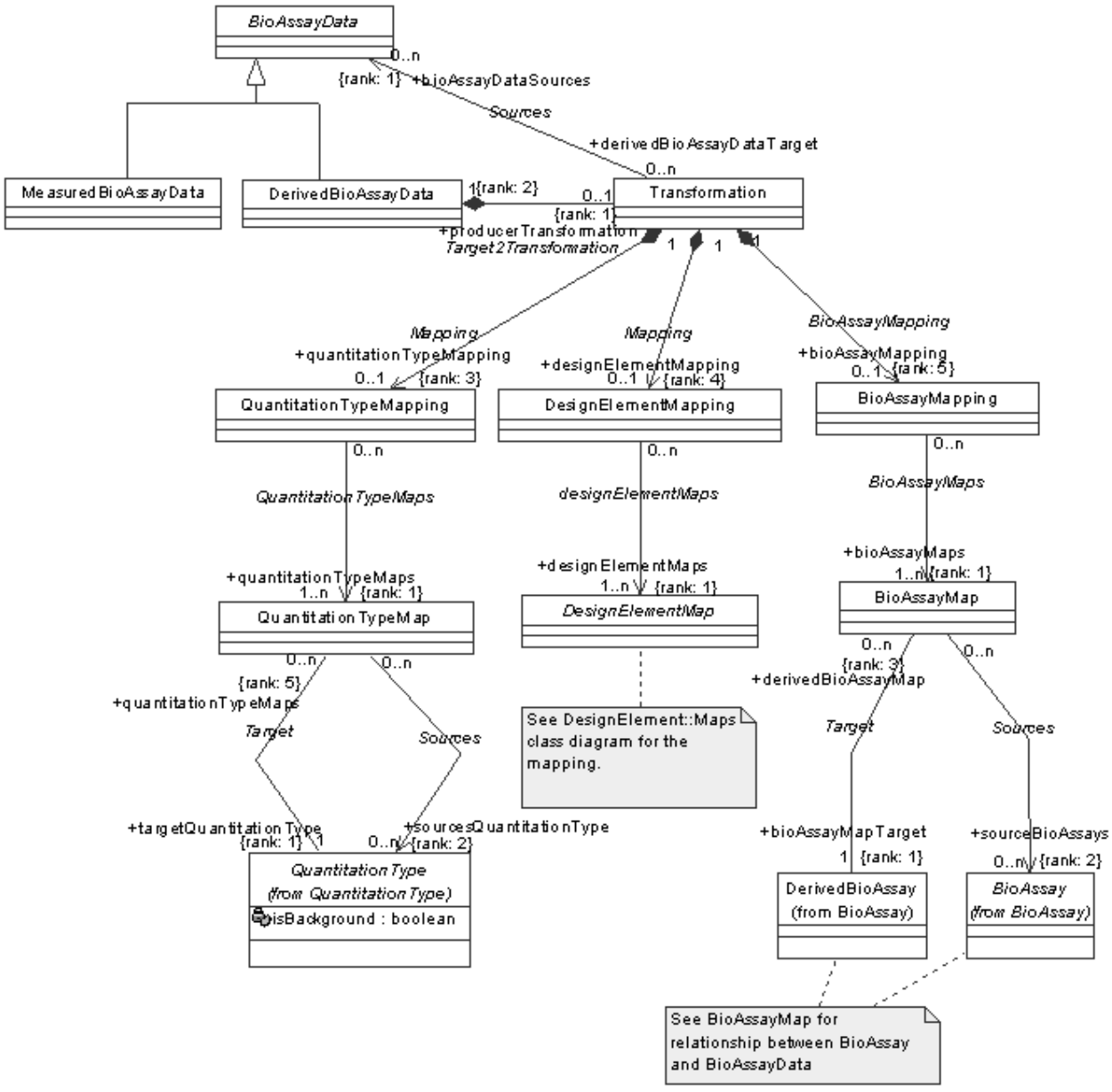

Figure 16: BioAssayData Transformation

Transformation

The process by which derivedBioAssays are created from measuredBioAssays and/or derivedBioAssays. It uses mappings to indicate the input and output dimensions.

Derived from BioEvent

Associations:

bioAssayDataSources : BioAssayData (0..n)
The **BioAssayData** sources that the **Transformation** event uses to produce the target **DerivedBioAssayData**.

bioAssayMapping : BioAssayMapping (0..1)
The collection of mappings for the **BioAssays**.

derivedBioAssayDataTarget : DerivedBioAssayData (1..1)
The association between the **DerivedBioAssayData** and the **Transformation** event that produced it.

quantitationTypeMapping : QuantitationTypeMapping (0..1)
The collection of mappings for the **QuantitationTypes**.

designElementMapping : DesignElementMapping (0..1)
The collection of mappings for the **DesignElements**.

BioAssayMapping

Container of the mappings of the input **BioAssay** dimensions to the output **BioAssay** dimension.

Derived from Extendable

Associations:

bioAssayMaps : BioAssayMap (1..n)
The maps for the **BioAssays**.

BioAssayMap

The **BioAssayMap** is the description of how source **MeasuredBioAssays** and/or **DerivedBioAssays** are manipulated (mathematically) to produce **DerivedBioAssays**.

Derived from Map

Associations:

bioAssayMapTarget : DerivedBioAssay (1..1)
The **DerivedBioAssay** that is produced by the sources of the **BioAssayMap**.

sourceBioAssays : BioAssay (0..n)
The sources of the **BioAssayMap** that are used to produce a target **DerivedBioAssay**.

DesignElementMapping

Container of the mappings of the input **DesignElement** dimensions to the output **DesignElement** dimension.

Derived from Extendable

Associations:

designElementMaps : DesignElementMap (1..n)
The maps for the DesignElements.

DesignElementMap

A **DesignElementMap** is the description of how source **DesignElements** are transformed into a target **DesignElement**.

Derived from Map

QuantitationTypeMapping

Container of the mappings of the input **QuantitationType** dimensions to the output **QuantitationType** dimension.

Derived from Extendable

Associations:

quantitationTypeMaps : QuantitationTypeMap (1..n)
The maps for the **QuantitationTypes**.

QuantitationTypeMap

A QuantitationTypeMap is the description of how source **QuantitationTypes** are mathematically transformed into a target **QuantitationType**.

Derived from Map

Associations:

targetQuantitationType : QuantitationType (1..1)
The **QuantitationType** whose value will be produced from the values of the source **QuantitationType** according to the **Protocol**.

sourcesQuantitationType : QuantitationType (0..n)
The **QuantitationType** sources for values for the transformation.

2.1.13 Experiment

Represents the container for a hierarchical grouping of **BioAssays**. Can have the end results of Clustering Analysis specified and, through the **ExperimentDesign**, a description and annotation of the overall design of the experiment and what it was to show.

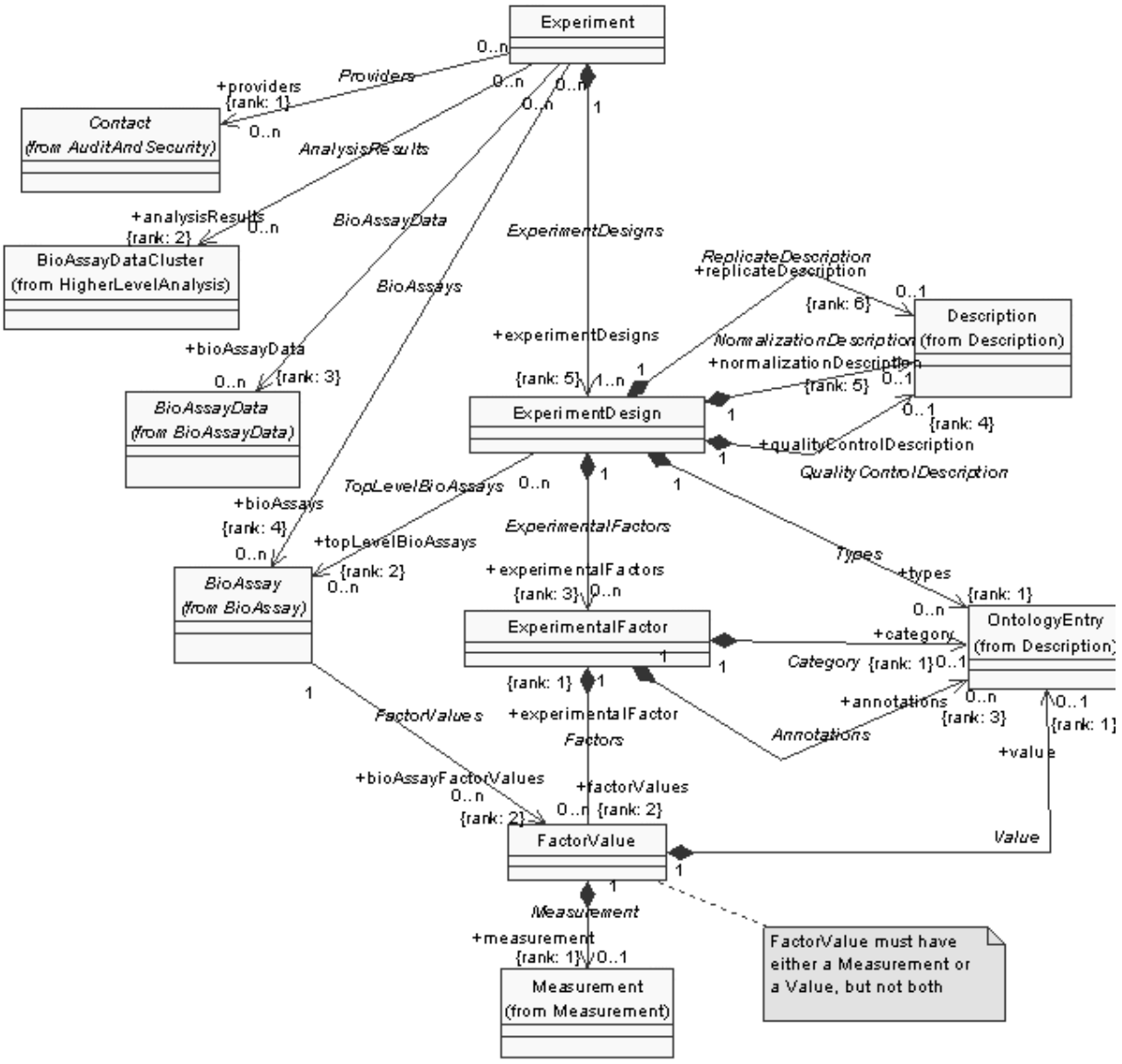

Figure 17: Experiment

Experiment

The **Experiment** is the collection of all the **BioAssays** that are related by the **ExperimentDesign**.

Derived from Identifiable

Associations:

analysisResults : BioAssayDataCluster (0..n)
The results of analyzing the data, typically with a clustering algorithm.

bioAssays : BioAssay (0..n)
The collection of **BioAssays** for this **Experiment**.

providers : Contact (0..n)
The providers of the **Experiment**, its data and annotation.

bioAssayData : BioAssayData (0..n)
The collection of **BioAssayDatas** for this **Experiment**.

experimentDesigns : ExperimentDesign (1..~~1~~n)
The association to the description and annotation of the **Experiment**, along with the grouping of the top-level **BioAssays**.

ExperimentDesign

The **ExperimentDesign** is the description and collection of **ExperimentalFactors** and the hierarchy of **BioAssays** to which they pertain.

Derived from Describable

Associations:

topLevelBioAssays : BioAssay (0..n)
The organization of the **BioAssays** as specified by the **ExperimentDesign** (TimeCourse, Dosage, etc.)

experimentalFactors : ExperimentalFactor (0..n)
The description of the factors (TimeCourse, Dosage, etc.) that group the **BioAssays**.

replicateDescription : Description (0..1)
Description of the replicate strategy of the **Experiment**.

qualityControlDescription : Description (0..1)
Description of the quality control aspects of the **Experiment**.

normalizationDescription : Description (0..1)
Description of the normalization strategy of the **Experiment**.

types : OntologyEntry (0..n)
Classification of an experiment. For example 'normal vs. diseased', 'treated vs. untreated', 'time course', 'tili ng', etc.

ExperimentalFactor

ExperimentFactors are the dependent variables of an experiment (e.g. time, glucose concentration, ...).

Derived from Identifiable

Associations:

factorValues : FactorValue (0..n)

The pairing of **BioAssay FactorValues** with the **ExperimentDesign ExperimentFactor**.

annotations : OntologyEntry (0..n)
Allows describing additional information such as concentration of Tamoxafin with a CASRegistry #.

category : OntologyEntry (0..1)
The category of an **ExperimentalFactor** could be biological (time, [glucose]) or a methodological factor (differing cDNA preparation protocols).

FactorValue

The value for a **ExperimentalFactor**

Derived from Identifiable

Associations:

experimentalFactor : ExperimentalFactor (1..1)
The pairing of **BioAssay FactorValues** with the **ExperimentDesign ExperimentFactor**.

measurement : Measurement (1..1)
The measured value for this factor.

value : OntologyEntry (0..1)
Allows a more complex value to be specified for a FactorValue than a simple Measurement.

Use Case: Searching the BioAssay Hierarchy

Given an **Experiment** locate the list of **PhysicalBioAssays** that produced each of the top-level **BioAssays** in the experiment (in this case we limit the search to **BioAssays** created from Hybridization events).

1) Fetch list of top-level **BioAssays** from **Experiment** using:

topLevelAssays = experiment.experimentDesign.topLevelBioAssays

2) For each top-level **BioAssay**, find the list of **PhysicalBioAssays** hybridized to create it. To do this we traverse the hierarchy of **BioEvents** that created the **BioAssays**. We access the **BioEvents** differently depending on the type of **BioAssay**:

if assay.type == **DerivedBioAssay**

childAssays = assay.bioAssayMap.sources

if assay.type == **MeasuredBioAssay**

childAssay = assay.featureExtraction.source

if assay.type == **PhysicalBioAssay**

// we may be done, check recursion end condition

if assay.bioAssayCreation != NULL return

// not done, keep recursing

childAssay = assay.bioAssayTreatment.source

2.1.14 HigherLevelAnalysis

Describes the results of performing analysis on the result of the **BioAssayData** from an **Experiment**.

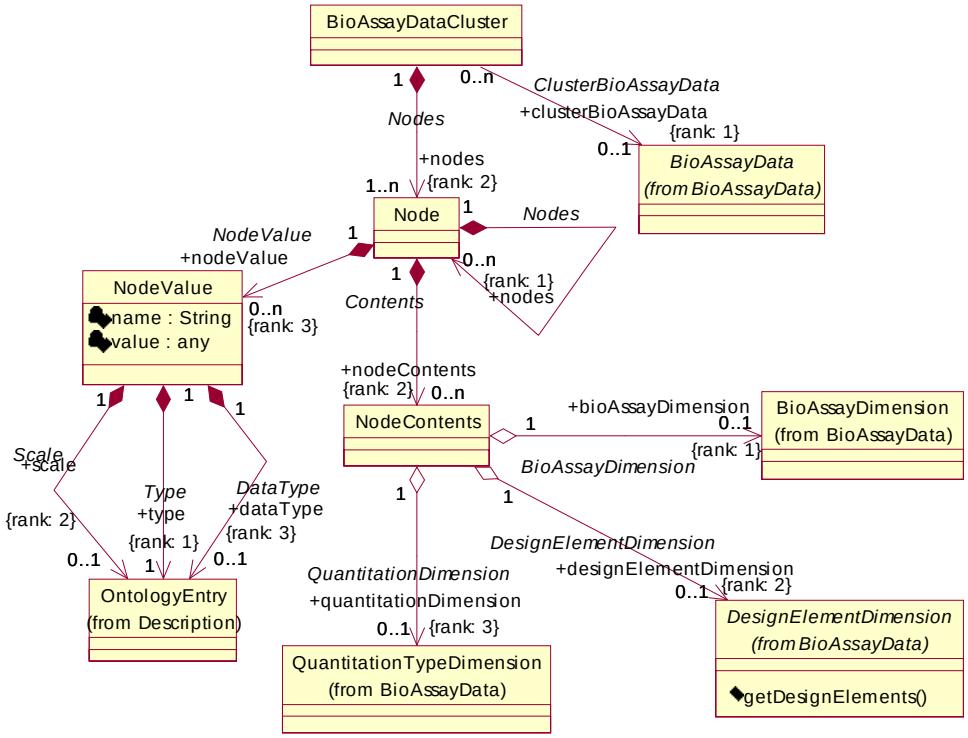

Figure 18: HigherLevelAnalysis

BioAssayDataCluster

A mathematical method of higher level analysis whereby **BioAssayData** are grouped together into nodes.

Derived from Identifiable

Associations:

nodes : Node (1..n)
The nodes of the cluster.

clusterBioAssayData : BioAssayData (0..1)
The **BioAssayData** whose values were used by the cluster algorithm.

Node

An individual component of a clustering. May contain other nodes.

Derived from Describable

Associations:

nodes : Node (0..n)
Nested nodes of the **BioAssayDataCluster**.

nodeContents : NodeContents (0..n)
The contents of the node, expressed as either a one, two or three dimensional object.

nodeValue : NodeValue (0..n)
Values or measurements for this node that may be produced by the clustering algorithm. Typical are distance values for the nodes.

NodeContents

The contents of a node for any or all of the three Dimensions. If a node only contained genes just the **DesignElementDimension** would be defined.

Derived from Describable

Associations:

designElementDimension : DesignElementDimension (0..1)
The relevant **DesignElements** for this **NodeContents** from the **BioAssayData**.

quantitationDimension : QuantitationTypeDimension (0..1)
The relevant **QuantitationTypes** for this **NodeContents** from the **BioAssayData**.

bioAssayDimension : BioAssayDimension (0..1)
The relevant **BioAssays** for this **NodeContents** from the **BioAssayData**.

NodeValue

A value associated with the **Node** that can rank it in relation to the other nodes produced by the clustering algorithm.

Derived from Extendable

Associations:

scale : OntologyEntry (0..1)
The scale (linear, log10, ln, etc.) of the value.

dataType : OntologyEntry (0..1)
The data type of the any element.

type : OntologyEntry (1..1)
The type of value, distance, etc.

Attributes:

name : String (optional)
The name for this value.

value : any (required)
The value for this **NodeValue**.

2.1.15 Measurement

The classes of this package provide utility information on the quantities of other classes to each other.

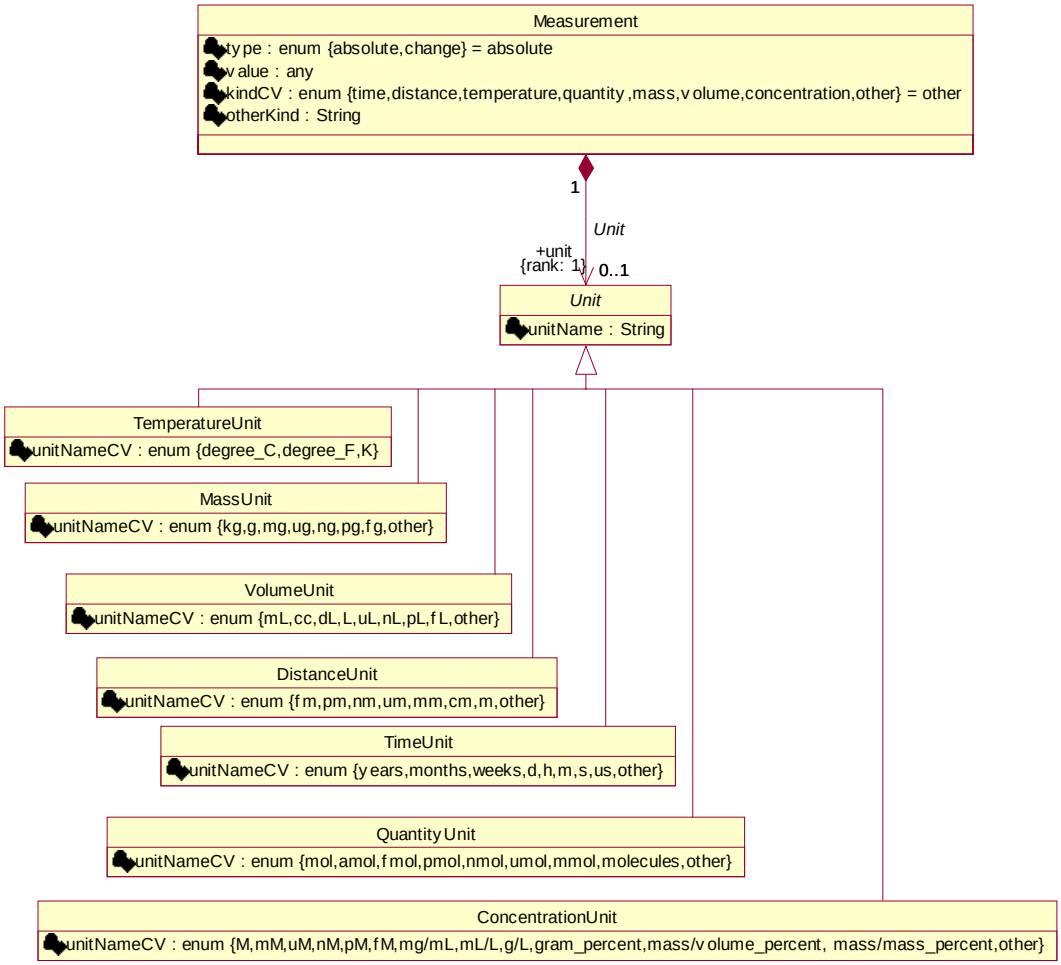

Figure 19: Measurement

Measurement

A **Measurement** is a quantity with a unit.

Derived from Extendable

Associations:

unit : Unit (0..1)
The **Unit** associated with the **Measurement**.

Attributes:

type : Type (default: *absolute*)
enumeration {*absolute* | *change*}
The type of measurement, for instance if the measurement is five feet, it can be either absolute (five feet tall) or change (five feet further along).

value : any (optional)
The value of the measurement.**kindCV** (and **otherKind**) determine with **Unit** the datatype of value.

kindCV : KindCV (default: *other*)
enumeration {*time* | *distance* | *temperature* | *quantity* | *mass* | *volume* | *concentration* | *other*}
One of the enumeration values to determine the controlled vocabulary of the value.

otherKind : String (optional)
Name of the controlled vocabulary if it isn't one of the **Unit** subclasses.

Unit

The unit is a strict enumeration of types.

Derived from Extendable

Attributes:

unitName : String (optional)
The name of the unit.

ConcentrationUnit

Concentration

Derived from Unit

Attributes:

unitNameCV : UnitNameCV (required)
enumeration {*M* | *mM* | *uM* | *nM* | *pM* | *fM* | *mg/mL* | *mL/L* | *g/L* | *gram_percent* | *mass/volume_percent* | *mass/mass_percent* | *other*}

DistanceUnit

Distance

Derived from Unit

Attributes:

unitNameCV : UnitNameCV (required)
enumeration {*fm* | *pm* | *nm* | *um* | *mm* | *cm* | *m* | *other*}

MassUnit

Mass

Derived from Unit

Attributes:

unitNameCV : UnitNameCV (required)
enumeration {*kg* | *g* | *mg* | *ug* | *ng* | *pg* | *fg* | *other*}

QuantityUnit

Quantity

Derived from Unit

Attributes:

unitNameCV : UnitNameCV (required)
enumeration {*mol* | *amol* | *fmol* | *pmol* | *nmol* | *umol* | *mmol* | *molecules* | *other*}

TemperatureUnit

Temperature

Derived from Unit

Attributes:

unitNameCV : UnitNameCV (required)
enumeration {*degree_C* | *degree_F* | *K*}

TimeUnit

Time

Derived from Unit

Attributes:

unitNameCV : UnitNameCV (required)
enumeration {*years* | *months* | *weeks* | *d* | *h* | *m* | *s* | *us* | *other*}

VolumeUnit

Volume

Derived from Unit

Attributes:

unitNameCV : UnitNameCV (required)
enumeration {*mL* | *cc* | *dL* | *L* | *uL* | *nL* | *pL* | *fL* | *other*}

2.1.16 Protocol

Provides a relatively immutable class, **Protocol**, that can describe a generic laboratory procedure or analysis algorithm, for example, and an instance class,

ProtocolApplication, which can describe the actual application of a protocol. The **ProtocolApplication** provides values for the replaceable parameters of the **Protocol** and, through the **Description** association of **Describable**, any variation from the **Protocol**.

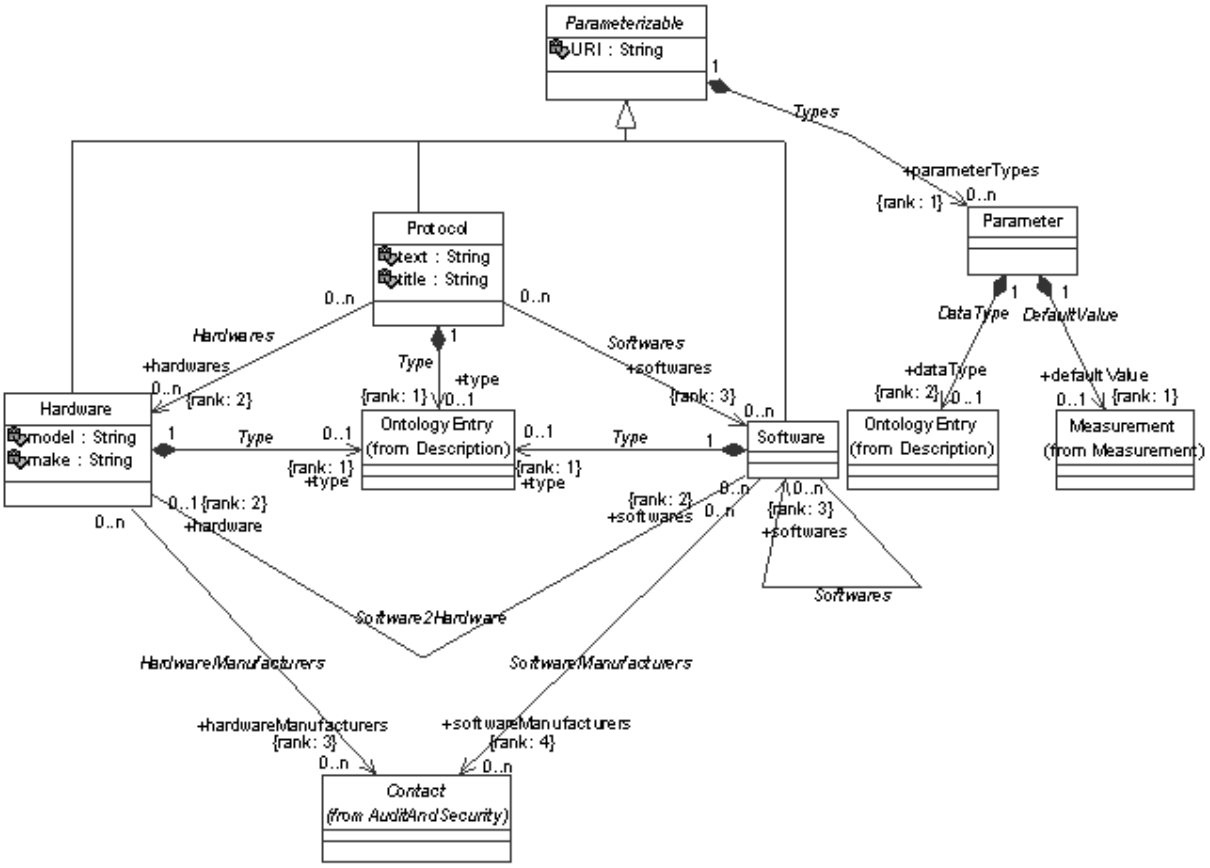

Figure 20: Parameterizable

Parameterizable

The **Parameterizable** interface encapsulates the association of **Parameters** with **ParameterValues**.

Derived from Identifiable

Associations:

parameterTypes : Parameter (0..n)
The description of the parameters for the **Parameterizable** class instance.

Attributes:

URI : String (optional)
Where an instantiated **Parameterizable** is located.

Protocol

A **Protocol** is a parameterizable description of a method. **ProtocolApplication** is used to specify the **ParameterValues** of it's **Protocol's Parameters**.

Derived from Parameterizable

Associations:

softwares : Software (0..n)
Software used by this Protocol.

hardwares : Hardware (0..n)
Hardware used by this protocol.

type : OntologyEntry (0..1)
The type of a **Protocol**, a user should provide/use a recommended vocabulary. Examples of types include: RNA extraction, array washing, etc...

Attributes:

text : String (optional)
The text description of the **Protocol**.

title : String (optional)
The title of the **Protocol**

Hardware

Hardware represents the hardware used. Examples of **Hardware** include: computers, scanners, wash stations etc...

Derived from Parameterizable

Associations:

hardwareManufacturers : Contact (0..n)
Contact for information on the **Hardware**.

softwares : Software (0..n)
Associates **Hardware** and **Software** together.

type : OntologyEntry (0..1)
The type of a piece of **Hardware**. Examples include: scanner, wash station...

Attributes:

model : String (optional)
The model (number) of a piece of hardware.

make : String (optional)
The make of the **Hardware** (its manufacturer).

Software

Software represents the software used. Examples of **Software** include: feature extraction software, clustering software, etc...

Derived from Parameterizable

Associations:

softwareManufacturers : Contact (0..n)
Contact for information on the software.

hardware : Hardware (0..1)
Associates **Hardware** and **Software** together.

softwares : Software (0..n)
Software packages this software uses, i.e. operating system, 3rd party software packages, etc.

type : OntologyEntry (0..1)
The type of a piece of **Software**. Examples include: feature extractor...

Parameter

A **Parameter** is a replaceable value in a **Parameterizable** class. Examples of **Parameters** include: scanning wavelength, laser power, centrifuge speed, multiplicative errors, the number of input nodes to a SOM, and PCR temperatures.

Derived from Identifiable

Associations:

dataType : OntologyEntry (0..1)
The type of data generated by the parameter i.e. Boolean, float, etc...

defaultValue : Measurement (0..1)
Allows the optional specification of a default values and the unit for the Parameter

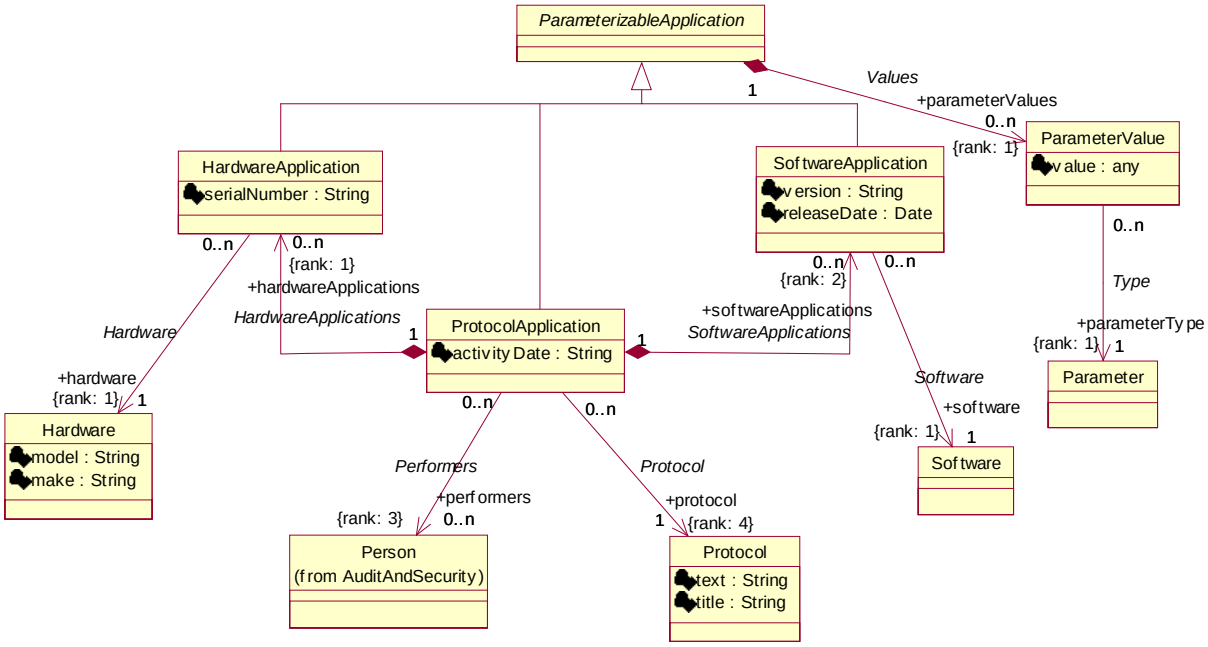

Figure 21: ParameterizableApplication

ParameterizableApplication

The interface that is the use of a **Parameterizable** class.

Derived from Describable

Associations:

parameterValues : ParameterValue (0..n)
The **parameter** values for this **ParameterizableApplication**.

ProtocolApplication

The use of a protocol with the requisite **Parameters** and **ParameterValues**.

Derived from ParameterizableApplication

Associations:

performers : Person (0..n)
The people who performed the protocol.

protocol : Protocol (1..1)
The protocol that is being used.

softwareApplications : SoftwareApplication (0..n)
The use of software for the application of the protocol.

hardwareApplications : HardwareApplication (0..n)
The use of hardware for the application of the protocol.

Attributes:

activityDate : String (required)
When the protocol was applied.

HardwareApplication

The use of a piece of hardware with the requisite **Parameters** and **ParameterValues**.

Derived from ParameterizableApplication

Associations:

hardware : Hardware (1..1)
The underlying hardware.

Attributes:

serialNumber : String (optional)
Manufacturer's identifier for the **Hardware**.

SoftwareApplication

The use of a piece of software with the requisite **Parameters** and **ParameterValues**.

Derived from ParameterizableApplication

Associations:

software : Software (1..1)
The underlying software.

Attributes:

version : String (optional)
The version of the software.

releaseDate : Date (optional)
When the software was released.

ParameterValue

The value of a **Parameter**.

Derived from Extendable

Associations:

parameterType : Parameter (1..1)
The parameter this value is for.

Attributes:

value : any (optional)
The value of the parameter. Will have the datatype of its associated **Parameter**.

2.1.17 QuantitationType

This Package defines the classes for quantitations, such as measured and derived signal, error, and pvalue. The subclasses of **StandardQuantitationType** will be the best fit from **FeatureExtraction** or **Transformation Protocol** for the values obtained. Other values can be specified using **SpecializedQuantitationType**.

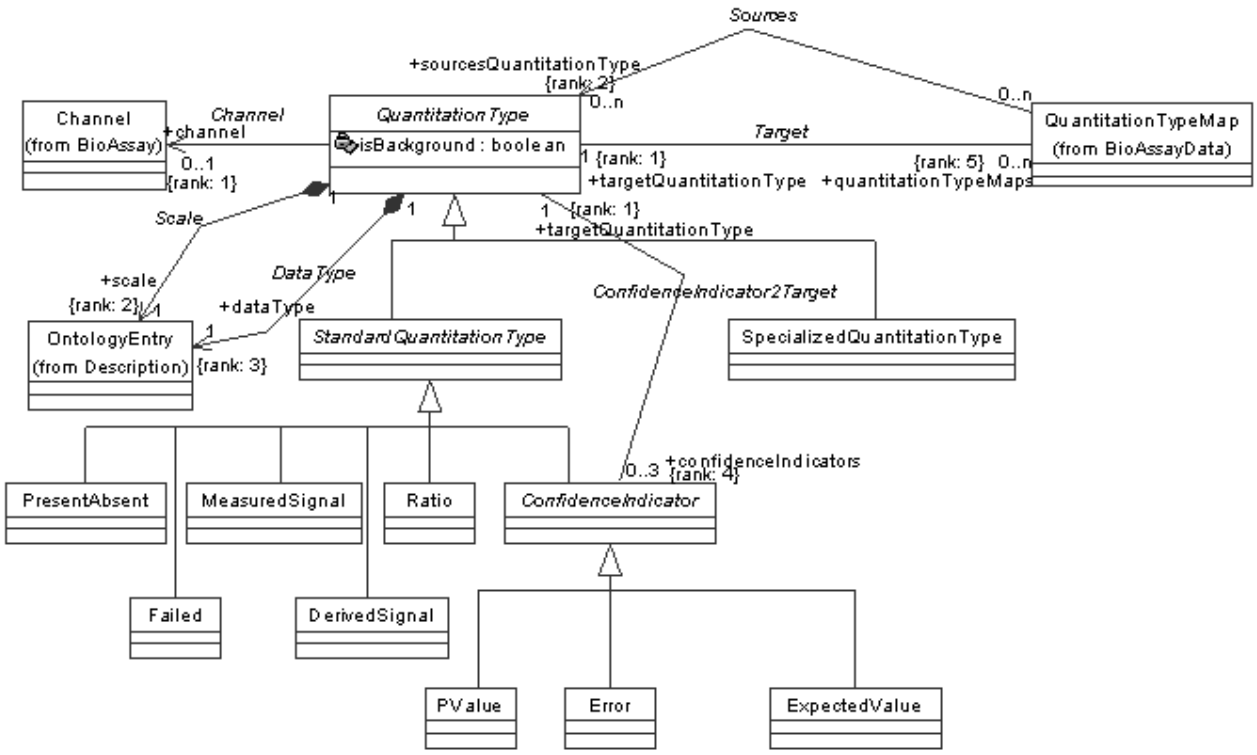

Figure 22: QuantitationType

QuantitationType

A method for calculating a single datum of the matrix (e.g. raw intensity, background, error).

Derived from Identifiable

Associations:

channel : Channel (0..1)
The optional channel associated with the **QuantitationType**.

dataType : OntologyEntry (1..1)
The specific type for the quantitations. From a controlled vocabulary of {float, int, boolean, etc.}

scale : OntologyEntry (1..1)
Indication of how to interpret the value. From a suggested vocabulary of {LINEAR | LN | LOG2 |LOG10 | FOLD_CHANGE | OTHER}

confidenceIndicators : ConfidenceIndicator (0..3)
The association between a **ConfidenceIndicator** and the **QuantitationType** its is an indicator for.

quantitationTypeMaps : QuantitationTypeMap (0..n)
The QuantitationType whose value will be produced from the values of the source QuantitationType according to the Protocol.

Attributes:

isBackground : boolean (required)
Indicates whether the quantitation has been measured from the background or from the feature itself.

StandardQuantitationType

Superclass for the named quantitation type. Useful for mapping to those languages that can use a fly-weight for processing the subclasses.

Derived from QuantitationType

DerivedSignal

A calculated measurement of the intensity of a signal, for example, after a transformation involving normalization and/or replicate **DesignElements**. Of type float.

Derived from StandardQuantitationType

MeasuredSignal

Best measure from feature extraction as to the presence and intensity of the signal. Of type float.

Derived from StandardQuantitationType

PresentAbsent

Indicates relative presence or absence. From the enumeration AbsoluteCallTypeEnum {Present | Absent | Marginal | No call} or ComparisonCallTypeEnum {Increase I Marginal Increase | Decrease | Marginal Decrease |No change | No Call | Unknown }, as specified by the dataType.

Derived from StandardQuantitationType

Ratio

The ratio of two or more signals, typically between two channels. Of type float.

Derived from StandardQuantitationType

Failed

Values associated with this QuantitationType indicate a failure of some kind for a particular DesignElement for a BioAssay. Of type boolean.

Derived from StandardQuantitationType

ConfidenceIndicator

Indication of some measure of confidence for a standard quantitation type.

Derived from StandardQuantitationType

Associations:

targetQuantitationType : QuantitationType (1..1)
The association between a **ConfidenceIndicator** and the **QuantitationType** its is an indicator for.

Error

Error measurement of a quantitation. Of type float.

Derived from ConfidenceIndicator

ExpectedValue

Indication of what value is expected of the associated standard quantitation type.

Derived from ConfidenceIndicator

PValue

Measurement of the accuracy of a quantitation. Of type float.

Derived from ConfidenceIndicator

SpecializedQuantitationType

User defined quantitation type.

Derived from QuantitationType

3 XML Platform Specific Model

3.1 MicroArray and GeneExpression Markup Language (MAGE-ML)

The MAGE-ML model defines the elements for supporting gene expression data.

Because the exchange of gene expression data can be abstracted from the source from which it was obtained, it can be represented by XML files, which are both human readable and machine readable. This facilitates an independence between the export and the import of the gene expression data as illustrated below.

NOTE: Although XML documents are human readable a question has been raised as to its value when the documents produced are in the 10-100 megabyte range. It has proved to be useful in two ways. The first is that when problems occur, because it is text based, tools such as Perl can be used to write scripts to discover facts based on the symptoms of the problem and/or the parsing code can be modified to examine the relevant elements of the XML more closely. The second benefit is in prototyping conversions from current scanning technologies. Because these are feature based, it is only necessary to map the output for a few illustrative features, and use that to verify correctness. Once the correctness of the conversion is established then the full document containing all the features can be reliably produced.

Ad hoc queries, when the XML files are directly accessible, can take advantage of the suite of W3C recommendations, including XSLT or XMLQuery. Queries against repositories could be specified a number of ways, including through an IDL interface that had as its query language either of the above choices or OQL based on MAGE-OM.

The DTD file, MAGE-ML.dtd, is generated from MAGE-OM from a fixed set of rules. In one area, **BioAssayData**, further modifications were made to offer alternatives and efficiency to the parsing. This will be discussed below.

3.1.1 General

The vocabulary of MAGE-ML is organized into sub-vocabularies in such a way that the sub-vocabularies are independent of each other. These sub-vocabularies are driven by the packages and **Identifiable** classes of the MAGE-OM, which correspond to discreet groupings of events and results of Gene expression experiments. This will allow a valid XML document to contain the data from an individual sub-vocabulary, such as **BioMaterial** or **ArrayDesign**, or to contain any combination of these sub-vocabularies, such as all the **BioAssay** and **BioAssayData** for an experiment. Implementations may impose additional ordering, such as **ArrayDesigns** before their **Arrays,** or they may require that they be exported to separate files.

The independence of sub-vocabularies is possible through the use of reference elements to link data from two sub-vocabularies within a document where the reference elements' identifier attribute can be matched to the object with that same identifier. That is, a **Reporter** or **CompositeSequence** element can be linked to its **BioSequence(s)** by using the identifier attribute of its **BioSequence_ref(s)**. Further, an import implementation is free to use the **identifier** attribute of a ***_ref** element to form an alternate key in its database and, for instance, either find that the biological

sequence already exists in its database, or, optionally create a place holder for it. An implementation is also free to consider it an error if a referenced element doesn't exist in the document or doesn't exist in its database.

Export implementations can decide to what extent that they export references without providing the referenced element in the same document. Export implementations can choose to allow exports which contain unresolved reference elements, to allow a subset of reference elements to be unresolved, or to not allow any unresolved references in a document.

Documentation for the MAGE-ML model is equivalent to the documentation of the MAGE-OM model under the mapping rules and will not be duplicated below.

3.1.2 Mapping from MAGE-OM to MAGE-ML

One of the design principals of the model was to keep it straight-forward so that the translation to XML would be as simple as possible. These mapping rules are not meant to be extendable to arbitrary models, nor were they designed to do more than map MAGE-OM to this DTD. The data-centric (as opposed to process-centric) nature of gene expression data allowed the UML model to not need advanced modeling constructs. This allowed the parsing of the XMI for this model to focus on the narrow range of elements used, primarily the elements representing the Model, the Packages, the Classes, the Associations, the DataTypes, and the ExtensionMechanism (for the documentation and the constraints).

Because the target audience is a science-based, it was decided to not use name spacing to keep implementation of the parsing of documents in the MAGE-ML format as straight-forward as possible. The generating code also correctly creates the reference elements for the classes that derive from Identifiable. Entities are used to encapsulate subclassing. To create the content list, the names of association end roles, adorned with suffixes to indicate their aggregation and cardinality, were used to create container elements (described below). Initially, only those end roles whose names were different than their class were going to be generated but by doing this for all roles code can be easily generated to implement parsing into another platform, a future goal. The parsing will be able to take advantage of the strict alternation between elements representing classes and elements representing roles.

There is a need for an element to contain all the elements not in an association where the aggregation is Composite. This element, MAGE-ML, can be seen as representing the Model and would be specified as the DOCTYPE in the XML documents. Rather than having all these independent elements directly in the content list of MAGE-ML element, elements are created that represent the packages. The MAGE-ML element has these package elements in its content list and each of these package elements has the container lists for these independent classes.

Java was used to implement this generating algorithm and the code is included as *GeneratingCode.zip dtc/*2002-09-05).

Generating Algorithm

There are two major steps in the generation of the DTD.

The first simply creates a list of subclasses of CreateFile, where each class, package and the model itself is represented on this list. The information for each of the

CreateFiles is obtained from parsing the XMI file and gathering all the information that may be in different element subtrees of the XMI file into the CreateFiles variables that are used to generate the data structures in the target platform. This collection of files is seen as another representation of the model. This list of CreateFiles can then be passed to code to generate the data structures for a particular platform.

The second step takes these CreateFiles and passes them to a class that iterates through the list and creates a wrapper class, derived from WriteDTDElement, for each of the CreateFiles. These classes then create the entity, elements and attlist declarations needed for their CreateFile then writes them out to the DTD file. An additional initial step orders the CreateFiles so that the entities representing base classes will be printed in the proper dependency order and so that the packages and their classes will be grouped together.

The description of the algorithm below does not describe the implementation of the generating code but rather describes how the entity, element and attribute declarations are generated for the Model, Packages and Classes specified in the model. First we list some general patterns then show examples of how classes, associations and attributes in the model are mapped to entity, element and attribute declarations in the DTD.

Content Lists:

In the discussion below, content lists are formed from the names of the association end roles that are navigable from the class. They are capitalized and have the suffix '**_assn**' added. Additionally they are adorned to indicate if they are referenced, as opposed to composite, and to indicate the cardinality. So for an association end named **characteristics**,

Composite and single valued: **Characteristics_assn**.

Composite and multi-valued: **Characteristics_assnlist**.

Referenced and single valued: **Characteristics_assnref**.

Referenced and multi-valued: **Characteristics_assnreflist**.

The ordering of the content lists for elements representing classes is provided from the model by a constraint on the association end.

Attributes:

In the discussion below, attributes are formed of three parts:

The name of the attribute.

The type, 'CDATA' unless the attribute is an enumeration, in which case the enumeration produces a choice list for the attribute.

The value, either the initial value, if specified, or '#REQUIRED' if it is marked as required in the model or '#IMPLIED' if it is optional.

Entity Declarations:

Entities are only generated for those classes that are base classes in the model. These entities are then used in the context of the base class in two ways. One, for those roles this base class is the end class, the ***_class** or ***_ref** (where ***** is replaced by the base class name) entity is used and is a choice group of all the instantiable classes of this base class for ***_class** and for the ***_ref**. Two, the ***_content** and ***_attrs** entities

are used by the instantiable classes in the content list and attribute declaration, respectively.

If the base class is the end class for a role, then the ***_class** will be generated, and if it has a role in which it isn't aggregated by composition, then the ***_ref** entity is also generated. So for the abstract base class **BioAssay** the following entities are created because **BioAssay** is associated by reference from **Experiment**:

```
<!ENTITY % BioAssay_classes "PhysicalBioAssay |
        DerivedBioAssay |
        MeasuredBioAssay" >
<!ENTITY % BioAssay_ref "PhysicalBioAssay_ref |
        DerivedBioAssay_ref |
        MeasuredBioAssay_ref" >
```

The ***_content** and ***_attrs** are always created for base classes. The ***_content** is formed with the adorned names of the association end roles that are marked as navigable from it. The ***_attrs** are formed from the name of the attributes of the class with the appropriate type and value (note that since **BioAssay** inherits from **Identifiable**, it uses its content and attribute entities):

```
<!ENTITY % BioAssay_content "(%Identifiable_content;),
        Channels_assnreflist?,
        BioAssayFactorValues_assnreflist?" >
<!ENTITY % BioAssay_attrs "%Identifiable_attrs;" >
```

Element Declarations:

For the class representing the model, MAGE-ML, the following element is generated:

```
<!ELEMENT MAGE-ML ((%Identifiable_content;),
        AuditAndSecurity_package?,
        Description_package?,
        Measurement_package?,
        BQS_package?,
        BioEvent_package?,
        Protocol_package?,
        BioMaterial_package?,
        BioSequence_package?,
        DesignElement_package?,
        ArrayDesign_package?,
        Array_package?,
        BioAssay_package?,
        QuantitationType_package?,
        BioAssayData_package?,
        Experiment_package?,
        HigherLevelAnalysis_package?) >
```

The content list is composed of the name of the packages, adorned with '_package'. This corresponds to the element declaration that the packages will generate. The code that generates this declaration assures that the order of the content list will remain the same. It was decided that this should also inherit the content of Identifiable, for one, this allows for high level descriptions of the file.

For classes representing the packages, the following element is generated:

```
<!ELEMENT Array_package (ArrayGroup_assnlist?,
        Array_assnlist?,
        ArrayManufacture_assnlist?) >
```

Where its content are containers for any class in the package that is not in any navigable end association role that is aggregated by composition. The exception to this is that any class that has a base class that fulfills this condition is represented by

the base class on this list. The name of the container elements are the class name with the suffix '_assnlist'. The code that generates this content list assures that the container elements always appear in the same order on the content list.

For the classes representing the classes in the model there is not only the element representing itself that needs to be generated, but the container elements used elsewhere in content lists. There is potentially an element generated for the container on the package content list:

```
<!ELEMENT ArrayGroup_assnlist (ArrayGroup+) >
```

For each navigable association role that it is the end class for, the container element for that role needs to be generated:

```
<!ELEMENT ContainedFeatures_assnreflist (Feature_ref+) >
<!ELEMENT ControlFeatures_assnreflist (Feature_ref+) >
<!ELEMENT ControlledFeatures_assnreflist (Feature_ref+) >
<!ELEMENT Feature_assnref (Feature_ref) >
<!ELEMENT Features_assnlist (Feature+) >
```

And if it has any association in which it is referenced, the reference element and attribute need to be generated:

```
<!ELEMENT Feature_ref EMPTY >
<!ATTLIST Feature_ref identifier CDATA #REQUIRED
       name CDATA #IMPLIED >
```

Finally, the actual element declaration for the class is generated using the rule above to generate names on the content list if the class is not abstract:

```
<!ELEMENT Feature ((%DesignElement_content;),
       ControlFeatures_assnreflist?,
       ControlledFeatures_assnreflist?,
       Position_assn?,
       Zone_assnref?,
       FeatureLocation_assn?) >
```

The order of the content list is based on the constraint on the association end in the model that is of the form 'rank: n' where n is where it appears on the content list after the base class content, if any. If two or more associations have the same rank, they will form a choice group:

```
<!ELEMENT CompositeSequence ((%DesignElement_content;),
       BiologicalCharacteristics_assnreflist?,
       (ReporterCompositeMaps_assnreflist? |
              CompositeCompositeMaps_assnreflist?)) >
```

If the rank is –1, the associations with that rank will form a set of choice groups that allow any ordering of them. This is to take care of a special case (see *Transforming BioDataValues*, section 4.1.4).

Attribute Declarations:

The attribute declaration for the MAGE-ML class are simply the Identifiable attributes:

```
<!ATTLIST MAGE-ML %Identifiable_attrs; >
```

There are no attributes for the package classes.

The attribute declaration for the classes that represent classes in the model are formed from the base class entity, if it exists, and then the attributes of the class using the rules above:

```
<!ATTLIST Measurement %Extendable_attrs;
       type (absolute|
           change) "absolute"
       value CDATA #REQUIRED
       kindCV (time|
           distance|
           temperature|
           quantity|
           mass|
           volume|
           concentration|
           other) "other"
       otherKind CDATA #IMPLIED >
```

Relationship to XMI DTD Generating Rules

We have used existing tools (from Rational Rose and Unisys) to generate MAGE.xmi which conforms to the XML Document Production specified in Chapter 6 of the XMI 1.1 specification. We then decided, however, to use a different mapping to produce the MAGE-ML.dtd than is specified in Chapter 4, **XML DTD Production** from that specification. Our reasons are the following.

The reasons, whose implications will be further discussed, were three. There was a need for simplicity and as little choice for an implementer as possible, related was that it should bear a resemblance to the two existing formats, GE ML.dtd and MAML.dtd, both of which had existing support. We also felt we needed multiplicities and ordering of elements called out. Lastly, we wanted to provide the documentation from the model as DTD comments, since, for many, that would be the only place they would see it. We found that anything in the DTD that offered choices, even if optional, caused confusion and made the actual use of the DTD harder to comprehend to the expected users of the format in the Gene Expression community.

We desired for simplicity of understanding, as in GEML.dtd and MAML.dtd, that the names of the elements and attributes were terms based solely on Gene Expression and Biology. This meant we did not want the necessary XMI DTD declarations that the XMI specification requires. We also decided, especially with the limited support for DTDs, not to use namespacing.

Another area where choice is offered by the XMI specification is the two entities **XMI.element.att** and **XMI.link.att**. These allow each element generated for a class to either represent the information for that class directly, and identified in one of three ways by **XMI.element.att** or to be a reference to the actual information for that class either in the same document or in some other XML document using the **XMI.link.att** which provides two ways to make the reference. This is illustrated in the XMI specification Section 3.8.3 where a Constraint element can be fully defined in more than one section of the DTD. Rather than offer this choice, we used the Identifiable class and reference elements (see section 1.1.1 and 1.1.2) to specify absolutely where in an XML document an element would be fully defined (see rules above) to eliminate any ambiguity.

XMI 1.1 allows both attributes and associations to be on both the content list and as attributes. As this is also potentially confusing we decided that attributes for this specification were simple enough so that we feel they can be in the DTD solely as attributes and the use of reference elements supercedes the need for including the association as an attribute of type **IDREFS**.

Calling out the multiplicities by creating elements that represents the association ends (see rules above, *Content Lists*) also helps to simplify the use of the format by eliminating choice and providing information to the user of type of the contained element, whether it is a list and whether it will be by reference. Also, in some cases, in processing the data it can provide greater efficiency to have the list ordered, for instance, if **CompositeSequences** are in the document, they must come before the **Reporters** that refer to them. Therefore, when processing the document, if the **CompositeSequence** is not found, then it is already known to be external. These association role elements also allow parsing to know when a list is beginning and when it is ended so that it is clear where initial processing and end processing can take place if desired.

There is also no generating rule for the documentation that is in the model in the XMI specification. We felt that the DTD would be of more value if the documentation were included.

3.1.3 Example Mapping

Using the model below, the rules above are followed to generate an example mapping to a DTD. Although the documentation of the model can't be shown in the diagram, the comments in the DTD are generated from it

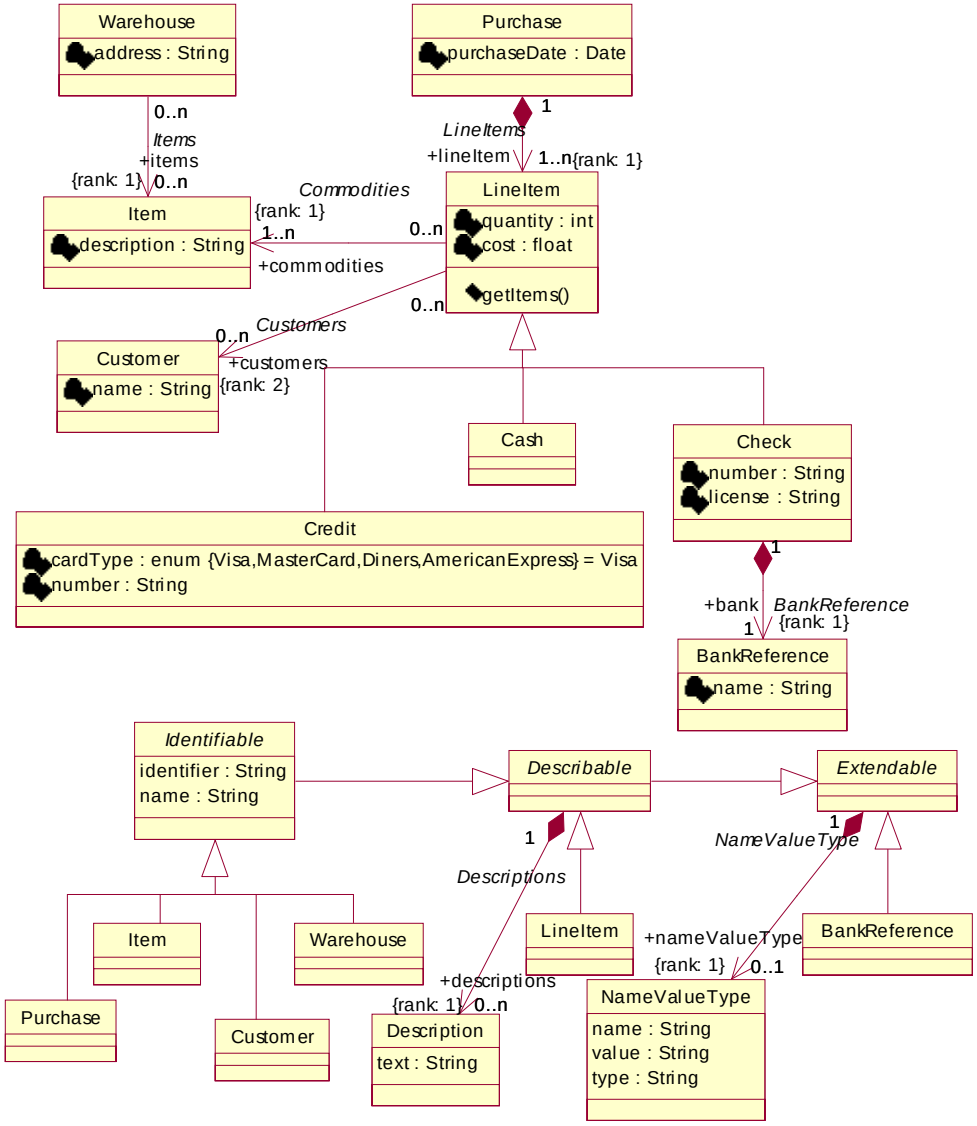

Figure 23: Example Mapping

Generated DTD

```
<!--
    Entities for Extendable

    Abstract class with a relationship to NameValueType for arbitrary
    property sets.

    Associations
        nameValueType: Asscoiation for property sets.
-->
<!ENTITY % Extendable_content "NameValueType_assn?" >
<!ENTITY % Extendable_attrs "" >
```

```
<!--
    Entities for Describable

    Abstract class with an association to Description.

    Associations
        Inherites associations from base class Extendable.

        descriptions: Descriptions for the instance of a Describable.

    Attributes
        Inherites attributes from base class Extendable.
-->
<!ENTITY % Describable_content "(%Extendable_content;),
       Descriptions_assnlist?" >
<!ENTITY % Describable_attrs "%Extendable_attrs;" >

<!--
    Entities for Identifiable

    An abstract class for classes that can be referenced.

    Associations
        Inherites associations from base class Describable.

    Attributes
        Inherites attributes from base class Describable.

        identifier: The unambiguous identifier for the instance.

        name: A potentially non-unique common name.
-->
<!ENTITY % Identifiable_content "(%Describable_content;)" >
<!ENTITY % Identifiable_attrs "%Describable_attrs;
       identifier CDATA #REQUIRED
       name CDATA #IMPLIED" >

<!--
    Purchasing_package

    Entities for classes in this package.
-->
<!--
    Entities for LineItem

    The information on the purchase of an item.

    Associations
        Inherites associations from base class Describable.

        commodities: The item purchased.

        customers: The person purchasing the item.

    Attributes
        Inherites attributes from base class Describable.

        quantity: How much or many of the items were purchased.

        cost: The total cost of the purchase of the item(s).
-->
<!ENTITY % LineItem_classes "LineItem |
       Cash |
       Credit |
       Check" >
<!ENTITY % LineItem_content "(%Describable_content;),
       Commodities_assnreflist,
       Customers_assnreflist?" >
<!ENTITY % LineItem_attrs "%Describable_attrs;
       quantity CDATA #REQUIRED
       cost CDATA #REQUIRED" >
```

```
<!--
    MAGE-ML

    The top-level element that contains the packages.  Each of the
    package elements contain the lists of independent elements in that
    package.
-->
<!ELEMENT MAGE-ML ((%Identifiable_content;),
       Purchasing_package?) >
<!ATTLIST MAGE-ML %Identifiable_attrs;  >

<!--
    Element and Attlist declarations for Description

    A text description.

    Attributes
        text: The description.
-->
<!ELEMENT Descriptions_assnlist (Description+) >

<!ELEMENT Description EMPTY >
<!ATTLIST Description text CDATA #REQUIRED >

<!--
    Element and Attlist declarations for NameValueType

    For specifying an arbitrary property.

    Attributes
        name: Name of property.

        value: Value of the property.

        type: Type of the property.
-->
<!ELEMENT NameValueType_assn (NameValueType) >

<!ELEMENT NameValueType EMPTY >
<!ATTLIST NameValueType name CDATA #REQUIRED
       value CDATA #REQUIRED
       type CDATA #IMPLIED >

<!--
    Purchasing_package

    The package for this model.
-->
<!ELEMENT Purchasing_package (Purchase_assnlist?,
       Warehouse_assnlist?,
       Item_assnlist?,
       Customer_assnlist?) >

<!--
    Element and Attlist declarations for BankReference

    A reference by name to a bank.

    Associations
        Inherites associations from base class Extendable.

    Attributes
        Inherites attributes from base class Extendable.

        name: The name of the bank.
-->
<!ELEMENT Bank_assn (BankReference) >

<!ELEMENT BankReference ((%Extendable_content;)) >
<!ATTLIST BankReference %Extendable_attrs;
       name CDATA #REQUIRED >
```

```
<!--
    Element and Attlist declarations for Customer

    A purchaser of items.

    Associations
        Inherites associations from base class Identifiable.

    Attributes
        Inherites attributes from base class Identifiable.

        name: The name of the customer.
-->
<!ELEMENT Customer_assnlist (Customer+) >

<!ELEMENT Customers_assnreflist (Customer_ref+) >

<!ELEMENT Customer_ref EMPTY >
<!ATTLIST Customer_ref identifier CDATA #REQUIRED
       name CDATA #IMPLIED >

<!ELEMENT Customer ((%Identifiable_content;)) >
<!ATTLIST Customer %Identifiable_attrs;
       name CDATA #REQUIRED >

<!--
    Element and Attlist declarations for Item

    A purchasable commodity

    Associations
        Inherites associations from base class Identifiable.

    Attributes
        Inherites attributes from base class Identifiable.

        description: The characteristics of the item.
-->
<!ELEMENT Item_assnlist (Item+) >

<!ELEMENT Commodities_assnreflist (Item_ref+) >
<!ELEMENT Items_assnreflist (Item_ref+) >

<!ELEMENT Item_ref EMPTY >
<!ATTLIST Item_ref identifier CDATA #REQUIRED
       name CDATA #IMPLIED >

<!ELEMENT Item ((%Identifiable_content;)) >
<!ATTLIST Item %Identifiable_attrs;
       description CDATA #REQUIRED >

<!--
    Element and Attlist declarations for LineItem

    The information on the purchase of an item.

    Associations
        Inherites associations from base class Describable.

        commodities: The item purchased.

        customers: The person purchasing the item.

    Attributes
        Inherites attributes from base class Describable.

        quantity: How much or many of the items were purchased.

        cost: The total cost  of the purchase of the item(s).
-->
<!ELEMENT LineItem_assnlist ((%LineItem_classes;)+) >
```

```
<!ELEMENT LineItem ((%LineItem_content;)) >
<!ATTLIST LineItem %LineItem_attrs; >

<!--
    Element and Attlist declarations for Cash

    A cash purchase.

    Associations
        Inherites associations from base class LineItem.

    Attributes
        Inherites attributes from base class LineItem.
-->
<!ELEMENT Cash ((%LineItem_content;)) >
<!ATTLIST Cash %LineItem_attrs; >

<!--
    Element and Attlist  declarations for Check

    Purchase made by check.

    Associations
        Inherites associations from base class LineItem.

        bank: An external reference to the bank the check is written to.

    Attributes
        Inherites attributes from base class LineItem.

        number: The check number.

        license: The license or other identification of the customer
    writing the check.
-->
<!ELEMENT Check ((%LineItem_content;),
       Bank_assn) >
<!ATTLIST Check %LineItem_attrs;
       number CDATA #REQUIRED
       license CDATA #IMPLIED >

<!--
    Element and Attlist declarations for Credit

    A purchase made by credit card.

    Associations
        Inherites associations from base class LineItem.

    Attributes
        Inherites attributes from base class LineItem.

        cardType: The type of credit card used.

        number: The credit card number
-->
<!ELEMENT Credit ((%LineItem_content;)) >
<!ATTLIST Credit %LineItem_attrs;
       cardType (Visa|
           MasterCard|
           Diners|
           AmericanExpress) "Visa"
       number CDATA #IMPLIED >

<!--
    Element and Attlist declarations for Purchase

    Contains all the things needed to know about a purchase.

    Associations
        Inherites associations from base class Identifiable.
```

```
        lineItem: The detailed information on the items to be purchased.

    Attributes
        Inherites attributes from base class Identifiable.

        purchaseDate: The day the item was purchased
-->
<!ELEMENT Purchase_assnlist (Purchase+) >


<!ELEMENT Purchase ((%Identifiable_content;),
       LineItem_assnlist) >
<!ATTLIST Purchase %Identifiable_attrs;
       purchaseDate CDATA #REQUIRED >

<!--
    Element and Attlist declarations for Warehouse

    The warehouse stores items.

    Associations
        Inherites associations from base class Identifiable.

        items: The items stored in the warehouse.

    Attributes
        Inherites attributes from base class Identifiable.

        address: The address of the warehouse.
-->
<!ELEMENT Warehouse_assnlist (Warehouse+) >


<!ELEMENT Warehouse ((%Identifiable_content;),
       Items_assnreflist?) >
<!ATTLIST Warehouse %Identifiable_attrs;
       address CDATA #IMPLIED >
```

3.1.4 Transforming BioDataValues

In the model there are two ways that **BioDataValues** can be represented: one subclass, **BioDataTuples**, corresponds to a set of relational tuples of dimension four, the three **BioAssayData** Dimensions and a **value**; the other subclass, **BioDataCube**, represents the data as a three-dimension array. The first is very good for transporting the data, the second is good for viewing the data from different perspectives.

Neither one ideally maps into the DTD by the above rules. In fact, the three dimensional array **cube** attribute of **BioDataCube** has no mapping rule. For the sake of argument we will create an element, **Data**, for it, with content #PCDATA. We describe the further transformation of the subclasses for **BioDataValues** here:

The DTD for BioDataValues by the mapping rules:

```
<!-- BioDataValues Entities-->

<!ENTITY % BioDataValues_classes "BioDataTuples|BioDataCube">
<!ENTITY % BioDataValues_content "%Extendable_content;" >
<!ENTITY % BioDataValues_attrs "%Extendable_attrs;" >
<!ENTITY % BioAssayData_content "%Identifiable_content;,
              SummaryStatistic_list?,
              BioAssayDimension_ref?,
              DesignElementDimension_ref?,
```

```
                QuantitationTypeDimension_ref?,
                (%BioDataValues_classes;)?"          >
<!Element BioDataTuples ((%Extendable_content;), BioAssayDatum_list?)>
<!ATTLIST BioDataTuples %Extendable_attrs>
<!ELEMENT BioAssayDatum_list (BioAssayDatum+)>
<!ELEMENT BioAssayDatum (BioAssay_ref, DesignElement_ref, QuantitationType_ref)>
<!ATTLIST BioAssayDatum value CDATA #REQUIRED>
<!ELEMENT BioDataCube ((%Extendable_content;), Data)>
<!ATTLIST BioDataCube %Extendable_attrs>
<!ELEMENT Data (#PCDATA)>
```

Transforming BioDataCube

There is a desire that the data values, themselves, can be recorded as whitespace delimited #PCDATA, as above, or in an external file in a specified for mat, such as whitespace delimited, tab delimited, NetCDF, etc. To be compliant for this RFP the data must be either **BioAssayTuples** or **InternalData** and the dimensions chosen must be in the order of **BioAssayDimension**, **DesignElementDimension** and then **QuantitationTypeDimension**. To be MIAME compliant there is no restriction in ordering and the data can be external to the XML.

Transforming BioDataTuples

With the representation of **BioDataTuples** above there is a redundancy of data and that the data can be unordered, making it difficult to parse efficiently. All of the **BioDataTuple_list** would need to be read to know how to organize the data. To facilitate the requirements of parsing, the contents of **BioAssayDatum** will be 'turned inside out'. By having a **BioAssayTuple**, the **BioAssay_ref** is only in the XML once and it is assured that all the data within the **BioAssayTuple** is the data for that **BioAssay**. This is also useful in that it can capture the data from one **MeasuredBioAssay**, i.e. the results of one Hybridization, in a single XML file and all the **MeasuredBioAssay** for an **Experiment** can then be in separate files. This eases the importation of the potentially thousands of **BioAssays** that can be associated with a single **Experiment**.

Transformation Generating Code

The code to produce this transformation takes the list of CreateFiles generated by the parsing of the XML.

1. It searches for the representations of the original classes involved in the transformation, **BioAssayData**, **BioDataCube**, **BioDataTuples**, and **BioAssayDatum**. **BioAssayDatum** is removed from the CreateFile list.
2. The dimension associations of BioAssayData are transformed to have cardinalities 0..1 from 1..1
3. The CreateFiles to represent the **DataInternal** and **DataExternal** elements are created and their attributes are added. **DataInternal** also has #PCDATA added for its content list for the white-space delimited data.
4. **BioDataCube** has the associations 1..1 to **DataInternal** and **DataExternal** added with a rank of 2 which will cause the generating code to add them as a Choice group to the content list.
5. The tag-delimited form of the data representation, BioAssayTuples, now is transformed. The association to BioAssayDatum is removed.
6. The CreateFile to represent the **Datum** element is created with a **value** attribute.

7. The CreateFile to represent the **QuantitationTypeTuple** is created with an association 1..1 to **Datum** and the association 1..1 to **QuantitationType** from the **BioAssayDatum**.

8. The CreateFile to represent the **DesignElementTuple** is created with an association 1..n to **QuantitationTypeTuple** and the association to **DesignElement** from the **BioAssayDatum**.

9. The CreateFile to represent the **BioAssayTuple** is created with an association 1..n to **DesignElementTuple** and the association 1..1 to **BioAssay** from the **BioAssayDatum**.

10. An association 1..n to **BioAssayTuple** is added to **BioAssayTuples**.

11. Lastly, the new CreateFiles are added to the list of CreateFiles.

The code to generate the DTD now treats these new classes as if they came from the UML model directly. The class to do the transformation is is specified as a command line option or from a configuration file.

The Transformed Elements:

```
<!ELEMENT BioDataCube ((%BioDataValues_content;),
              (DataInternal_assn | DataExternal_assn)) >
<!ATTLIST BioDataCube %BioDataValues_attrs; >

<!ELEMENT DataExternal_assn (DataExternal) >

<!ELEMENT DataExternal EMPTY >
<!ATTLIST DataExternal dataFormat CDATA "whitespace"
       dataFormatInfoURI CDATA #IMPLIED
       filenameURI CDATA #REQUIRED >

<!ELEMENT DataInternal_assn (DataInternal) >

<!ELEMENT DataInternal (#PCDATA) >

<!ELEMENT BioDataTuples ((%BioDataValues_content;),
       BioAssayTuples_assnlist?) >
<!ATTLIST BioDataTuples %BioDataValues_attrs; >

<!ELEMENT BioAssayTuples_assnlist (BioAssayTuple+) >

<!ELEMENT BioAssayTuple (BioAssay_assnref,
       DesignElementTuples_assnlist) >

<!ELEMENT DesignElementTuples_assnlist (DesignElementTuple+) >

<!ELEMENT DesignElementTuple (DesignElement_assnref,
       QuantitationTypeTuples_assnlist) >

<!ELEMENT QuantitationTypeTuples_assnlist (QuantitationTypeTuple+) >

<!ELEMENT QuantitationTypeTuple (QuantitationType_assnref,
       Datum_assn) >

<!ELEMENT Datum_assn (Datum) >

<!ELEMENT Datum EMPTY >
<!ATTLIST Datum value CDATA #REQUIRED >
```

4 Applicability to Sage and Proteomics

MAGE-OM will store essentially any array based data with very few if any modifications. Applications other than gene expression for which MAGE-OM would be applicable include: protein diagnostics, genotyping, and sectioned tissue analysis. MAGE-OM is not presently capable of storing non-array based expression profiling technologies such as SAGE, and extensive modifications would be necessary to support this.

A References

Object Management Group. 2000. Gene Expression RFP. OMG Document lifesci/00-03-09.

Object Management Group. 2001. Gene Expression RFP Response, Joint Revised Submission, European Bioinformatics Institute and Rosetta Inpharmatics. OMG Document lifesci/01-08-01.

Object Management Group. 2001. Gene Expression RFP Response (DRAFT), Joint Revised Submission, European Bioinformatics Institute, NetGenics, and Rosetta Inpharmatics. OMG Document lifesci/01-06-02.

Object Management Group. 2000. Gene Expression RFP response, Initial Submission, EMBL-EBI (European Bioinformatics Institute). OMG Document lifesci/00-11-16.

Object Management Group. 2000. Gene Expression RFP Response, NetGenics, Inc. OMG Document lifesci/00-11-06.

Object Management Group. 2000. Rosetta Inpharmatics Initial Submission regarding the Gene Expression RFP (with errata). OMG Document lifesci/00-12-04.

Annotations Working Group, Microarray Gene Expression Database Group. 2001. Minimum Information About a Microarray Experiment – MIAME Version 1.0. http://www.mged.org/Annotations-wg/index.html

MicroArray Format Working Group, Microarray Gene Expression Database Group. 2001. Microarray Markup Language (MAML) Specification Primer. http://www.ncbi.nlm.nih.gov/geo/maml.

Object Management Group. 2001. Bibliographic Query Service Specification. OMG Document dtc/01-04-05.

Object Management Group. 1999. Biomolecular Sequence Analysis RFP response. OMG Document lifesci/99-10-01.

Object Management Group. 2000. Biomolecular Sequence Analysis Entities (BSANE) RFP response. OMG Document lifesci/00-12-16.

Object Management Group. 1998. Interoperable Naming Service. OMG Document orbos/98-10-11.

Object Management Group. 1998. OMG IDL Style Guide. OMG Document ab/98-06-03.

Object Management Group. 2000. Query Service Specification. OMG Document formal/00-06-23.

Anderson, Richard, et al. 2000. Professional XML. Wrox Press Ltd. ISBN: 1-861003-11-0.

Apparao, Victor, et al, eds. 1998. Document Object Model (DOM) Level 1 Specification, W3C Recommendation. 1 October 1998.

Bairoch, Amos, et al. 1997. The Swiss-Prot Protein Sequence Data Bank User Manual. Release 35; November 1997.

Curran, John, and Leslie Daigle, chairs. 2001. Uniform Resource Names (urn). http://www.ietf.org/html.charters/urn-charter.html

Daniel, Ron Jr., Steve Rose, and Eve Maler, eds. 2000. XML Pointer Language (XPointer) Version 1.0, W3C Candidate Recommendation. 7 June 2000.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley. ISBN: 0-201-63361-2.

Haynes, Paul A., John R. Yeats III. 2000. Proteome profiling—pitfalls and progress. Yeast, Vol. 17, 81-87, 2000.

Hughes, Timothy R., et al. 2000. Functional Discovery via a Compendium of Expression Profiles. Cell, Vol. 102, 109-126. July 7, 2000.

International Organization for Standardization. 1988. International Standard for representation of dates and times. http://www.iso.ch/markete/8601.pdf

McArthur, Douglas C. 1999. Rosetta Inpharmatics Gene Expression RFI Response. OMG Document lifesci/99-08-11.

Megginson, David, Peter Murray-Rust, Tim Bray, XML-DEV community. 1998. SAX 1.0: The Simple API for XML. http://www.megginson.com/SAX/SAX1/

National Center for Biotechnology Information, et al. 1997. The DDJB/EMBL/GenBank Feature Table: Definitions. Version 2.0. December 15, 1997.

NCBI Taxonomy database, http://www.ncbi.nlm.nih.gov/Taxonomy/tax.html.

Wolf , Misha, and Charles Wicksteed. 1997. Date and Time Formats. http://www.w3.org/TR/NOTE-datetime

World Wide Web Consortium. 2000. Extensible Markup Language (XML) 1.0 (Second Edition). http://www.w3.org/TR/2000/REC-xml-20001006

World Wide Web Consortium. 2000. XML Schema. http://www.w3.org/XML/Schema.html

World Wide Web Consortium. 2001. XML Schema Part 2: Datatypes. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502

B Minimum Information About a Microarray Experiment

The Microarray Gene Expression Database Group (MGED) has, in parallel to the OMG-LSR effort, developed a set of requirements for Gene Expression data interchange and MicroArray Markup Language (MAML) to fulfill those requirements. These were presented to the OMG in response to the Gene Expression RFP (lifesci/00-03-09) by EBI (lifesci/00-11-16). EBI and MGED are working with the other submitters to develop MAGE in this combined submission. The submitters feel that MAGE must not only allow fulfillment of the requirements of the Gene Expression RFP but also allow fulfillment of the requirements of MIAME. MGED as provided the text in Section B.1 to B.9 for inclusion in this document. Comments on compliance are indicated by italics in the text.

B.1 MIAME Requirements

The goal of the MIAME is to specify the minimum information that must be reported about an array based gene expression monitoring experiment in order to ensure the interpretability of the results, as well as potential verification by third parties. This is to facilitate establishing repositories and a data exchange format for array based gene expression data. The MGED group will encourage the scientific journals and funding agencies to adopt policies requiring data submissions to repositories, once MIAME compliant repositories and annotation tools are established.

B.2 Introduction:

The definition of the minimum information is aimed at cooperative data providers, and is not intended to close possible loopholes in not providing the information.

Among the concepts in the definition is a list of ‘qualifier, value, source’ triplets, by means of which we would like to encourage the authors to define their own qualifiers and provide the appropriate values so that the list as the whole gives sufficient information to fully describe the particular part of the experiment. The idea stems from the information sciences where ‘qualifier’ defines a concept, and ‘value’ contains the appropriate instance of the concept .‘Source’ is either user defined, or a reference to an externally defined ontology or controlled vocabulary, such as the species taxonomy database. The judgment regarding the necessary level of detail is left to the data providers. In the future these ‘voluntary’ qualifier lists may be gradually substituted by predefined fields, as the respective ontologies are developed.

Parts of the MIAME can be provided as references or links to pre-existing and identifiable descriptions. For instance for commercial or other standard arrays, all the required information should normally be provided only once by the array provider and referenced thereafter by the users. Standard protocols should also normally be provided only once. It is necessary that either a valid reference or the information itself is provided for every experiment set.

B.3 Definition:

The minimum information about a published microarray based gene expression experiment should include a description of the:

1. Experimental design: the set of hybridisation experiments as a whole
2. Array design: each array used and each element (spot) on the array
3. Samples: samples used, extract preparation and labeling
4. Hybridisations: procedures and parameters
5. Measurements: images, quantitation, specifications
6. Normalisation controls: types, values, specifications

An additional section dealing with the data quality assurance will be added in the next MIAME release.

The following details should be provided for each array, sample, hybridisation and measurement in the experiment set:

B.4 Experimental design: the set of hybridisation experiments as a whole

This section describes the experiment, which may consist of one or more hybridisations, as a whole. Normally 'experiment' should include a set of hybridisations which are inter-related and address a common question. For instance, it may be all the hybridisations related to research published in a single paper.

(a) author (submitter), laboratory, contact information, links (URL), citations
Experiment has an association (Provider) to Contact, Contact provides this information

(b) type of the experiment - maximum one line, for instance:
ExperimentalDesign has association (Type) to OntologyEntry where these can be specified.

(i) normal vs. diseased comparison

(ii) treated vs. untreated comparison

(iii) time course

(iv) dose response

(v) effect of gene knock-out

(vi) effect of gene knock-in (transgenics)

(vii) shock (multiple types possible)

(c) experimental variables, i.e. parameters or conditions tested (e.g., time, dose, genetic variation, response to a treatment or compound)
ExperimentalDesign has a list of ExperimentalFactors

(d) single or multiple hybridisations.

(i) For multiple hybridisations:

1. serial (yes/no)

    a. type (e.g., time course, dose response)
    through ExperimentalDesign ontology entry

2. grouping (yes/no)

    a. type (e.g., normal vs. diseased, multiple tissue comparison)
    through ExperimentalFactors

Relationships between all the samples, arrays and hybridisations in the experiment. Each sample, each array, and each hybridisation should be given a unique ID, and all the relationships should be listed (with appropriate comments where necessary). For instance:

Samples:	S1, S2, S3
Extracts:	e1S1, e1S2, e1S3
Labeled extracts:	l1e1S1, l2e1S1, l1e1S2, l1e1S3
Array types:	T1, T2
Arrays:	a1T1, a2T1, a3T2
Hybridisations:	H1 is l1e1S1+l1e1S2 on a1T1 H2 is l1e1S2+l1e1S3 on a2T1 H3 is l2e1S1+l1e1S2 on a3T2

Note that detailed descriptions of each sample, array and hybridisation are provided in further sections. In the general case each sample may produce more than one extract, and each extract, more than one labeled extract.

(e) quality related indicators
quality control steps taken:

(i) biological replicates?
ExperimentalDesign has association (ReplicateDescription) to Description and ArrayDesign allows features with replicate Reporters and CompositeSequences

(ii) technical replicates (replicate spots or hybs)?
if two Features have same BioMaterial (for spotted arrays) or they point to the same BioSequence (for printed arrays) they are technical replicates

(iii) polyA tails
ExperimentalDesign has association (QualityDescription) to Description, in BioSequence, hybridization Protocol could list whether poly A tails were blocked

(iv) low complexity regions
ExperimentalDesign has association (QualityDescription) to Description, Hybridization Protocol could list this

(v) unspecific binding
ExperimentalDesign has association (QualityDescription) to Description, Hybridization Protocol could list this

(vi) other
QualityDescription

(f) optional user defined "qualifier, value, source" list (see Introduction)
Everything derives from Extendable

(g) a free text description of the experiment set or a link to a publication
Experiment has Description, and has BibliographicReference through Description

B.5 Array design: each array used and each element (spot) on the array.

This section describes details of each array used in the experiment. There are two parts of this section: B.5.a describes the list of physical arrays themselves, each of these referring to specific array design types described in B.5.b. We expect that the array design type descriptions will be given by the array providers and manufactures, in which case the users will simply need to reference them.

(a) Array copy (physical instance)

(i) unique ID as used in part 1
Array has identifier attribute from Identifiable (which is unambiguous, not unique)

(ii) array design name (e.g., "Stanford Human 10K set")
ArrayDesign has name attribute from Identifiable

(b) Array design
The section consists of three parts a) description of the array as the whole, b) description of each type of elements (spot) used (properties that are typically common to many elements (e.g., 'synthesized oligo-nucleotides' or 'PCR products from cDNA clones'), and c) description of the specific properties of each element, such as the DNA sequence. In practice, the last part will be provided as a spread-sheet or tab-delimited file.

(i) array related information

1. array design name (e.g., "Stanford Human 10K set") as given in B.5.a
ArrayDesign is Identifiable, which has a name attribute

2. platform type: *in situ* synthesized, spotted or other
Generated through DesignElementGroup association Type

3. array provider (source)
ArrayManufacture has a Contact

4. surface type: glass, membrane, other
PhysicalArrayDesign has SurfaceType association

5. surface type name
   PhysicalArrayDesign has SurfaceType association

6. physical dimensions of array support (e.g. of slide)
   ArrayPackage has a Height and Width associations to Measurement

7. number of elements on the array
   DesignElementGroup has numberOfFeatures attribute

8. a reference system allowing to locate each element (spot) on the array (in the simplest case the number of columns and rows is sufficient)
   Each Feature has a Position

9. production date
   ArrayManufacture has attribute manufactureDate

10. production protocol (obligatory if custom produced)
    ArrayManufacture has a ProtocolApplication which has a Protocol

11. optional "qualifier, value, source" list (see Introduction)
    Everything is Extendable

(ii) properties of each type of elements (spots) on the array; elements may be simple, i.e., containing only identical molecules, or composite, i.e., containing different oligo-nucleotides obtained from the same reference molecule;

1. element type unique ID
   Feature is Identifiable

2. simple or composite
   Each DesignElementGroup is either a Feature a Reporter or a CompositeSequence

3. element type: synthetic oligo-nucleotides, PCR products, plasmids, colonies, other
   Each DesignElementGroup has association Types

4. single or double stranded
   Each DesignElementGroup has association Types

5. element (spot) dimensions
   Each FeatureGroup has attributes width, length, and height

6. element generation protocol that includes sufficient information to reproduce the element
   ArrayManufacture has ManufactureLIMS to associate Features and BioMaterials, and has a ProtocolApplication

7. attachment (covalent/ionic/other)
   Each DesignElementGroup has association Types

8. optional "qualifier, value, source" list (see Introduction)
   Everything is Extendable

(iii) specific properties of each element (spot) on the array:

1. element type ID from B.1.b.ii.1
   Each Feature is identifiable

2. position on the array allowing to identify the spot in the image (see B.8.a) below)
   Each Feature has a position

3. clone information, obligatory for elements obtained from clones:
   ArrayManufacture associates each feature with BioMaterial, BioSource has list of OntologyEntry's

   a. clone ID, clone provider, date, availability

4. sequence or PCR primer information:

   a. sequence accession number in DDBJ/EMBL/GenBank if known
      Reporter points to BioSequence

   b. sequence itself (if databases do not contain it)
      Reporter points to BioSequence

   c. primer pair information, if relevant
      ArrayManufacture has ManufactureLIMS which maps BioMaterial to Feature and BioMaterial has ProtocolApplication

5. for composite oligonucleotide elements:

   a. oligonucleotide sequences, if given
      Each Reporter has a BioSequence

   b. number of oligonucleotides and the reference sequence (or accession number), otherwise
      Each CompositeSequence records it's Reporters

6. one of the above should unambiguously identify the element
   Reporters have a BioSequence, which allow identification.

7. approximate lengths if exact sequence not known
   BioSequence has attribute length and attribute isApproximateLength

8. gene name and links to appropriate databases (e.g., SWISS-PROT, or organism specific databases), if known and relevant
   Reporters and CompositeSequences have BioSequences which have DatabaseReferences

9. Normally this information will be provided in one or more spread-sheets or tab-delimited files.

B.6 Samples: samples used, extract preparation and labeling

By a ‘sample’ we understand the biological material, from which the RNA gene products (or DNA) have been extracted for subsequent labeling, hybridisation and measuring. This section describes the source of the sample (e.g., organism, cell type or line), its treatment, as well as preparation of the extract and its labeling, i.e., all steps that precedes the contact with an array (i.e., hybridisation). This section is separate of each sample used in the experiment. In practice, if the treatments are similar, differing only slightly, the descriptions can be given together, clearly pointing out the differences.

(a) sample source and treatment (this section describes the biological treatment which happens before the extract preparation and labelling, i.e., biological sample in which we intend to measure the gene expression; for each sample only some of the qualifiers given below may be relevant):

(i) ID as used in section 1
All BioMaterials are Identifiable

(ii) organism (NCBI taxonomy)
BioSources have OntologyEntry's, one of which can be organism and NCBI taxonomy Ontology

(iii) additional "qualifier, value, source" list; each qualifier in the list is obligatory if applicable; the list includes:
BioSources have OntologyEntry's which can be these

1. cell source and type (if derived from primary sources (s))
2. sex
3. age
4. growth conditions
5. development stage
6. organism part (tissue)
7. animal/plant strain or line
8. genetic variation (e.g., gene knockout, transgenic variation)
9. individual
10. individual genetic characteristics (e.g., disease alleles, polymorphisms)
11. disease state or normal
12. target cell type
13. cell line and source (if applicable)
14. in vivo treatments (organism or individual treatments)
15. in vitro treatments (cell culture conditions)

16. treatment type (e.g., small molecule, heat shock, cold shock, food deprivation)

17. compound

18. is additional clinical information available (link)

19. separation technique (e.g., none, trimming, microdissection, FACS)

(iv) laboratory protocol for sample treatment
BioMaterial has a Treatment which has a ProtocolApplication

(b) hybridisation extract preparation

(i) ID as given in section 1
Hybridizations are Identifiable

(ii) laboratory protocol for extract preparation, including:

1. extraction method
BioMaterial has a Treatment which has a ProtocolApplication

2. whether total RNA, mRNA, or genomic DNA is extracted
BioMaterial has a Type ontology

3. amplification (RNA polymerases, PCR)
BioMaterial has a Treatment which has a ProtocolApplication

(iii) optional "qualifier, value, source" list (see Introduction)
Everything is Extendable

(c) labeling

(i) ID as given in section 1
LabeledExtract is Identifiable

(ii) laboratory protocol for labelling, including:

1. amount of nucleic acids labeled
BioMaterial has a Treatment which has a ProtocolApplication

2. label used (e.g., A-Cy3, G-Cy5, 33P, ….)
BioMaterial has an association to Compound

3. label incorporation method
BioMaterial has a Treatment which has a ProtocolApplication

(iii) optional "qualifier, value, source" list (see Introduction)
Everything is Extendable

B.7 Hybridisations: procedures and parameters

This section describes details of each hybridisation in the experiment. Each hybridisation has a separate section 4, though if they are similar they may be described together.

(a) ID as given in section 1
PhysicalBioAssays are Identifiable

(b) laboratory protocol for hybridisation, including:
PhysicalBioAssay has a Hybridization BioEvent which has a ProtocolApplication, there are additional BioAssayTreatments which can specify further treatments (wash, etc) if desired

(i) the solution (e.g., concentration of solutes)

(ii) blocking agent

(iii) wash procedure

(iv) quantity of labelled target used

(v) time, concentration, volume, temperature

(vi) description of the hybridisation instruments

(c) optional "qualifier, value, source" list (see Introduction)
Everything is Extendable

B.8 Measurements: images, quantitation, specifications:

This section describes the data obtained from each scan and their combinations

(a) hybridisation scan raw data:

(i) the scanner image file (e.g., TIFF, DAT) from the hybridised microarray scanning
PhysicalBioAssay has an association to Image which has a URI to the actual image file

(ii) scanning information:
ImageAcquisition has an association to ProtocolApplication and PhysicalBioAssay(hybridization) which has an association to Image,

1. input: hybridisation ID as in Section 1
2. image unique ID
3. scan parameters, including laser power, spatial resolution, pixel space, PMT voltage;
4. laboratory protocol for scanning, including:
    a. scanning hardware
    b. scanning software

(b) image analysis and quantitation

(i) the complete image analysis output (of the particular image analysis software) for each element (or composite element - see B.5.b.ii), for each channel – normally given as a spread-sheet or other external file
MeasuredBioAssay has a BioAssayData

(ii) image analysis information:
FeatureExtraction has a ProtocolApplication which has association to SoftwareApplication and PhysicalBioAssay and the MeasuredBioAssay which has a BioAssayData which has a QuantitationDimension which has the output Quantitations which are Identifiable

1. input: image ID
2. quantitation unique ID
3. image analysis software specification and version, availability, and the description or identification of the algorithm
4. all parameters

(c) summarized information from possible replicates

(i) derived measurement value summarizing related elements as used by the author (this may constitute replicates of the element on the same or different arrays or hybridisations, as well as different elements related to the same entity e.g., gene)
DerivedBioAssay contains the data and, through Transformation and Maps, allows identifying replicates.

(ii) reliability indicator for the value of c.i) as used by the author (e.g., standard deviation); may be "unknown"
DerivedBioAssays can have a Quantitation of type ConfidenceIndicator

(iii) specification how c.i and c.ii are calculated
Transformation has a ProtocolApplication

1. input: one or more quantitation ID’s
2. the specification should be based on values provided in b1

B.9 Normalisation controls, values, specifications

This section will be further detailed in the next MIAME version

(a) Normalisation strategy
Experiment has NormalizationDescription association

(i) spiking

(ii) “housekeeping” genes

(iii) total array

(iv) optional user defined “quality value”

(b) Normalisation algorithm
Experiment has NormalizationDescription association.
DerivedBioAssayData has Transformation.

(i) linear regression

(ii) log-linear regression

(iii) ratio statistics

(iv) log(ratio) mean/median centering

(v) nonlinear regression

(vi) optional user defined “quality value”

(c) Control array elements
DesignElementGroup and individual DesignElements have ControlType association

(i) position (the abstract coordinate on the array)

(ii) control type (spiking, normalization, negative, positive)

(iii) control qualifier (endogenous, exogenous)

(iv) optional user defined “quality value”

(d) Hybridisation extract preparation
Hybridization has a ProtocolApplication, and QuantitationTypes can have a related ExpectedValue which enables indicating spikes

(i) spike type

(ii) spike qualifier

(iii) target element

(iv) optional user defined “quality value”

Section 7 on quality control will be added to the next MIAME version.

This document represents overall consensus of MGED working group on microarray data annotations in all parts except section B.8.a, ‘hybridisation scan raw data’. A considerable majority of the working group supports the view that providing raw image data is an essential part of MIAME. However, there is also a notable minority that does not agree to this view. It is possible, that this requirement may be platform specific. We would like to encourage the microarray community to give us their views on the question, as well as on MIAME version 1.0 in general.

Glossary

Amino Acid
Any of a class of 20 small molecule building blocks that are combined to form proteins in living things (21 amino acids if selenocysteine is included). The sequence of amino acids in a protein and hence protein function are determined by the nucleotide sequence of its gene and the genetic code. The terms residue and amino acid are often used interchangeably.

Array
Array refers to the physical substrate to which biosequence reporters are attached to create features. In gene expression profiling experiments, arrays are hybridized with labeled sample and then scanned and analyzed to generate data. In MAGE, this term refers to the manufactured output of an ArrayDesign.

ArrayDesign
An ArrayDesign is the layout or blueprint of one or more **Arrays**. An ArrayDesign is conceptual and an **Array** is a physical object.

ArrayManufacture
An ArrayManufacture is a manufacturing event which produces one or more of arrays with a particular ArrayDesign or ArrayDesigns.

Background
Background is the measured signal outside of a feature on an array. In many gene expression analysis methods, background subtraction is performed to correct measured signals for observed local and/or global background. In MAGE, this term is one common example of a QuantitationType.

Base
see Nucleotide.

BioMaterial
In MAGE, BioMaterial refers to a biological material used in an experiment. It can represent a biological source material (see **BioSource**), an intermediate preparation derived from a **BioSource** (see **BioSample**), or a labeled extract (see **LabeledExtract)**.

BioSample
In MAGE, BioSample refers to an intermediate preparation derived from a biological source (typically a tissue, cell line, or whole organism) but not yet a **LabeledExtract.**

BioSequence
A **BioSequence** is an abstraction of a biological sequence, such as the ordered nucleotides of a DNA chain or the ordered amino acid residues of a protein molecule.

Biosequence Cluster
A biosequence cluster refers to a set of biosequences that are believed to be derived from the same gene. Biosequences that are derived from the same cDNA clone or that have been "clustered" together by resources such as UniGene are grouped together into biosequence clusters for analysis. Biosequence cluster should not be confused with cluster analysis.

BioSource
A biological source material used in an experiment. It usually represents a tissue or a cell line.

Cell Line
A cell line is a collection of cells propagated in culture. Cell lines are typically homogenous cell populations. In MAGE this is one example of an attribute for a BioSource.

Channel
A channel is an intensity-based portion of an expression dataset that consists of the set of signal measurements across all features on an array for a particular labeled preparation used in a hybridization. In some cases, such as Cy3/Cy5 array hybridizations, multiple channels (one for each label used) may be combined in a single expression profile to create ratios. In MAGE each image is associated with a single channel, and a BioAssay has one or more channels.

Chip
The physical medium of many arrays used in gene expression. This term is often used synonymously with Array.

Chromosome
One of the linear or sometimes circular nucleic acid containing bodies of viruses, prokaryotic organisms, and the cell nucleus of eukaryotic organisms that contain most or all of the genes of the individual.

Cluster Analysis
Cluster Analysis is a multivariate analysis technique that seeks to organize information about variables so that relatively homogeneous groups , or "clusters," can be formed. The clusters formed with this family of methods should be highly internally homogeneous in their behavior (members are similar to one another) and highly externally heterogeneous in their behavior (members are not like members of other clusters). In expression profiling, cluster analysis refers to the grouping of hybridizations, expression profiles, or combines across common sets of genes as well as grouping of features, reporters, biosequences, or biosequence clusters across sets of expression profiling experiments.

Compound
A Compound is a small molecule such as a drug. These small molecules can interact with cellular components to produce results that are therapeutic, toxic, or produce a desired experimental effect. Compounds are commonly used in Treatments of cellular samples in gene expression experiments.

Control
The reference for comparison when determining the effect of some procedure or treatment.

Deletion
Refers to a sequence that varies from its target sequence by the removal of one or more nucleotides.

Detrending
Detrending refers to the removal or reduction of systematic biases from a data set.

Developmental Stage
Physically describable state in the maturation of an organism. Common stages include blastocyst or larva. In MAGE, this term is one example of an attribute of BioSource.

Dosage Series
A type of experimental design in which a series of observations is made in which the variable is the amount or concentration of some compound, such as a drug.

Error Model
An error model is an algorithm that computes quality statistics such as P-values and error bars for each gene expression measurement. Error models are typically specific to a particular expression profiling technology.

Experiment
An experiment studies some observable system under controlled conditions while some variable is manipulated in hopes of understanding the effects of the variable on the system. In gene expression, one varies some set of parameters such as time, drug, developmental stage, or dosage on a sample. In addition, for microassay analyses the sample is processed and a preparation from the sample is often labeled with a detectable tag so that it can be used as the LabeledExtract for hybridization with an Array.

Expression
The conversion of the genetic instructions present in a DNA sequence into a unit of biological function in a living cell. Typically involves the process of transcription of a DNA sequence into an RNA sequence followed by translation of the RNA into protein. The RNA may be spliced before translation to remove introns.

Extract
A extract is preparation taken from a biological source (**BioSource**) such as a tissue or cultured cells. It could consist of purified protein, RNA, DNA, or other cellular material. For microarray gene expression experiments, mRNA or total RNA preparations are labeled and then hybridized to arrays.

Feature
A feature refers to a specific instance of a **Reporter** positioned upon an **Array**. Commonly referred to as a spot in a microarray experiment.

Feature Extraction (Image Analysis)
Quantitative analysis of an array image or scan to measure the expression values. In MAGE FeatureExtraction is a **BioEvent** that takes a single image as input and produces a **MeasuredBioAssay**, and the corresponding **BioAssayData** cube.

Fluor
Also, fluorophore. Short for fluorescent label. Such a tag is bound to mRNA or cDNA extracted from a sample. When properly excited the fluor gives off measurable fluorescence which is the observable in an experiment.

Gene
A gene refers to DNA which codes for a particular protein, or, in certain cases, a functional or structural RNA molecule. Genes may be inferred from the DNA sequence by way of a coding sequence.

Genome
The complete set of genetic information for a particular organism.

Hybridization (See Experiment)
A hybridization is the act of treating an **Array** with one or more **LabeledExtracts** under a specified set of conditions. In MAGE, hybridization is one example of a **BioAssayCreation** event.

Label
Label refers to a reagent, system, or mechanism of differentiating one preparation used in a given expression profiling experiment from others used in the same experiment. Cy3 and Cy5 fluorescent labels, for example, are commonly used to distinguish baseline & experimental preparations in gene expression microarray

hybridizations.

Mismatch (See control)
Refers to a sequence that varies from its target sequence by the replacement of one or more nucleotides.

Normalization
Mean signal intensity can vary dramatically among expression profiles or channels. Normalization is the procedure by which signal intensities from two or more expression profiles (or channels) are made directly comparable through application of an appropriate algorithm.

Nucleic Acid
A polymer of nucleotides. DNA and RNA are different classes of nucleic acids. May be double- or single-stranded.

Nucleotide
A subunit of DNA or RNA consisting of a nitrogenous base (adenine, guanine, thymine, or cytosine in DNA; adenine, guanine, uracil, or cytosine in RNA), a phosphate molecule, and a sugar molecule (deoxyribose in DNA and ribose in RNA).

Oligo/Oligonucleotide
Usually short strings of DNA or RNA to be used as reporters. These short stretches of sequence are often chemically synthesized.

Photolithography
One method of creating spots composed of oligonucleotides.

Probe
In some organizations, probe is used as a synonym for **Feature**. Probe is also occasionally used to refer to the **LabeledExtract** hybridized to an **Array** (sometimes know as target). Due to the confusion of these two terms, MAGE uses the terms **Feature** and **LabeledExtract** to be unambiguous.

Profile
A profile, short for expression profile, is a data set extracted from an expression experiment.

Protein
A biological molecule which consists of many amino acids chained together by peptide bonds. The sequence of amino acids in a protein is determined by the sequence of nucleotides in a DNA molecule. Proteins perform most of the enzymatic and structural roles within living cells.

Protocol
A protocol is a collection of descriptions and parameters which describe an event in such a way as to enable a researcher to reproduce the event. A protocol is conceptual and **ProtocolApplication** is an implementation of a protocol which defines the values for each of the protocols parameters.

ProtocolApplication
A ProtocolApplication is an implementation of a **Protocol** which defines the values for each of the protocols parameters.

Radiation type
Radiation type corresponds to any physical type of radiation. It could be ultra-violet, gamma, infrared, X-ray or others. The proposed model includes the possible use of radiation treatment as part of an experimental design.

QuantitationType
A QuantitationType is one element of the **QuantitationDimension** of a **BioAssayData** cube. This dimension describes the measurements associated with a given **DesignElement** (e.g. **Feature**, **Reporter**, or **CompositeSequence**) in that **BioAssay.**

Ratio
Also referred to as 'fold change'. A ratio refers to a normalized signal intensity generated from one **DesignElement** (e.g. **Feature**, **Reporter**, or **CompositeSequence**) in a given channel divided by a normalized signal intensity generated by the same **DesignElement** (e.g. **Feature**, **Reporter**, or **CompositeSequence**) in another channel. The channels compared are typically baseline versus experimental, e.g., normal vs. diseased or untreated vs. treated. In MAGE, this term is one example of a **QuantitationType**.

Repeated Measurements
Repeated measures are measurements that are made more than once on the same data. For instance fluorescence from spots on arrays may be non-uniform over the spot. Making repeated measurements may permit a statistical estimate of confidence in the values obtained by looking at their variation.

Replicate Experiments
The ability to reproduce an experiment is key to the validity of science. A replicate set refers to repeated experiments where the same type of array is used, and the same probe isolation method is used. The intent is to arrive at separately obtained results which can be compared with one or more other sets of observations to arrive at a consensus or more statistically meaningful interpretation of results.

Reporter
Description of the material that is placed on a feature to represent a biological sequence. A reporter, in some technologies, can vary from the biosequence it represents. These variations are usually a mismatch or a deletion in the nucleotide sequence. Multiple reporters for a given biosequence are possible, and the same reporter can be used at multiple features to generate replicate measurements.

SAGE
Serial Analysis of Gene Expression. The use of short 10-14bp sequence tags linked together to identify sequences in a sample and the number of instances of each sequence in a sample. This is non-array-based gene expression analysis, and is not covered by the current design of MAGE.

Sequence
The ordered set of nucleotides in a DNA or RNA molecule, or the ordered set of amino acids in a protein. Synonym for Biosequence.

Signal
Signal refers to an experimental measurement of expression level.

Solvent
A substance for dissolving or dispersing one or more other substances, typically a **Compound**(s), used in a treatment.

Species
Common designation, such as 'A. Thaliana' or 'E. coli', from which the sample was obtained, based on NCBI's Taxonomy.

Spot
See **Feature**.

Target
See **Probe**, **LabeledExtract**, and **Feature**.

Time Series
A type of experimental design in which the variable element is the time (duration) of treatment by some protocol.

Tissue
A tissue is a primary cellular sample isolated from a biological source organism that contains an enriched subset of one or more cell types from that organism, e.g. cardiac muscle or skeletal muscle. In MAGE it is one attribute of a **BioSource**.

Treatment
A treatment is the experimental manipulation of a sample such as a cell culture, tissue, or organism prior to extraction of a preparation. In MAGE, treatments are events which take **BioSamples** as input and create a new **BioSample**.

Virtual Array
The resulting **BioAssayData** of a **BioAssayCreation** and series of **BioAssayTreatments** may abstract away the actual lower level **DesignElements** so that the user sees the results only on the **CompositeSequence** or the **Reporter** level. The **VirtualArray** allows description and annotation of these **DesignElements** for reference in the **BioAssayData**.