MAGE-ML: MicroArray Gene Expression Markup Language

Links:

- Full MAGE specification: http://cgi.omg.org/cgi-bin/doc?lifesci/01-10-01
- MAGE-ML Document Type Definition (DTD): http://cgi.omg.org/cgi-bin/doc?lifesci/01-11-02

Microarray Gene Expression Markup Language (MAGE-ML) is a language designed to describe and communicate information about microarray based experiments. MAGE-ML is based on XML and can describe microarray designs, microarray manufacturing information, microarray experiment setup and execution information, gene expression data and data analysis results. This paper describes the basic MAGE-ML structure and its main concepts. The structure of MAGE-ML is not simple, therefore we envisage that typically MAGE-ML documents will not be created "by hand" (i.e., just using text editors), but by user-friendly tools providing web form interface and the ability to link external tab-delimited data, that are currently being developed. Therefore the purpose of this paper is twofold:

1. For microarray laboratory scientists it may be describing all they need to know about MAGE-ML;
2. For bioinformaticians that intend to develop tools for microarray data management or to run such tools for microarray laboratories, this paper can serve as the first introduction to MAGE-ML.

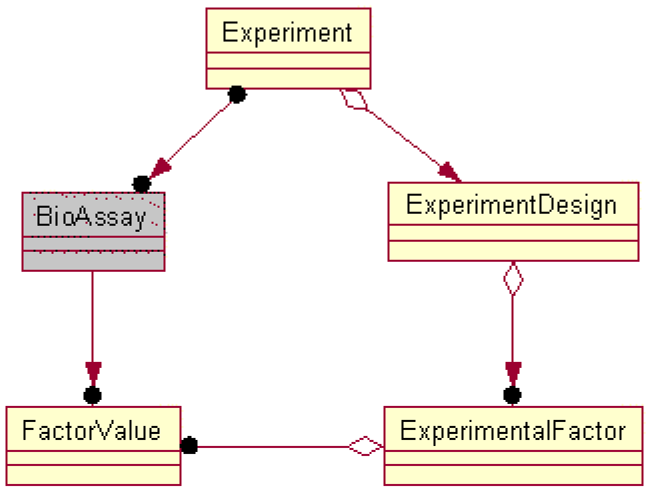
MAGE-ML has been automatically derived from Microarray Gene Expression Object Model (MAGE-OM), which is developed and described using the Unified Modelling Language (UML) – a standard language for describing object models. Descriptions using UML have an advantage over direct XML document type definitions (DTDs), in many respects. First they use graphical representation depicting the relationships between different entities in a way which is much easier to follow than DTDs. Second, the UML diagrams are primarily meant for humans, while DTDs are meant for computers. Therefore MAGE-OM should be considered as the primary model, and we will explain MAGE-ML by providing simplified fragments of MAGE-OM, rather then XML DTD or XML Schema. For full details of MAGE-OM, mapping rules used to derive MAGE-ML from MAGE-OM, as well as comprehensive glossary of terms see http://cgi.omg.org/cgi-bin/doc?lifesci/01-10-01

MAGE-OM is expressed in UML, however, for the purposes of this preliminary explanation knowledge of UML is not necessary. The main principle is that boxes represent *classes* of some objects and lines represent various *relationships* between the classes. Classes can represent various sets of entities: real-world physical objects (e.g., microarrays), real-world abstractions (e.g., microarray experiments), real-world actions (e.g., hybridization events) or information objects (e.g., microarray data matrices). Here we will not analyse what kinds of relationships there are in UML and how they are represented on diagrams, for details, see, e.g., http://www.ajug.org/info/tech/uml/uml.html

MAGE-OM is a bit two large to be represented on a single diagram in a readable way. In order to structure the model the UML notion of *packages* is used. Related classes are grouped together into packages, and quite often represented on the same diagrams. MAGE-OM will be explained package-by-package; a tool able to export MAGE-ML will probably have separate modules and/or user interface sections for separate packages, e.g., you can enter information about array designs in one UI section and information about steps of your microarray experiment using another UI section. On diagrams classes belonging to the package under discussion are coloured yellow, while classes belonging to other packages, therefore detailed elsewhere, but drawn on the current diagram for the purposes of showing inter-package relationships, are coloured grey.

Experiment

This package is for describing a microarray experiment as a unit. Note two parallel branches on the diagram. On the right-hand side we have experiment blueprint information – experiment design and one or more experimental factors that are changed in the course of the experiment to explore whether and how gene expression levels change (e.g., time or drug concentration). On the left-hand side there is experiment execution information. An experiment consists of one or more bioassays (experiment steps), and each bioassay can test for gene expression with one or more experimental factor values fixed (e.g., time = 30min, drug concentration = 15ug/ml).

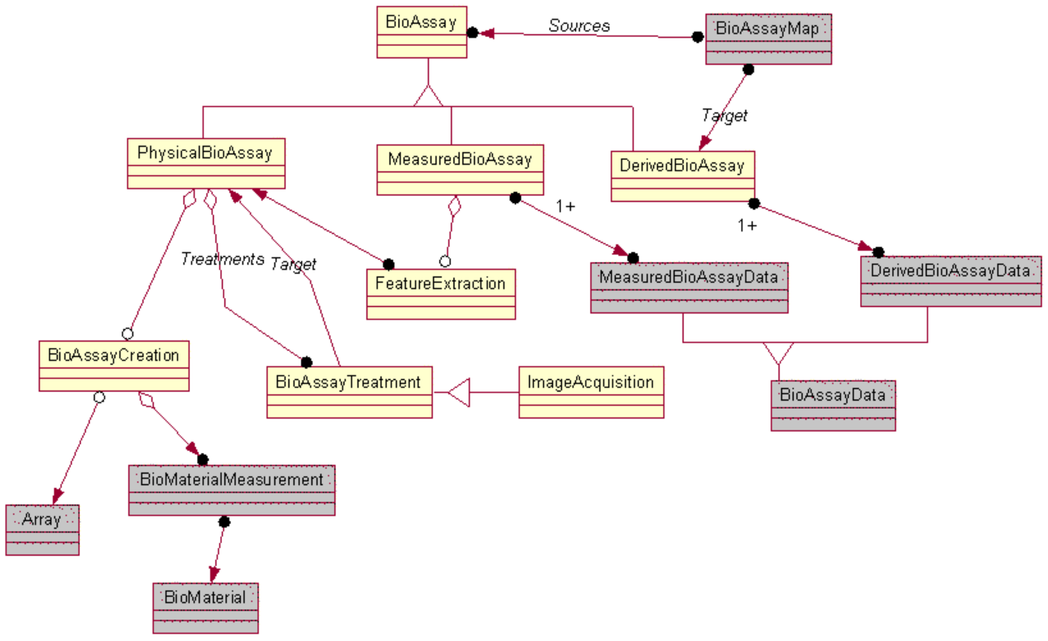


BioAssay

A bioassay is a single step within a microarray experiment. There are 3 types of bioassays. A physical bioassay correspond to wet-lab microarray experimental step. A measured bioassays corresponds to a situation after feature extraction has been performed. A derived bioassay corresponds to data processing experimental steps.

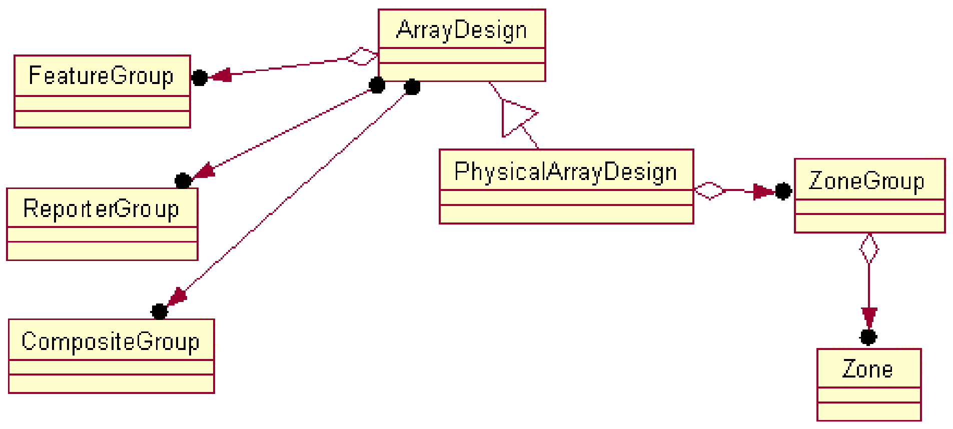
A physical bioassay is created by applying some amount of some biomaterial to a microarray. Bioassay treatment events (e.g., wash, apply blocking agent etc.) transform physical bioassays into new physical bioassays. A particular type of bioassay treatment is image acquisition.

Measured bioassays can have corresponding MeasuredBioAssayData objects (raw data). Derived bioassays are obtained by data transformations, they are linked by BioAssayMap objects.



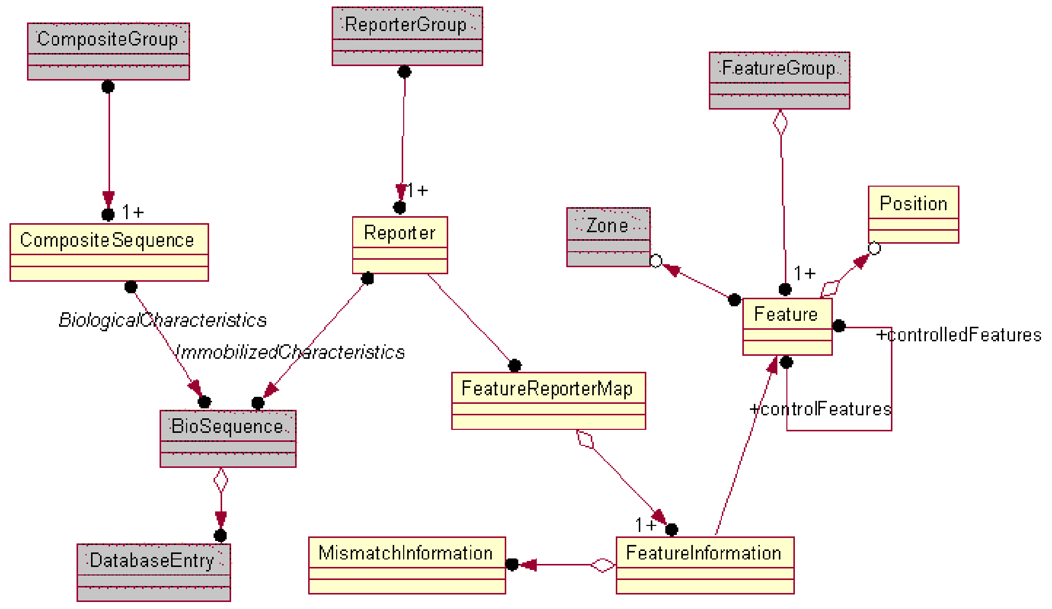
ArrayDesign

An array design consists of design element groups as well as information about element zone layout. Physical array design has been made as a subclass of array design, to allow “virtual” array designs with element groups but no zone layout information; such “virtual” designs can be used, e.g., to define different reporter-composite sequence mappings for the same physical array design. Zones can be grouped together to form zone groups; zones within the same zone group would have the same spacing between them, whereas zones from different zone groups can have different spacing. Zones/zone groups sometimes are referred to as blocks/metablocks.



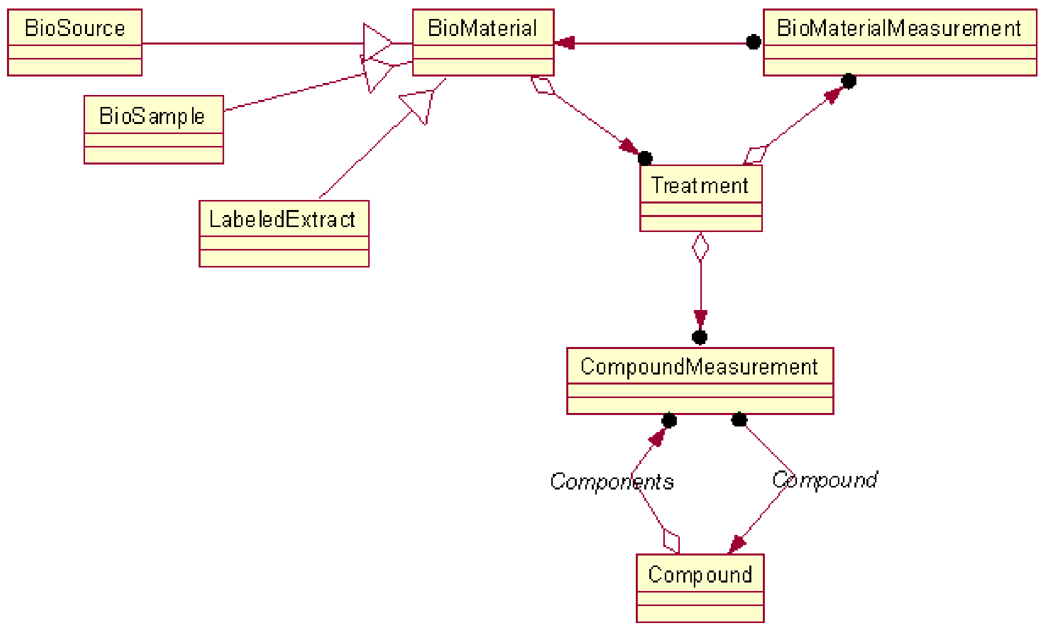
DesignElement

There are three types of design elements. A feature has a position on the array (within some zone), it can be a control feature for other features or controlled by other features. A reporter corresponds to the physical substance synthesized/printed on the array, it can be characterized by one or more biosequence objects which in turn can be characterized by database entries. There can be many features for the same reporter, and features can have one or several mismatches compared to reporter's reference sequence. The third, most abstract kind of design element is a composite sequence. It can have more than one reporter on the same array (e.g., different splice variants) and is characterized by biological characteristics, which are actually again sequences with corresponding database entries. The mapping between reporters and composite sequences is not shown on this diagram, but this is similar to the model of feature-reporter mapping. Also, composite sequences can be aggregated into more abstract composite sequences (also not shown here), e.g., genes of the same functional group etc.



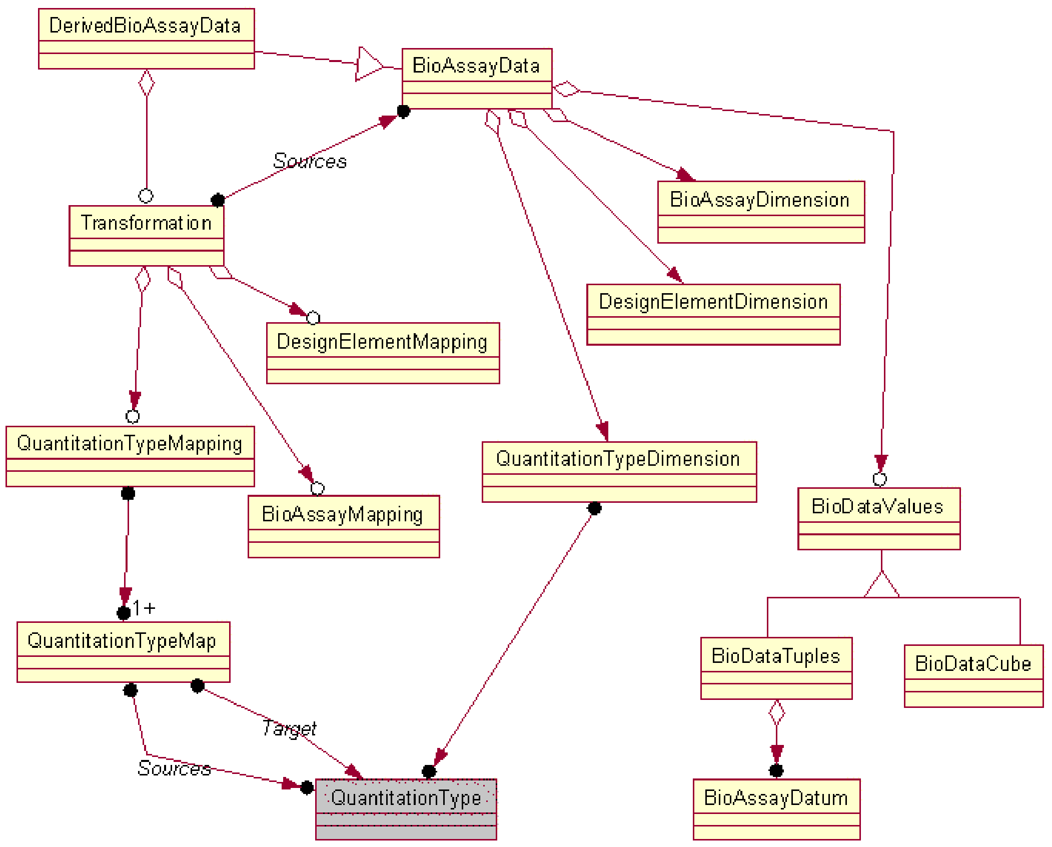
BioMaterial

BioMaterial is an abstraction of various states of biology-based materials used in various stages of the microarray experiment. Biosource refers to the initial source of material used in hybridization (e.g., cell line or tissue). Biosample is what is extracted from the biosource, and labeled extract is the last state of the biomaterial before hybridization. A biomaterial can be a result of a chain of treatments, each treatment involving one or more biomaterials in some amounts. A special kind of treatment is treatment with some amount of a compound. A simple way to model compounds consisting of other compounds is provided.



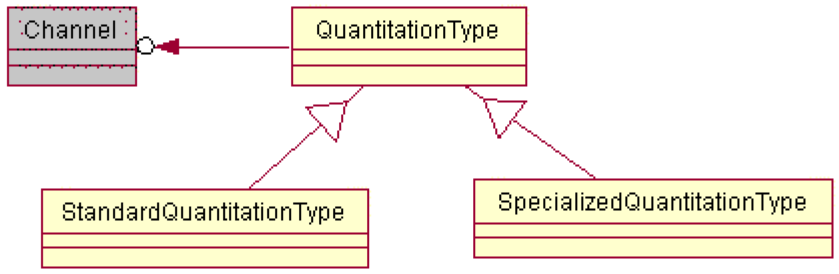
BioAssayData

One of the central principles of MAGE is that data objects are regarded as 3-dimensional matrices, where there are bioassays (experimental steps or conditions) along one dimension, design elements (spots) along the other dimension and quantitation types (e.g., signal intensity, background intensity) along the 3rd dimension. Bioassay data objects can be represented in one of two ways: as a set of vectors in the form (value, dimension1, dimension2, dimension3) (useful for small amounts of data), or as a 3-D matrix (Bio DataCube). Transformations (e.g., filtering, normalization) can be applied to one or more bioassay data objects, resulting in derived data objects. A transformation involves computing values of the resulting 3-D matrix from the values of source matrices, and it also transforms dimensions. On this diagram just the mapping of quantitation types into new quantitation types has been shown; DesignElementMapping and BioAssayMapping are modeled similarly. A quantitation type mapping transforms a list of quantitation types into another list of quantitation types, and it consists of maps that deal with single target quantitation types.



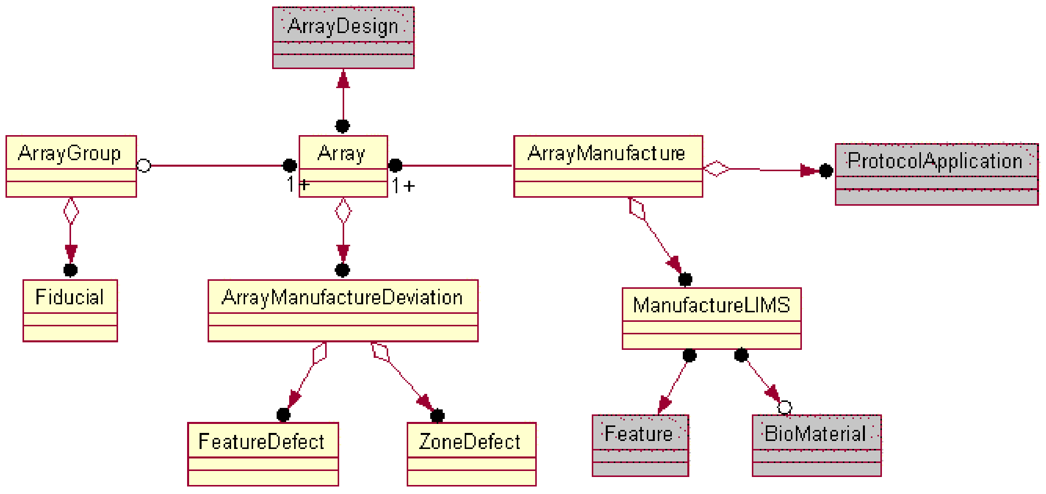
QuantitationType

A quantitation type can be either a standard quantitation type (a list of these is provided within MAGE) or a specialized quantitation type which should be described in detail. A quantitation type may reference a channel (e.g., Cy3 green signal intensity).



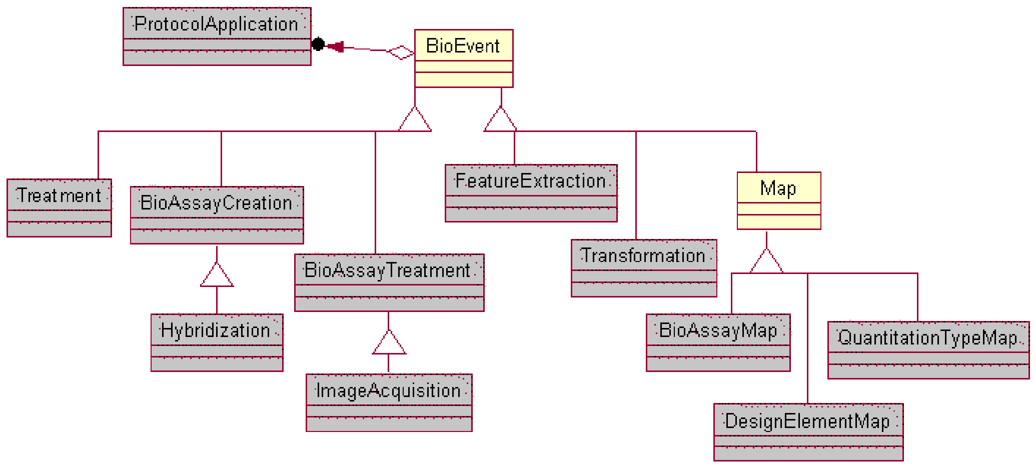
Array

An array is a physical array which corresponds to some array design. There are three types of information that can be captured about individual arrays and array production process. An individual array can have deviations from the design, either zone defects (e.g., a whole zone of spots is shifted) or individual feature defects. An array group can consist of more than one array printed on the same slide, and fiducials (markings on the surface of the slide that can be used to identify arrays' origins) can be printed to facilitate feature detection software accuracy. Array manufacture information also can refer to more than one individual array (sometimes referred to as an array batch), and it can contain protocol information (how the arrays were manufactured) as well as some limited LIMS information (what was printed on the array, on a feature-by-feature basis).



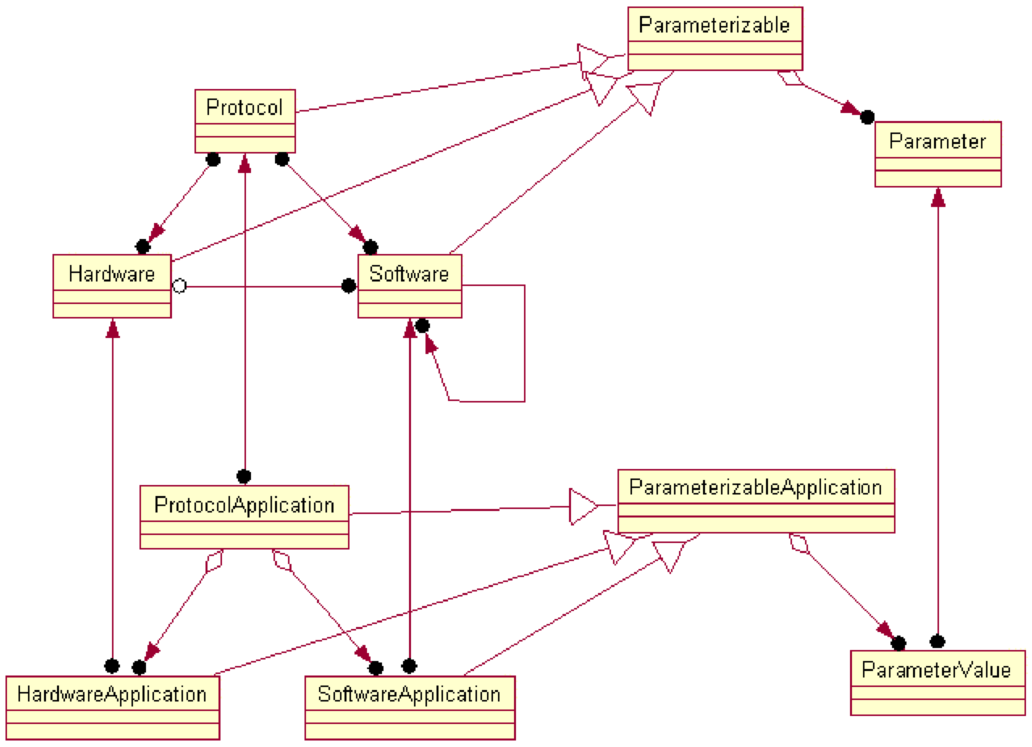
BioEvent

This diagram is provided to summarize what kinds of events are possible to describe in MAGE. Each event can have a sequence of protocol applications. On the left-hand side there are pysical events (biomaterial treatment, bioassay creation as a generalization for hybridization, and bioassay treatment and image acquisition as a special case), while on the right-hand side there are information processing events (feature extraction, data transformation, maps of data dimensions).



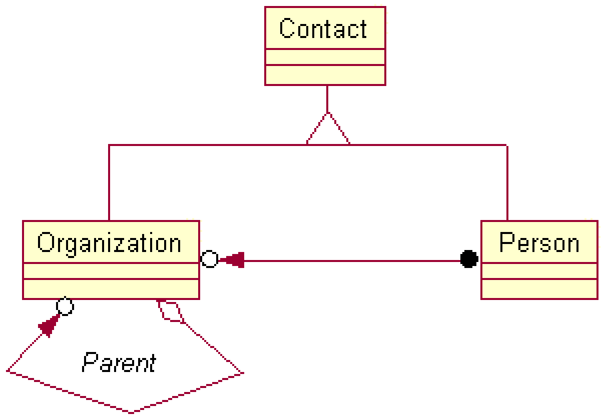
Protocol

There are two parts for this package. On the upper part there are Protocol, Hardware and Software classes, representing abstract entities. All the “parameterizable” objects can have parameters, a protocol can involve usage of specific hardware and software, specific hardware might be needed to run some software, and software objects can be composed of other software objects (modules). On the lower part there are classes representing application of abstract entities at a given time point, with parameters filled in by some parameter values.



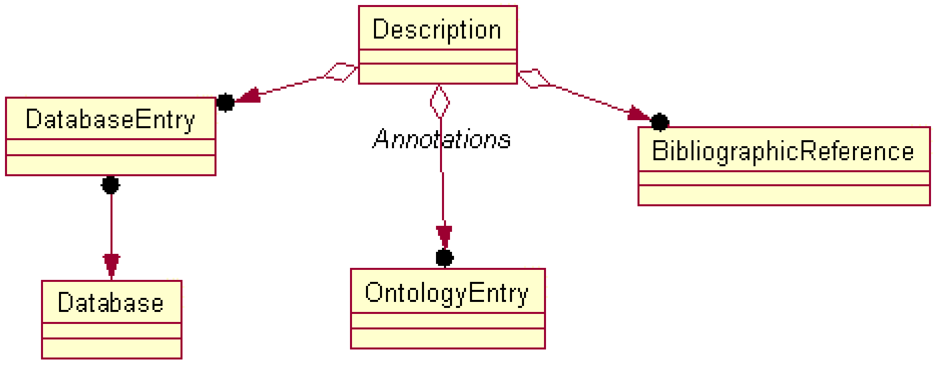
AuditAndSecurity

A contact can be either an organization or a person. A person can work for an organization, and an organization can consist of other sub-organizations.



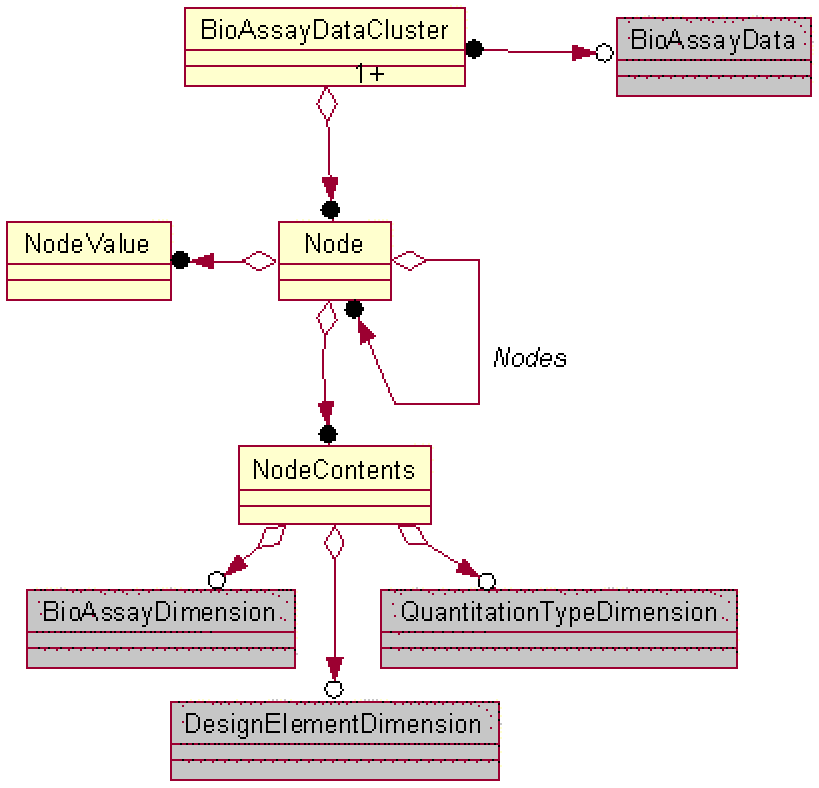
Description

Many MAGE objects can be further chracterized by attaching Descriptions to them, if there is some important information that cannot be recorded using provided attributes and relations. A Description can consist of a piece of free text, references to database entries, references to ontology entries (annotations) and one or more bibliographic references. A generic NameValueType class also is provided, objects of which can be attached to every MAGE object if even the Description functionality is not sufficient.



HigherLevelAnalysis

Experimental data (BioAssayData) can be clustered, obtaining one or more top level clusters. Each cluster consists of nodes, where each node in turn can contain subnodes (in the case of hierarchical clustering). A node can be characterized by its values (e.g., some metric of cluster quality), and a node groups together design elements (e.g., spots, genes) or bioassays (i.e., experimental conditions). BioAssayDimension is just an ordered list of bioassays, and DesignElementDimension is an ordered list of design elements.


BioAssayDataCluster
1+
BioAssayData
NodeValue
Node
Nodes
NodeContents
BioAssayDimension
QuantitationTypeDimension
DesignElementDimension