



caArray Cancer Array Informatics

Software Design Overview

NCI Center for Bioinformatics

April 14, 2004

Agenda

- Introductions / project goals
- Current project status
- caBIG compliance
- Architecture & design overview
- Proof of concept
- Q&A

caArray Project Goals

- ❑ Allow researchers to manage and document their experiment data with full MIAME 1.1 compliant annotations
- ❑ Support import and export of MAGE-ML, allows data exchange via an internationally standardized format.
- ❑ Support submission and retrieval of Affymetrix and GenePix native format files
- ❑ Consistent with caBIG principles, provides open-source application to NCI-designated cancer centers and other affiliated organizations

Key Features (Phase I)

- ❑ MIAME 1.1 compliant
- ❑ MAGE-ML import and export
- ❑ Submission and retrieval of native Affymetrix and GenePix files
- ❑ Use of MGED Ontology and controlled vocabularies
- ❑ N-tier architecture
- ❑ User-access management via the NCICB common security model

Key Features (Phase II)

- XpressionWay, a pathway visualization tool
- UCSF SPOT data
- SAGE data
- User interface for User Management

Current Status

❑ Completed Analysis

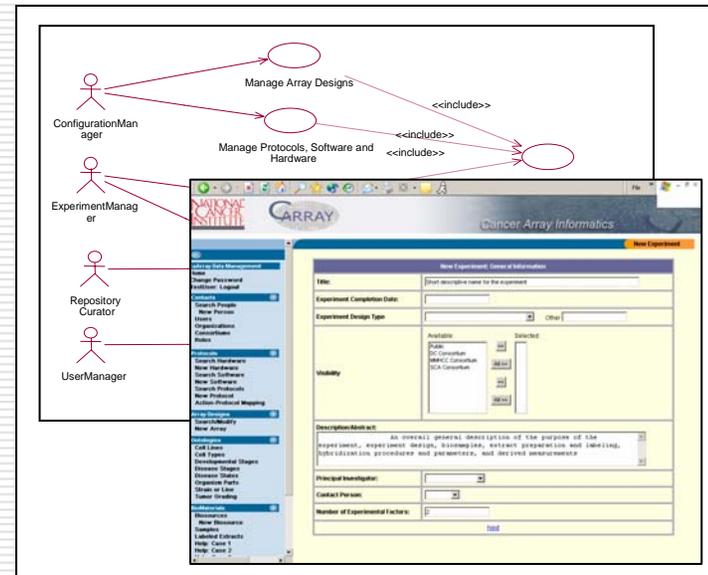
- Use case diagrams
- Wire frames

❑ Completed Design

- Sequence diagrams
- Class diagrams
- Reference implementation /

proof of concept of protocol management

❑ Completed MAGE-stk persistence



caBIG Compliance

General Principles

- Open access, open source
 - Both raw and processed data will be made available. Patient privacy and other concerns will be respected and addressed, but will not in and of themselves preclude sharing of appropriate data.
 - Software source code, use cases, designs, models, test plans, and other artifacts of the development process will be shared.
 - Where possible, open source enabling technologies will be selected
 - Good will among participants
 - caBIG Workspaces are collaborative teams, and participants are expected act accordingly.
 - Debate/discussion is encouraged, but professional demeanor and tone should be the norm.
- caArray allows submission and retrieval of all public data. The protection group/element concept in caArray allows researchers to preclude sharing of sensitive data.
 - All caArray use cases, design documents, test cases, source code and all other artifacts are being developed under an open source license. See caBIO license.
 - caArray utilizes only open source technologies such as MAGE-stk, JBoss, Xerces, Struts, Ant, OJB, etc
 - caArray is being built in an collaborative environment to address researchers' needs

caBIG Compliance

Architectural Principles

- Data and analytic services will be made available to caBIG using uniform application programming interfaces (APIs) and appropriate standard message formats.
 - The APIs and messages will be derived from common information models for biomedical data entities or objects, and will be accessible through interfaces that reflect these models.
 - APIs and messages will support the delivery of data and also of accompanying metadata, in order to ensure that aggregated data sets are comparable.
- caArray will allow programmatic interface to its data via the EJB Managers, MAGE-OM API, and the MAGE-ML document.
 - caArray is built upon MAGE-OM object model, MIAME and MGED-Ontology standards.
 - APIs and messages will support the delivery of data and also of accompanying metadata, in order to ensure that aggregated data sets are comparable.
 - caArray supports and extends MAGE-OM which allows for the deep annotation of microarray experiments according to MIAME.

caBIG Compliance

Architectural Principles, cont.

- Applications will be engineered to use the common caBIG APIs. → caArray is built to utilize and complement the caCORE infrastructure.
- Standards for data exchange formats will be used. → caArray is built upon the MIAME, MAGE-ML and MGED-Ontology standards.
- All systems, applications and selected standards will be documented. → caArray will include JavaDocs, a User Guide, and online Help.

caBIG Compliance Data Standards

- Data will be described by metadata elements that conform to an accepted standard such as ISO/IEC 11179. Metadata will be leveraged to achieve data interoperability and comparability.
 - caArray metadata will reside in caDSR, a ISO/IEC 11179 derived repository.
 - caArray is built to support MAGE OM (an OMG specification), as describe in Uniform Modeling Language (UML).

- Data and metadata will be constructed and drawn from appropriate vocabulary and ontology standards.
 - caArray utilizes the MGED Ontology, a set of open and standard controlled vocabularies and ontologies built to support annotation of microarray data.

- caBIG will consume of appropriate public, open access standards where they are available.
 - caArray is built upon MAGE-OM object model, MIAME 1.1, and MGED-Ontology standards. (See next slide)

Compliance with International Standardization Efforts

- MIAME
 - Minimum Information About a Microarray Experiment
 - 1.1 Draft 6 (April 1, 2002)
 - http://www.mged.org/Workgroups/MIAME/miame_1.1.html
- MAGE-ML
 - MicroArray and GeneExpression Object Model and Markup Language
 - 1.1 (October 2003)
 - <http://www.omg.org/docs/formal/03-10-01.pdf>
- MGED Ontology
 - Microarray Gene Expression Data Ontology
 - 1.1.8 (April 2004)
 - <http://mged.sourceforge.net/ontologies/MGEDontology.php>

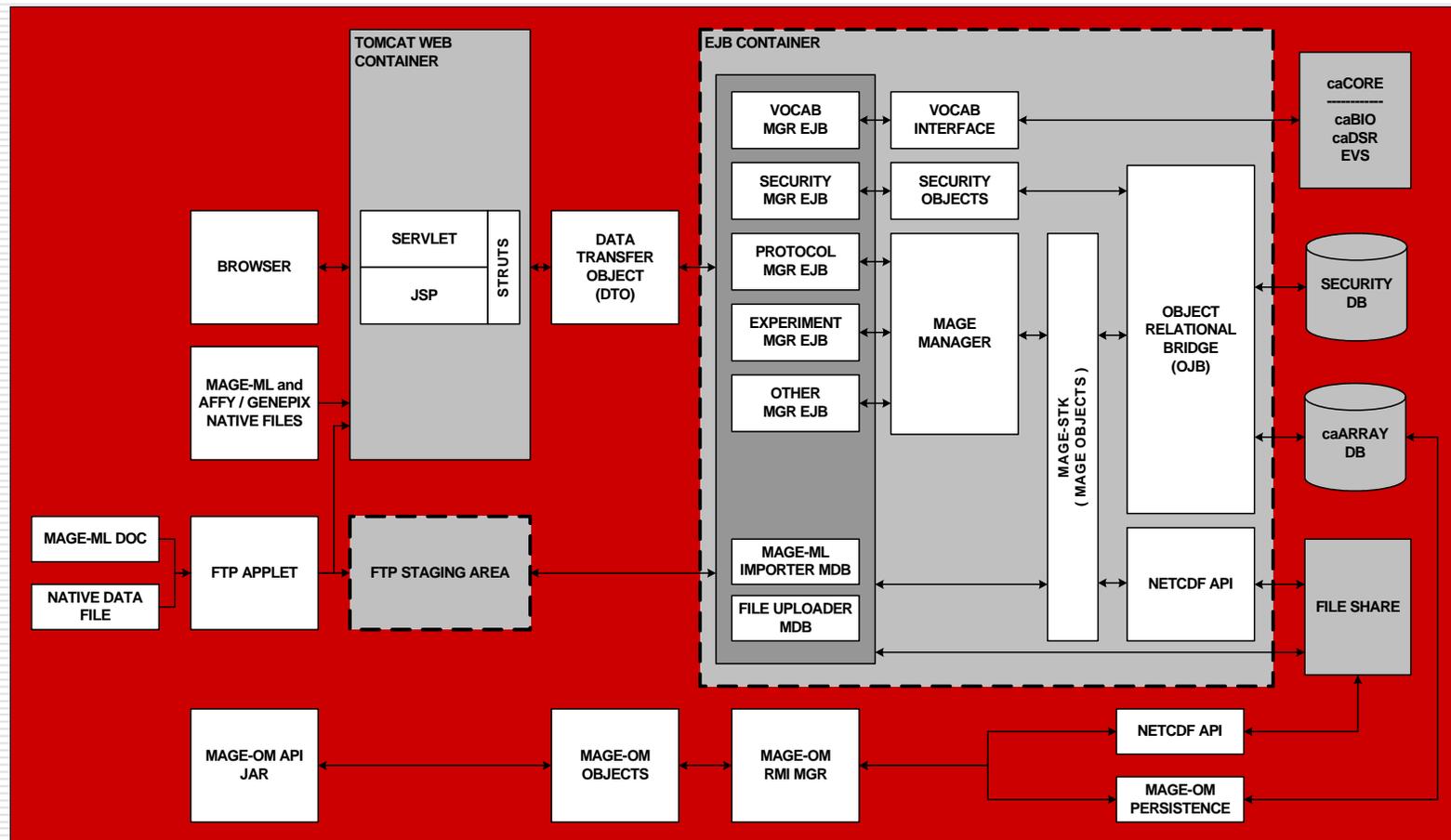
Issues Impacting caArray Architecture

- MAGE-ML Export/Import
 - Utilizes MAGE-stk 1.1 (evolving)
- User Access
 - Utilizes the Common NCICB Security Schema
 - Common NCICB authentication and authorization security module is in progress
- Scalable Robust Distributed Architecture
 - caArray is utilizing a both a J2EE web container and J2EE EJB container (JBoss implementation) for submission
 - Java Messaging Service (JMS) for asynchronous processing of large uploaded documents.
 - MAGE-OM API (RMI based) for retrieval
- Database Independence
 - caArray uses Apache's ObjectRelationalBridge (OBJ) as the Object Relational Mapping tool that allows transparent persistence for plain old Java Objects (POJO's) in the MAGE-stk toolkit against relational databases.
 - OBJ is also the current standard for caBIO allowing code reuse across projects.

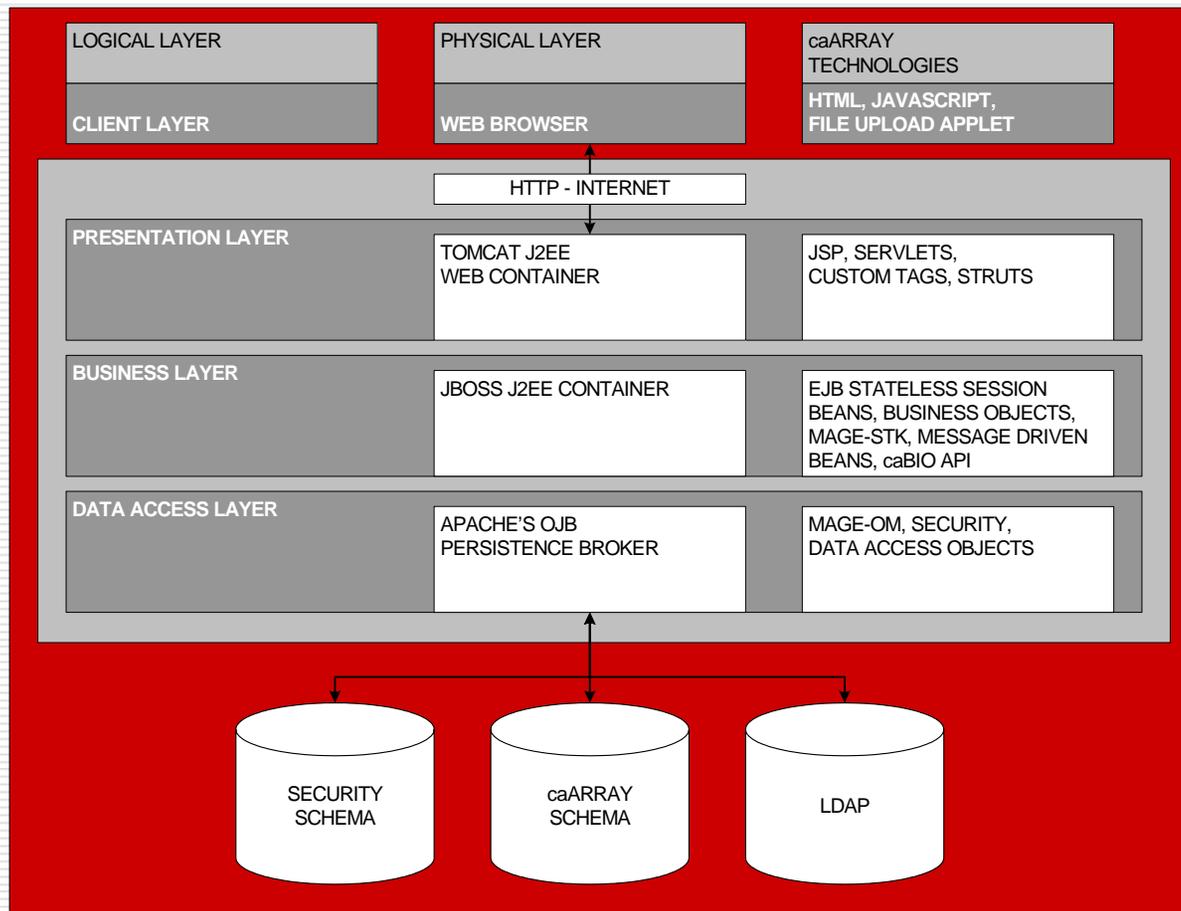
Data Access

- MAGE-OM API
- EJB API via data transfer objects (DTO)
- MAGE-ML export
- Native Affymetrix and GenePix file formats

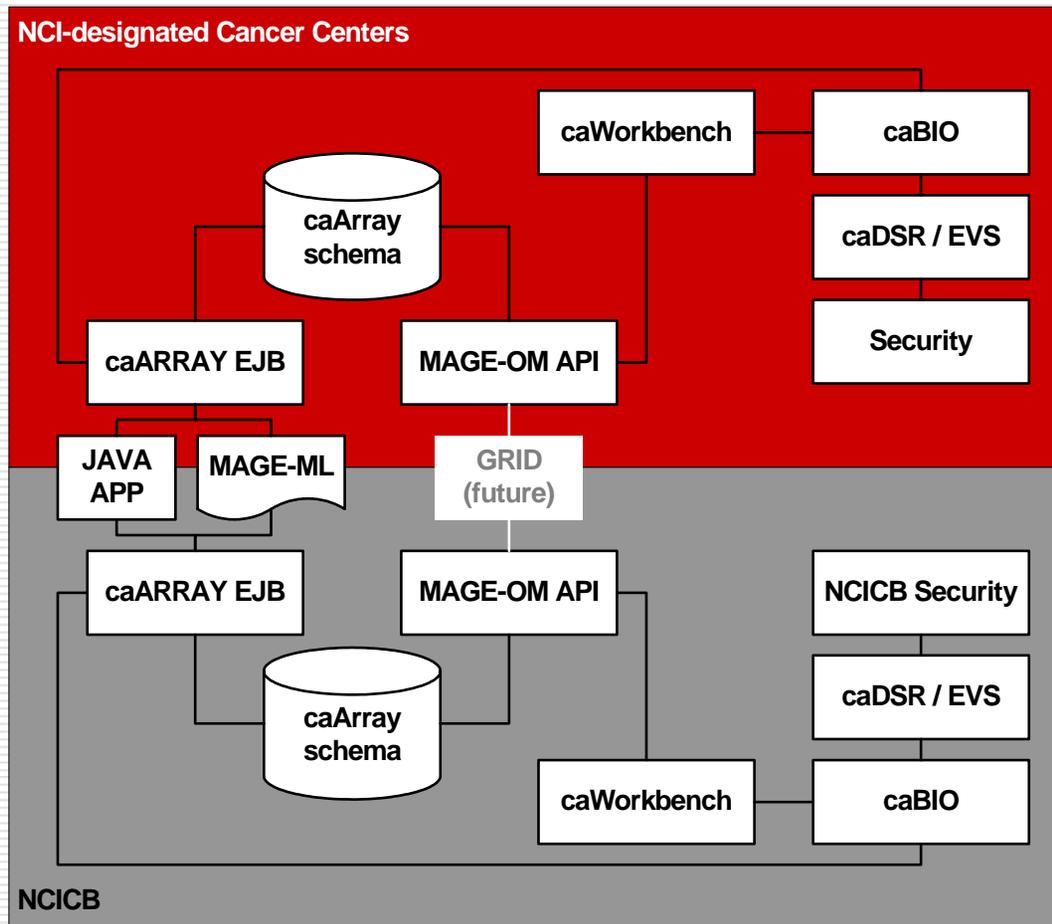
caArray Architecture



caArray Technologies



caArray at Cancer Centers



User Interface

- ❑ Conforms to NCICB UI Standards

The image displays two screenshots of the cancer.gov caARRAY website interface. The left screenshot shows a 'SPECIFY PERSON ATTRIBUTES' form with fields for First Name, Middle Initial, Last Name, Address, City, State, Postal Code, Toll Free Phone, Email Address, Fax Number, URL, and Affiliation. The right screenshot shows the 'CANCER ARRAY INFORMATICS' page with a magnifying glass over a microarray image, a 'WELCOME TO THE caARRAY' section, a 'LOGIN TO caARRAY' section with 'LOGIN ID' and 'PASSWORD' fields, and 'WHAT'S NEW' and 'DID YOU KNOW?' sections.

Design Challenges and Solutions

- Presentation
 - GUI framework
 - Model View Controller 2
 - Flexible web page layout
 - Composite View

Design Challenges and Solutions

- Business
 - Data encapsulation
 - Transfer Objects
 - Common object to locate/lookup services
 - Service Locator
 - Abstract and decouple business services
 - Business Delegate
 - Encapsulate the DTO to MAGE-OM logic
 - Transfer Object Assembler
 - Uniform coarse-grained service access layer to clients
 - Session Facade
 - Access to vocab/metadata services
 - Configurable Interface Pattern
 - Process asynchronous MAGE-ML import & file uploads
 - Service Activator

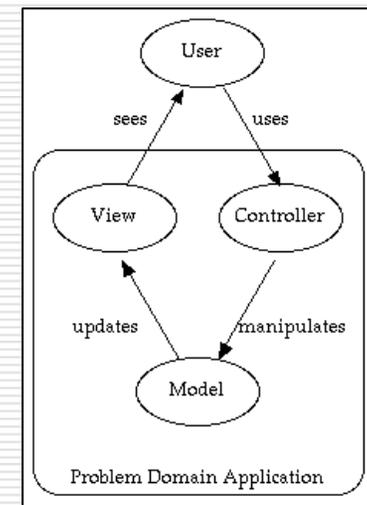
Design Challenges and Solutions

□ Persistence

- Efficient representation of large data sets
 - NetCDF Data Capture Strategy
- Decouple object and data source layer
 - Abstraction and Database Independence
- Efficient materialization of objects
 - Lazy Materialization Pattern

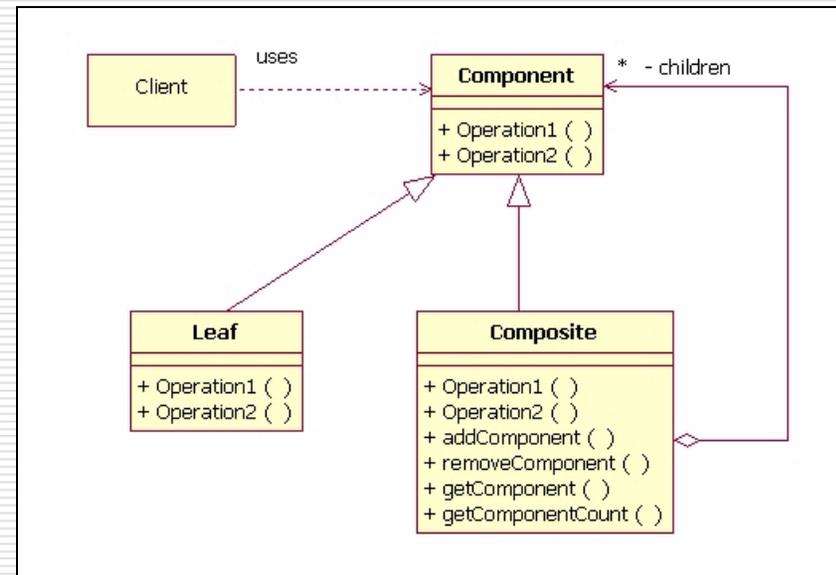
Model View Controller 2

- caArray's Presentation Layer utilizes the Model View Controller 2 (MVC2) design pattern
 - Separation of the application object (model) from the way it is represented to the user (view) from the way in which the user controls it (controller).
 - Implemented via Apache Struts Framework
<http://jakarta.apache.org/struts/>
- Benefit: Separates the UI from the underlying structure



Composite View

- ❑ The Composite design pattern lets you treat primitive and composite objects exactly the same.
- ❑ The Apache Struts framework includes a JSP tag library, known as Tiles, that lets you compose a Webpage from multiple JSPs.
- ❑ Benefit: Tiles allows us to build a flexible and reusable presentation layer.

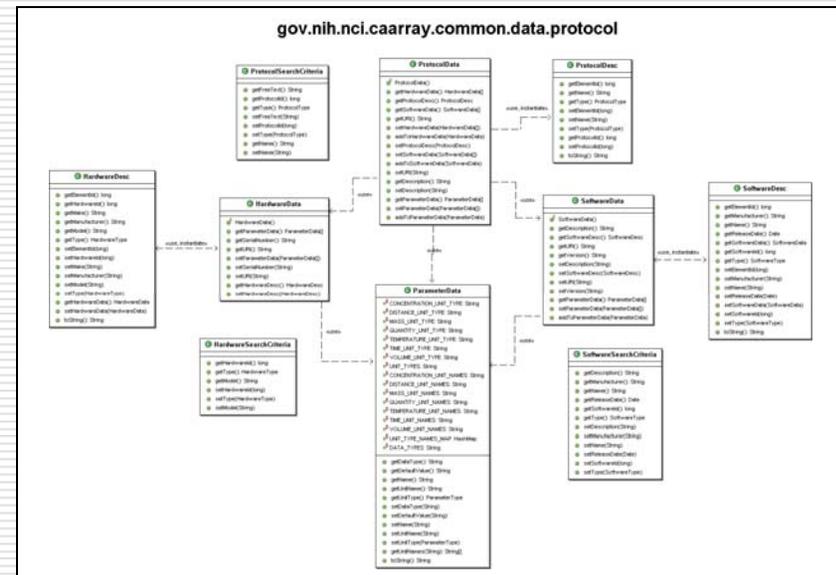


Transfer Objects Pattern

- ❑ Allows application client to exchange data with EJBs
- ❑ Client sets all user input in transfer object, which is sent to the EJB
- ❑ EJB processes business logic, sets all resulting data in the transfer object, and sends back to client

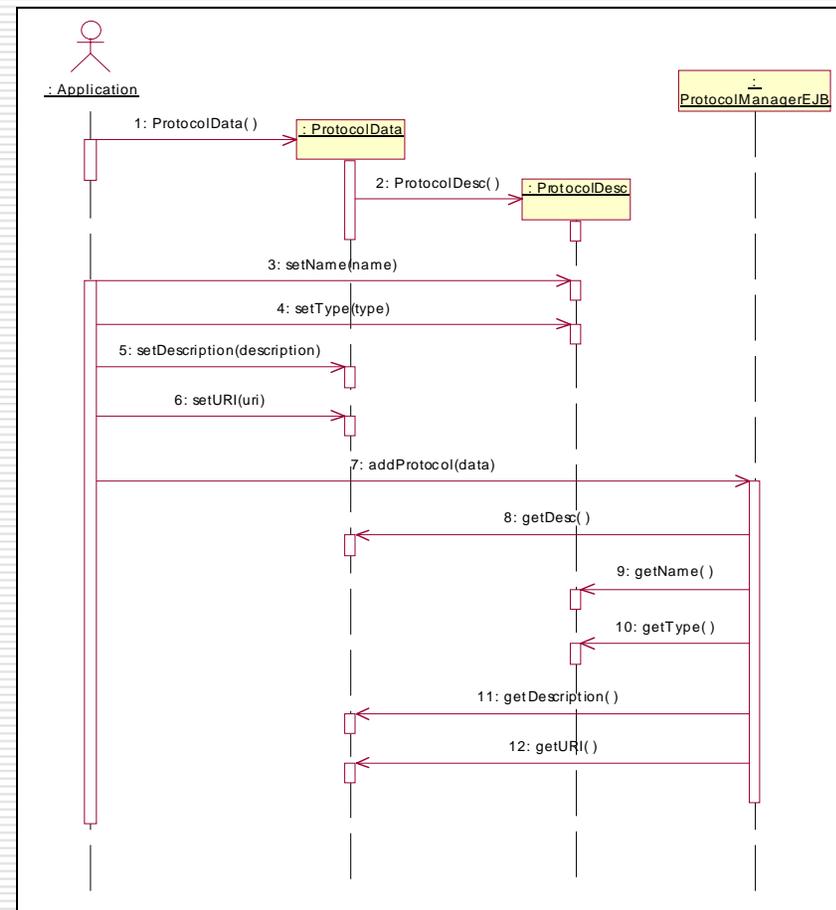
❑ Benefits

- All data needed by client and server-side processes are encapsulated in one object and sent/retrieved in one method call, lessening network impact
- Strongly-typed data transfer objects simplify server-side interface, providing easier code maintenance



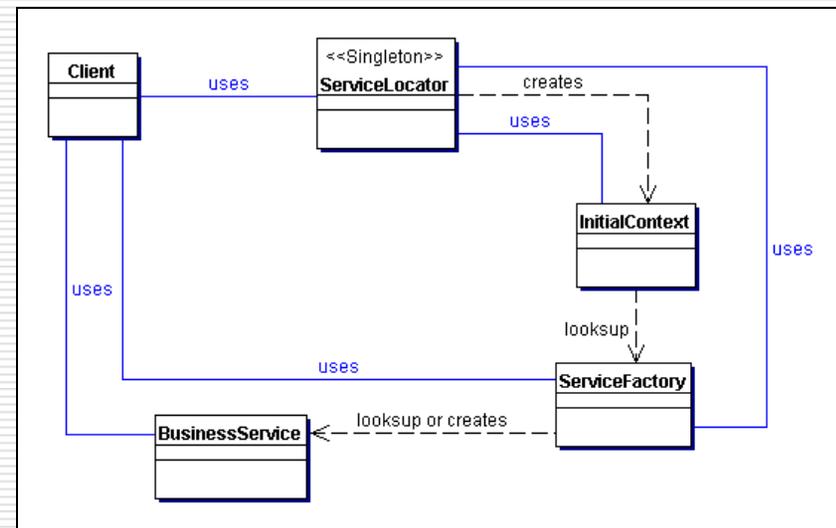
Transfer Objects Pattern

- Demonstrates application client setting user-entered values in the ProtocolData Transfer Object
- Client application then invokes EJB method to add protocol, sending the Transfer Object by value
- EJB method then retrieves all user-entered values from the Transfer Object, and begins business processing

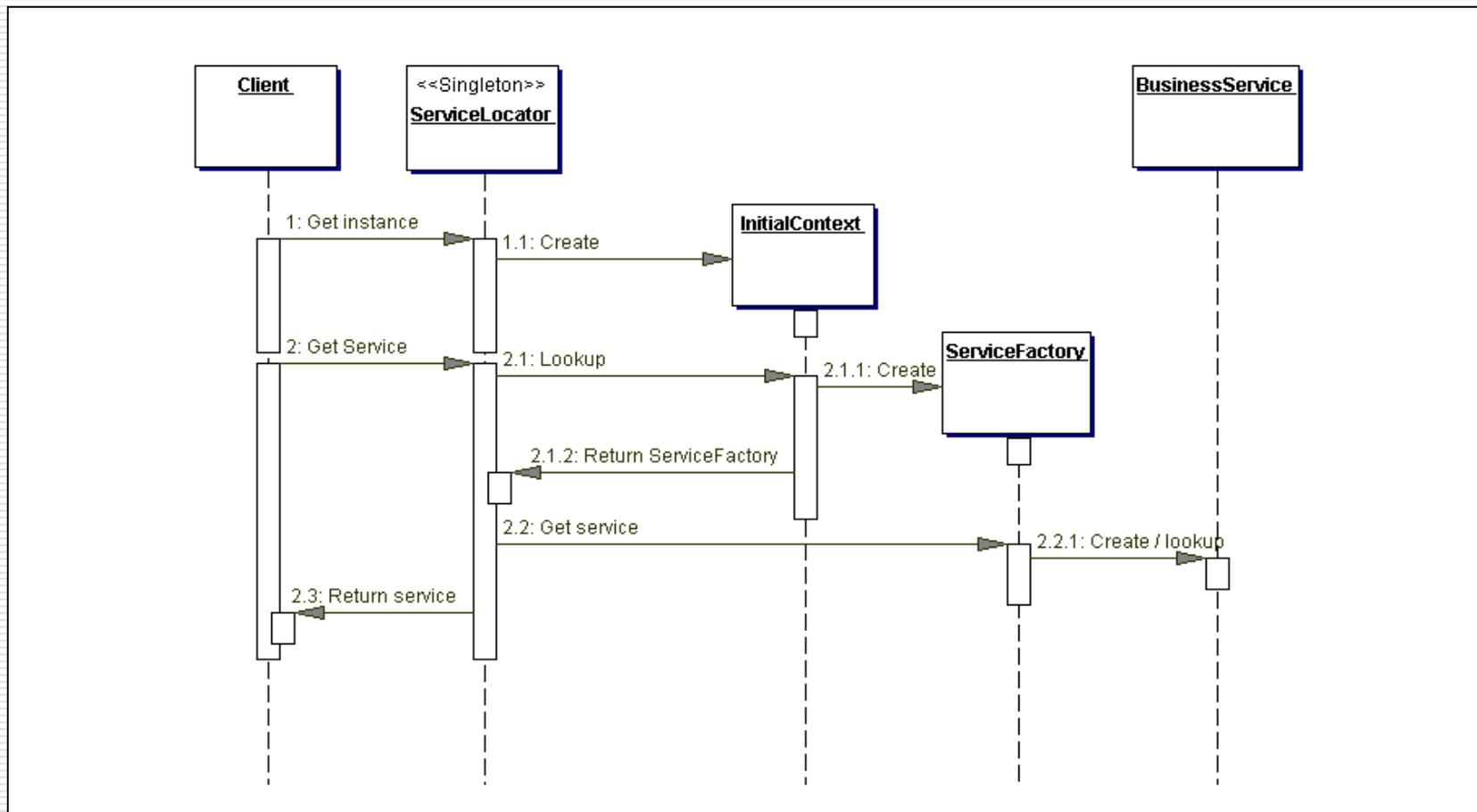


Service Locator Pattern

- ❑ caArray uses a common object to locate/lookup EJB home objects and JMS service components (Connection Factory, Session, Topic, etc.)
- ❑ Benefits
 - A single point of control of the complexity of the lookup operation
 - Ease of code maintenance for the creation of vendor-dependent initial context
 - Increased application performance with cached EJB home objects

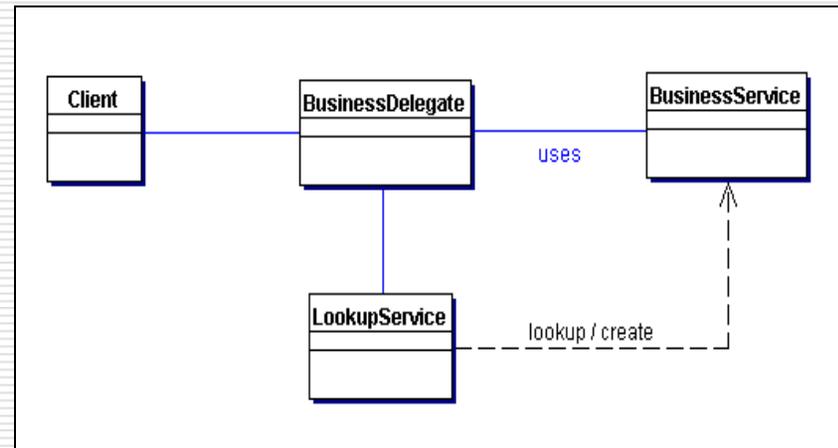


Service Locator Pattern

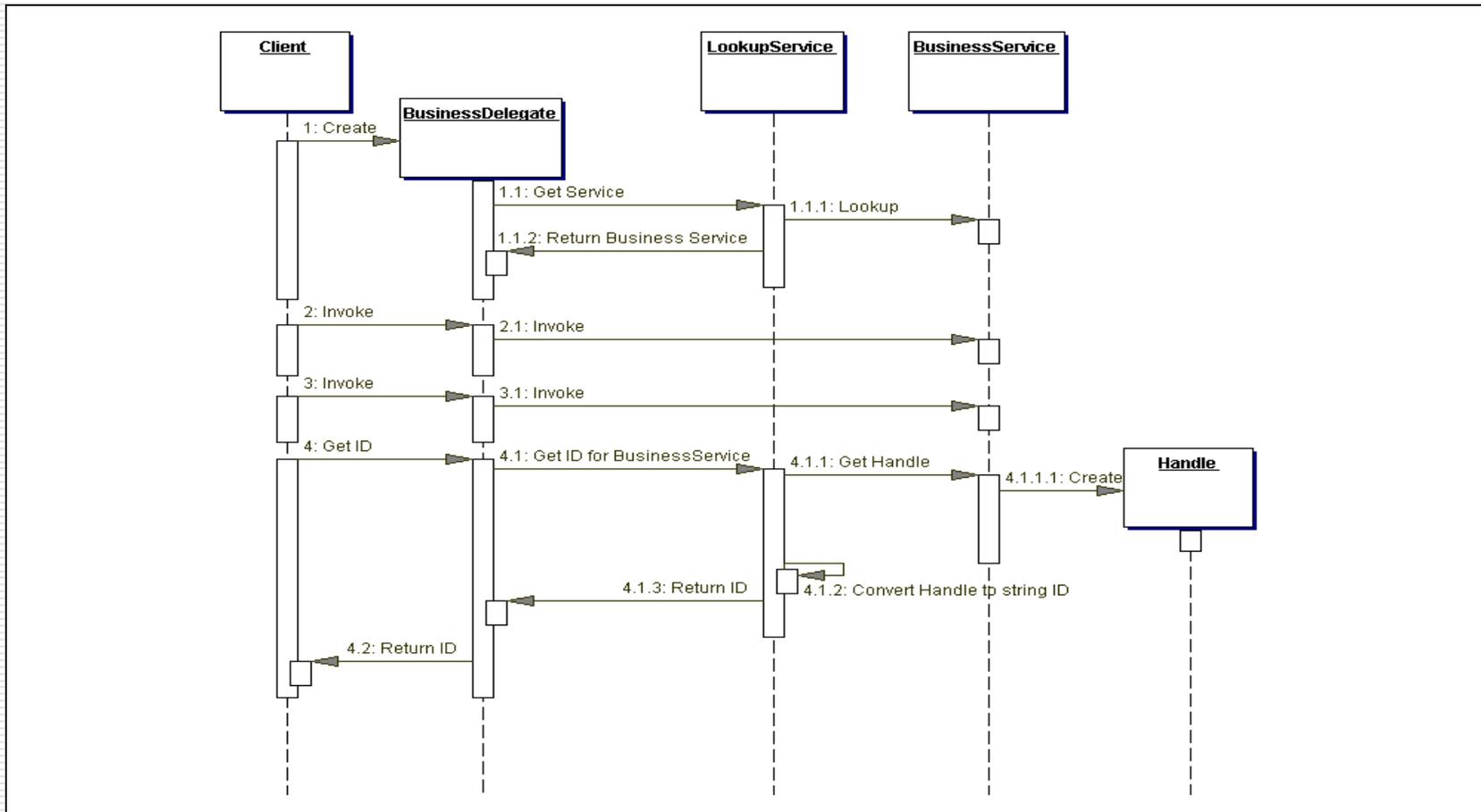


Business Delegate Pattern

- ❑ caArray uses a wrapper class for each of the specific EJB Managers such as ExperimentManager, ProtocolManager, etc. as Business Delegates to reduce coupling between presentation-tier clients and business services.
- ❑ Benefit
 - Hides the underlying implementation details of the business service, such as creation of EJB objects and access details of business operations.



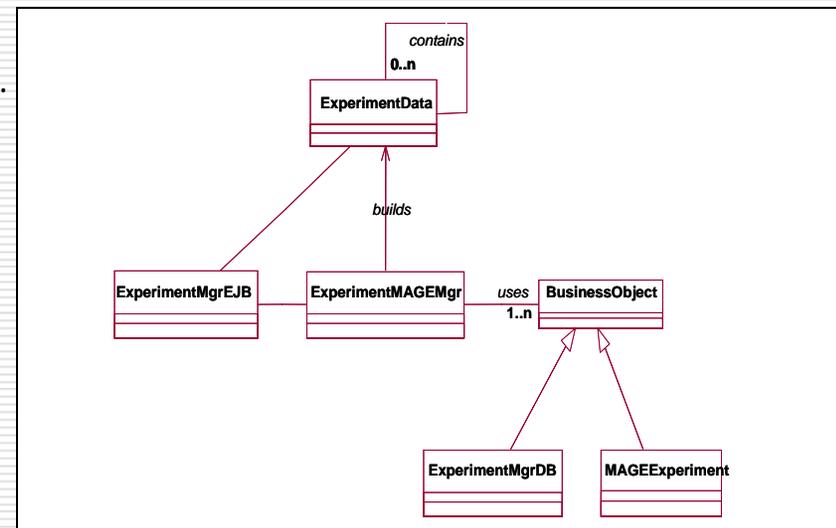
Business Delegate Pattern



MAGEManager

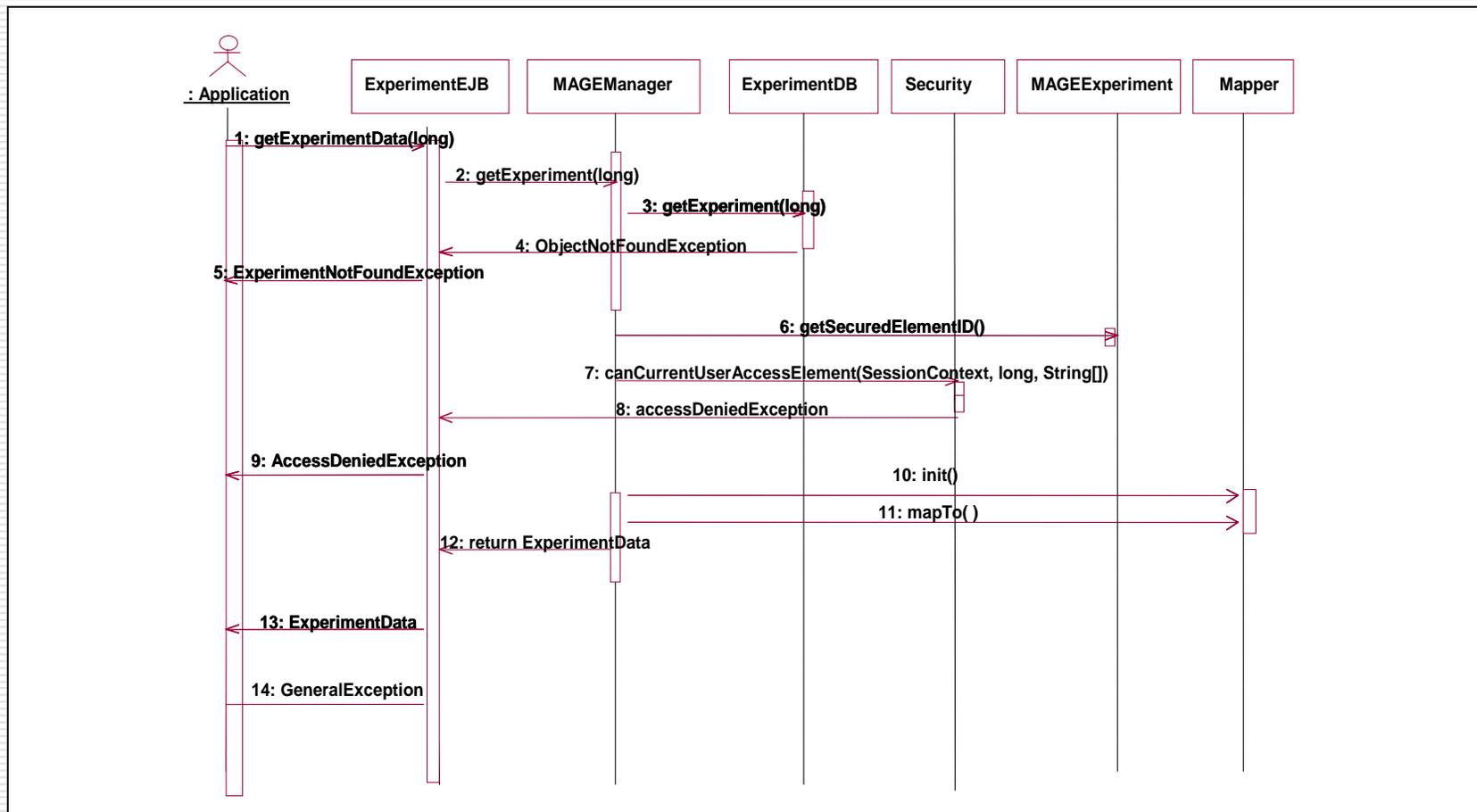
Transfer Object Assembler Pattern

- ❑ caArray uses the MageManager as a Transfer Object Assembler to build the required Mage model or submodel.
- ❑ The MageManager acts as the Transfer Object Assembler. It uses Transfer Objects to retrieve data from various MAGE-stk business objects that define the model or part of the model.
- ❑ The mapping between the DTOs and MAGE-stk is generated using xdoclet and persists in an XML document.
- ❑ Benefits
 - Encapsulates the MAGE-OM and hides from Business Logic.
 - Shields Business logic from complexity of assembling Transfer objects.
 - Use of Generic method reduces the code-base and enhances maintenance.



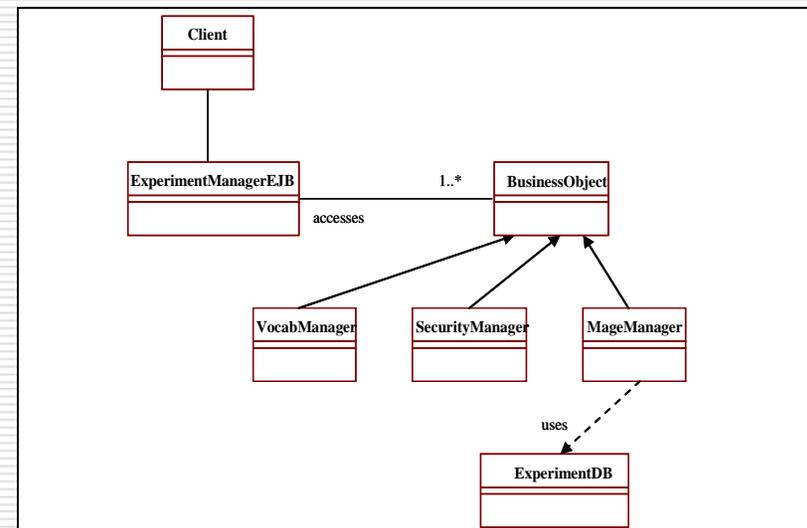
MAGEManager

Transfer Object Assembler Pattern

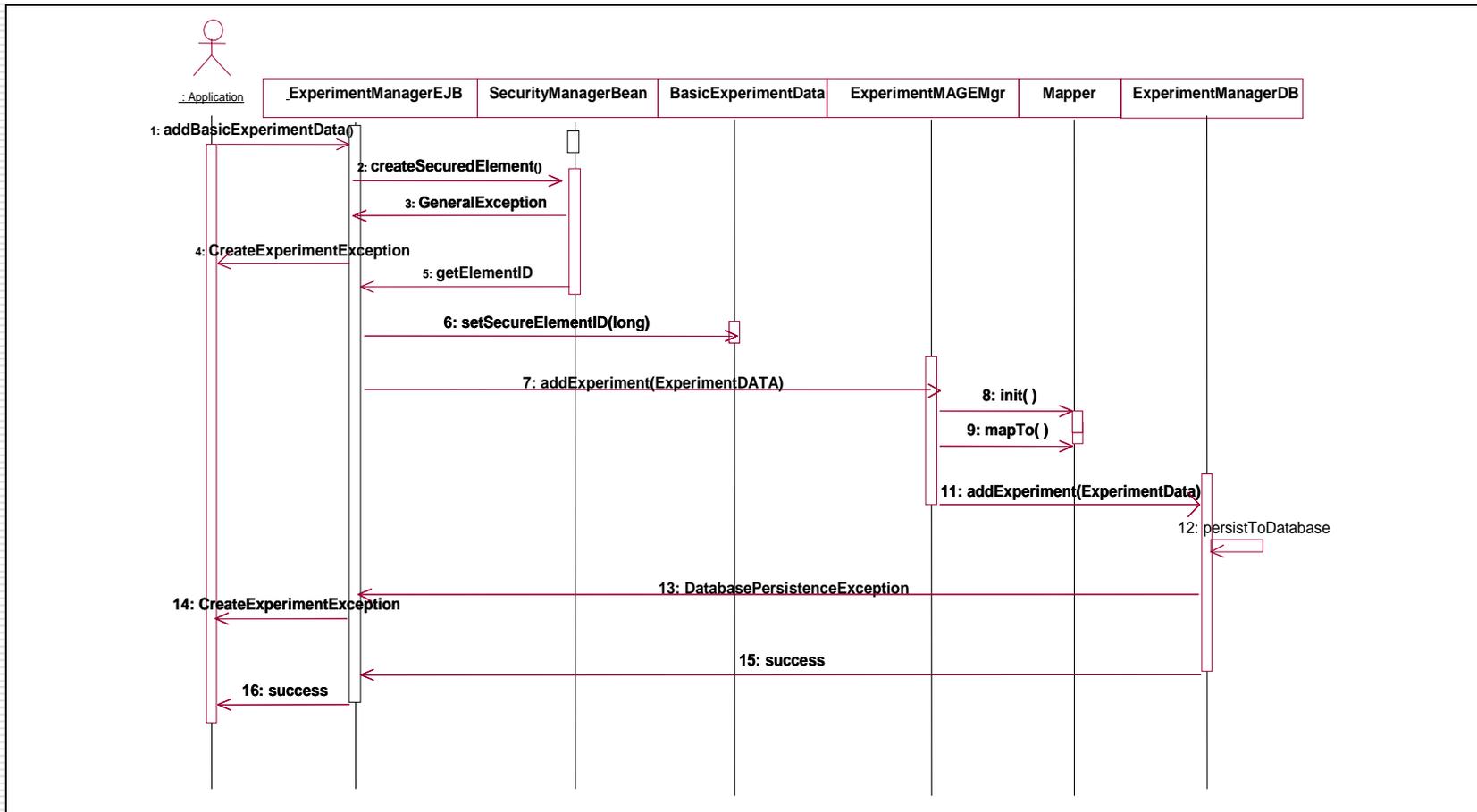


Session Façade Pattern

- ❑ caArray uses EJB Managers such ProtocolManager, ExperimentManager etc as a façade to hide the complexity of interactions between the business objects participating in a workflow.
- ❑ The Session Façade manages the business objects, and provides a uniform coarse-grained service access layer to clients.
- ❑ Benefits
 - Reduces coupling between client and server side code increasing manageability.
 - Provides common API interface access for multiple client applications.
 - Provides clean coarse grained access interface.
 - Allows for distributed deployment of client and server distributions.



Session Façade Pattern



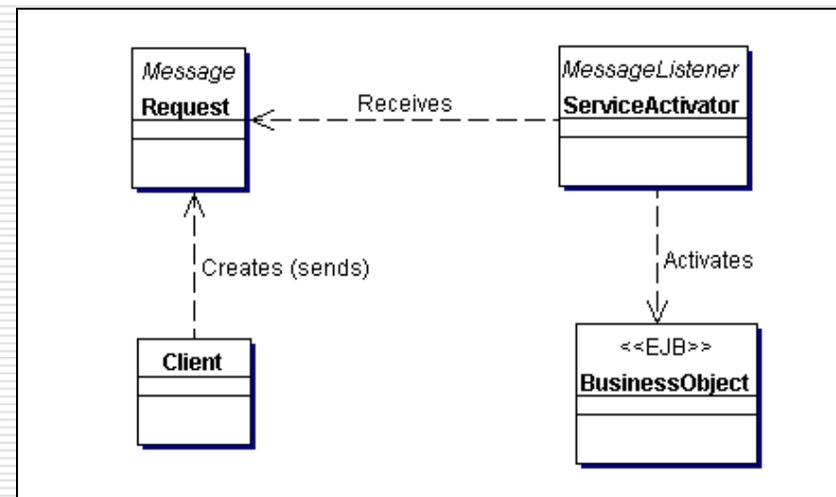
Vocab/MetadataManager

Configurable Interface Pattern

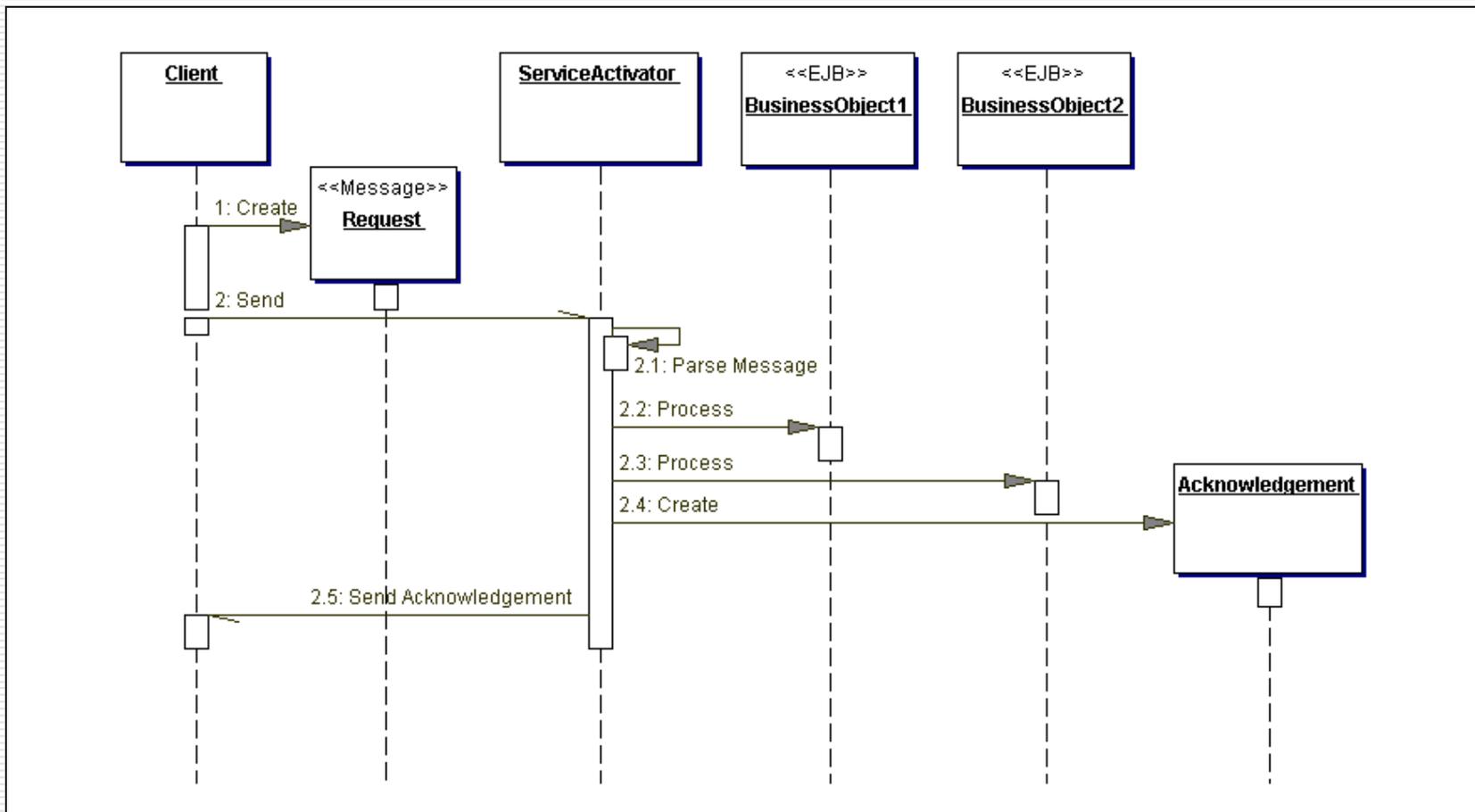
- ❑ The VocabManager retrieves controlled vocabularies and metadata via an interface pattern.
- ❑ The implementations associated with this interface are configurable. This allows us to plug caArray into either the caCORE API, or a different metadata repository, or an XML metadata descriptor file.
- ❑ Via this interface, the VocabManager will allow the caArray application to perform attribute type checking, validation and population of enumerated lists and other controlled vocabularies.

MAGE-ML Export/Import and File Upload Service Activator Pattern

- ❑ Use a Service Activator to receive asynchronous client requests for MAGE-ML import and File Uploads.
- ❑ On receiving a message, the Service Activator locates and schedules the MAGEParser module to parse MAGE-ML documents and invoke the appropriate business service component such as ExperimentManager or ArrayDesignManager to persist the resultant MAGE-stk objects asynchronously.
- ❑ Similarly after the data files are uploaded by the FileUploader module, MAGEParser is scheduled to parse and persist the data asynchronously.
- ❑ Benefit
 - Asynchronous processing of time & memory intensive operations



MAGE-ML Export/Import and File Upload Service Activator Pattern



NetCDF Data Capture Strategy

- ❑ Binary representation of numerical data.
- ❑ Easily manipulated by R and Java.
- ❑ Fast and efficient direct access to data matrices.
- ❑ Convertible to XML or tab-delimited forms
Java API can be written for transparent access.
- ❑ Self describing format.

Persistence Abstraction

- ❑ caArray uses Apache's ObjectRelationalBridge (OBJ) as the Object Relational Mapping tool that allows transparent persistence for plain old Java Objects (POJO's) in the MAGE-stk toolkit against relational databases.
- ❑ caArray utilizes OBJ's persistence facility to manage the connection to the data source, store, and restore MAGE-stk objects.
 - The mapping of MAGE-stk objects and their relationships to the database entities is described in OBJ repository files to allow OBJ to know how to persist or restore them.
- ❑ OBJ is also the current standard for caBIO allowing code reuse across projects.
- ❑ Benefit:
 - Decoupling between Object and Data source layers

Lazy Materialization Pattern

- Lazy materialization refers to only loading data collections when they are actually required. In caArray "lazy loading" aka "lazy materialization" is a capability that is implemented via OJB collection proxies.
- Lazy materialization is implemented using a Proxy to make the calls to manipulate the collection. This can help you in reducing unnecessary db lookups, and object materialization. Example:
 - Say you load a ArrayDesign object from the db which contains a collection of 15000 Feature objects.

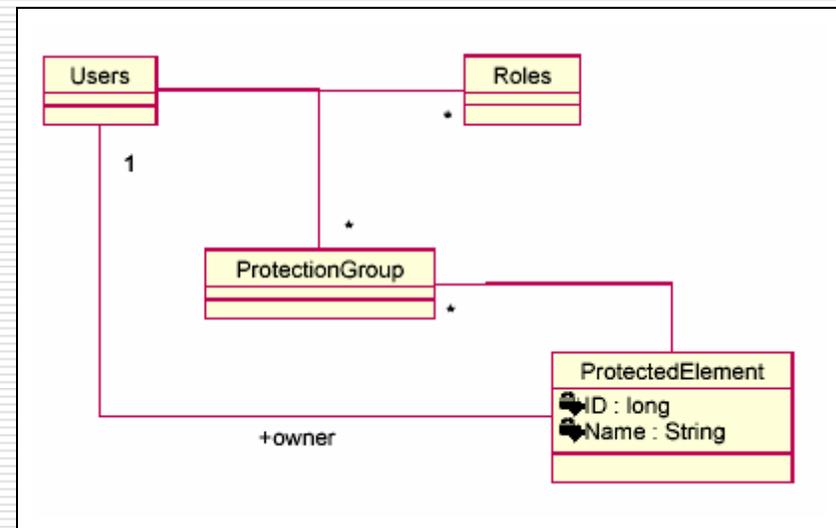
Without proxies all 15000 Feature objects are immediately loaded from the db, even if you are not interested in them but just want to lookup the description-attribute of the ArrayDesign object.

With a proxied class, the collection is implemented via a proxy, that implements the same interface as the "real collection" but only materializes the objects in the collection when necessary. Once you access such a proxied collection it loads its collection objects by OJB and executes the method call.

- Since the actual Java class uses an interface for the collection, the OJB proxy can be utilized without changing, or creating dependencies within, the classes code.
- Benefits
 - Allows materialization of objects efficiently in terms of both time and memory usage.

User Access Roles

- ❑ **User Manager**
 - A user who is responsible for creating data sharing consortiums, users, organizations and people.
- ❑ **Configuration Manager**
 - A user who is responsible for managing the protocols, hardware, software and array designs.
- ❑ **Ontology Manager**
 - A user who is responsible for managing dynamic vocabularies.
- ❑ **Experiment Manager**
 - A user who is responsible for annotating biomaterials and maintaining the data associated with experiments.
- ❑ **Curator**
 - A user who is responsible for the integrity of the data for a particular group or a set of groups.
- ❑ **Repository Curator**
 - A user who is responsible for the integrity of the data stored in the repository.
- ❑ **Data Owner**
 - Any user that is currently listed as the owner of any protected data element in the system.
- ❑ **User**
 - Any individual who will use the system to submit data or search through and utilize existing data.



Data Models

□ caArray data model

- The data model was derived by annotating the MAGE-OM API with Xdoclet tags.
- The Java files were processed by an Xdoclet module (Apache's Torque).
- The process generated the OJB repository.xml file and the SQL DDL schema.
- The schema was then optimized for performance.
- Benefit: Using a code generator, the schema can be regenerated when the MAGE-OM changes.

□ Security data model

- Utilizing NCICB common security data model.

Proof of Concept

- ❑ Developed Protocol Management reference implementation (proof of concept) that implemented end-to-end functionality for protocol management screens.
- ❑ Using the proof of concept, one can perform the following:
 - Search, Add, Update, and Delete Protocols
 - Add Hardware/Software/Parameter to Protocol



NCI CB caArray Cancer Array Informatics

Software Design Follow-Up

May 17, 2004

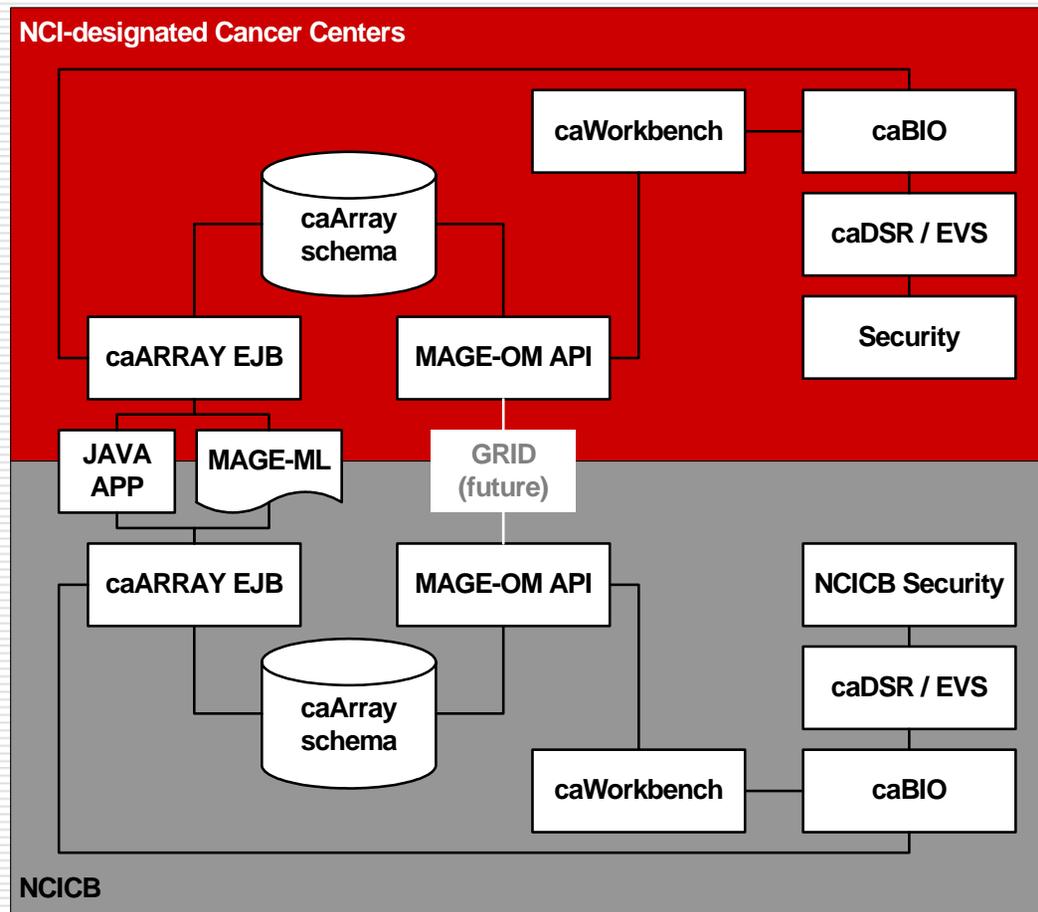
caArray Design Follow-Up

- ❑ Access to caArray via a Federated Architecture
- ❑ Timelines for incorporation of reporting/analysis modules in caArray
- ❑ Timelines for incorporation of solution for issues raised by use of MGED Ontology within a MAGE-ML document.
- ❑ Timelines for use of the caDSR and creation of CDEs in caArray
- ❑ Additional functionality need in MAGE-OM API

caArray Interfaces

- caArray provides access to its data via the following interfaces:
 - Programmatic
 - MAGE-OM API
 - caArrayEJB API via data transfer objects (DTO)
 - caArray Submission Portal
 - Document
 - MAGE-ML import/export
 - Native Affymetrix and GenePix file formats

Potential caArray Configuration



caArray Interfaces: Mage-OM API (cont'd)

- **MAGE-OM API** :Provides fine grain search and retrieval of all caArray data via a caBIO-like RMI based API.

For Phase I:

- The MAGE-OM API will be modified to map the MAGE objects to the new caArray database schema.
- RMI Security module will need to be incorporated and tested for user/group level data access.
- NetCDF API logic will need to be incorporated for faster retrieval of data

caArray Interfaces: Portal (cont'd)

- **caArray Submission Portal:** Via a user friendly GUI, caArray allows the users to submit, annotate and retrieve experiments.

For Phase I:

- MIAME 1.1 compliance
- Affymetrics & GenePix native file capabilities
- MGED Ontology for annotation
- MAGE-ML import/export

caArray Interfaces: caArrayEJB API (cont'd)

- **caArrayEJB API:** Provides transaction control, asynchronous processes, service location, common security and distributed capabilities for submission and retrieval of Microarray Experiments, MAGE-ML documents and its associated data files.

For Phase I:

- The caArrayEJB API will provide the above functionality via the caArray presentation layer.
- The caArrayEJB source code will be packaged and documented for installation at other cancer centers.
- The caArray team will only document how the caArrayAPI might be used for federated submission of micro array data.

caArray Interfaces: Federated (cont'd)

Future Phases:

- The actual proof of concept for federated submission of micro array data will be addressed during a future phase.

caArray Interfaces: Documents (cont'd)

- For MAGE-ML Export/Import:
 - caArray is planning to support both DataInternal and DataExternal .
 - The actual format of the DataExternal/ DataInternal is specified within the XML, such as tab delimited and others.
 - For Phase I, the Mage-ML import/export utility will only support the tab delimited format files such as output from Excel.

- Submission and retrieval of native Affymetrix and GenePix files

Timelines for incorporation of reporting/analysis modules in caArray

- For Phase I:
 - caWorkbench & webCGH will be modified and tested with the new MAGE-OM jar file.
- Future Phases (Iterative Development):
 - Following Initial release, XpressionWay and other reporting/analysis modules will be incorporated into caArray in iterative release cycles.

Timelines for incorporation of solution for issues raised by use of MGED Ontology

□ For phase I:

- Java Binding solution will be incorporated for MGED Ontologies associated with BioMaterial object.
- MAGE-OM API will also need to be modified to incorporate this feature.

□ Future Phase:

- To handle all MGED Ontologies, A true ontology to Java class code generator will need to be developed.

Timelines for use of caDSR and creation of CDEs in caArray

- For phase I:
 - Configurable Interface Pattern used to abstract the source of controlled vocabularies and metadata behind the caArrayEJB API.
- On going:
 - Arrange to get the MAGE-OM API loaded into caDSR via UML loader
 - Looking into creation of CDEs for caArray GUI
- Future phase:
 - Addition of caArrayEJB APIs Data Transfer Objects (DTOs) into caDSR

Additional functionality need in MAGE-OM API

□ For Phase I:

- RMI Security module will need to be incorporated and tested for user/group level data access.
- NetCDF API logic will need to be incorporated for faster retrieval of data
- Java Binding solution for MGED Ontologies associated with BioMaterial object will be incorporated.

Wrap Up

- Questions?
- <http://caarray.nci.nih.gov>