Experiences Using the DataFoundry Meta-Data Based Infrastructure

Terence Critchlow

Ron Musick

Center for Applied Scientific Computing

Tom Slezak

Joint Genome Institute / BBRP

Lawrence Livermore National Laboratory

Biomolecular Informatics Workshop

February, 2000

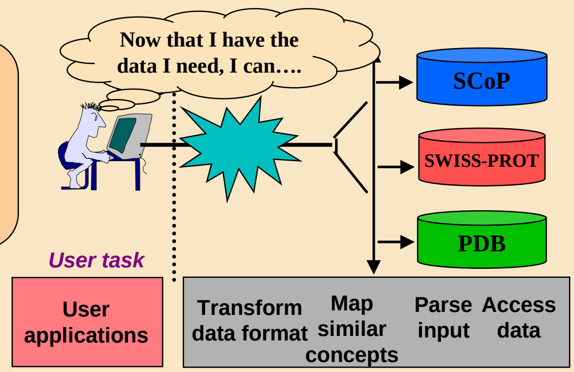
Outline

- Towards an ideal environment
- Implementation challenges
- Summary

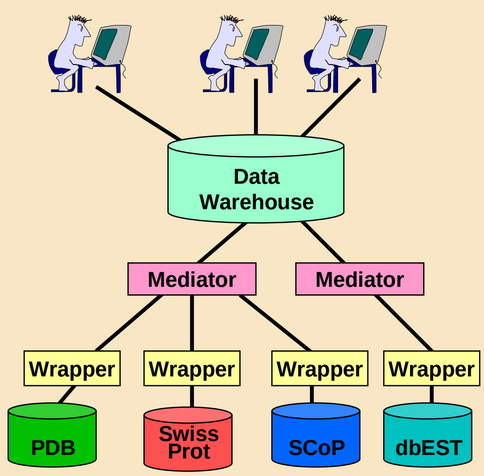
What is our ideal environment?

A *single* location that provides *effective* access to a *consistent* view of data from *many* sources through an *intuitive and useful* interface.

Businesses use data warehouses to accomplish this.

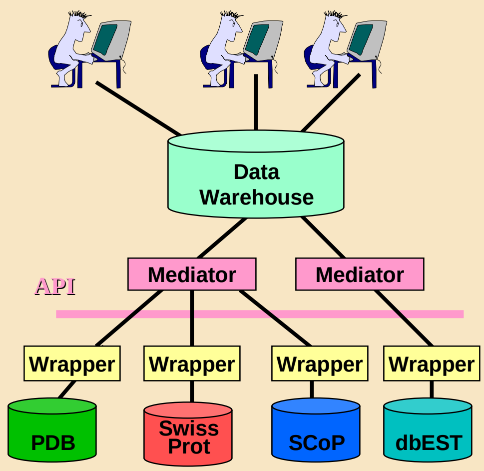


A brief overview of a mediated data warehouse architecture.



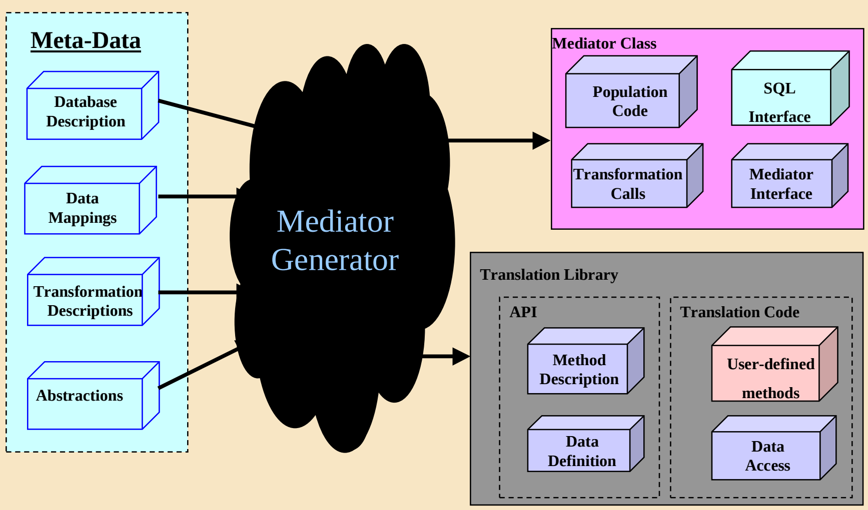
- **Interfaces**
  - provide intuitive access to the data
- **Warehouse**
  - stores a consistent view of data in a local repository
- **Mediator**
  - transform data from source format to warehouse format
- **Wrappers**
  - read data from source into internal representation
- **Data Sources**
  - autonomous, heterogeneous data repositories

Key insight.



Extensive use of meta-data and a well-defined API can dramatically reduce maintenance costs.

The DataFoundry approach.



Challenge 1:
Overcoming language constraints.

Problem
C++ does not handle memory deallocation automatically

Approach
Automatically generate destructors for each class.

Complications
- complex data structures automatically created
- destructors called at multiple points in code
- multiple objects may reference the same data
  - destructor may be called multiple times for one instance

Challenge 1:
Overcoming language constraints.

Solution

- Add semantic information about class generation to the mediator generator in order to create complex destructors

- Use static data structures inherited from the base class to keep track of which data structures have been deallocated.

Challenge 2:
Defining the underlying data warehouse.

Problem
A single, unified schema is simpler to query, but slower and less space efficient than having multiple schemata.

Approach
Define multiple schemata for efficiency.

Complications
• Mediators need to map to different schemata.
• Interface needs to query multiple schemata.

Challenge 2:
Defining the underlying data warehouse.

Solution

- Define schemata that have few overlapping concepts.
- Generate a unique mediator for each target schema.
- Define interfaces to the database that query over all relevant schemata.

Challenge 3: Keeping the data current.

Problem
Warehouse data needs to be updated regularly.

Approach
Use automatic ingest routines to identify modified data and call wrappers/mediators to enter into warehouse.

Complications
- Ability to handle in-place updates of data.
- Identifying parsing errors and schema changes.
- Ensuring confidence in resulting data (validation of updates).
- Needs to be reliable or users won't trust it.

Challenge 3: Keeping the data current.

Solution

- Use a specialized script to identify source data changes, download files and call wrapper/mediator programs.
- Schedule script to run regularly (using cron command).
- Email results of ingest to administrators for validation.
- Define integrity checking queries to identify if the database becomes corrupted.
- Let the users see when the last update has taken place.

Challenge 4:
Wrapping the data sources.

Problem
Wrappers need to parse complex, flat files to read data.

Approach
Use hand-coded parsers to read data from files and pass it to automatically generated mediators.

Complications
- File formats are complex.
- Poor documentation of data format and allowable options.
- Hand entry of data introduces significant errors.
- Need to be able to read all of the data.

CASC

Challenge 4:
Wrapping the data sources.

Solution

- Use data to provide additional details on allowed format.

- Use the same, automatically generated, data structures as the mediators.

- Develop code with excellent error-handling capabilities.
  - Identify which entry caused the error.
  - Automatically correct the error if possible.
  - Continue processing with next entry if possible.

- Manually correct files with format errors and re-enter.

CASC

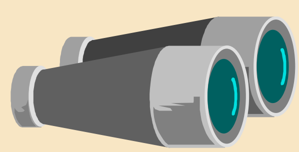
Current status:

Mediator generator in use since late 1998.

- **Has been used to integrate two new sources into the data warehouse.**
  - **dbEST has gone through 2 revisions since data entry**
- **3 sources automatically updated weekly.**
- **Meta-data translation to XML complete.**
  - **mediator generator currently being updated to read this representation**

Possible future directions:

- **Using meta-data to increase interface functionality.**
  - **define allowable queries, and how to execute them**



- **Using meta-data to provide light data integration.**
  - **integrate core data sources into warehouse and dynamically access the rest without interpreting the results**
  - **define set of relevant data sources, how to query them and where to obtain results**

Results:

Integrating SCoP into warehouse that already contains PDB and SWISS-PROT.

Activity/ integration style	manual	meta-data	diff	%diff
understanding SCOP	2.0	2.0	0.0	0
writing wrapper	4.5	2.5	2.0	44%
modifying schema	0.5	0.5	0.0	0
writing mediator	4.0	0.0	4.0	---
modifying meta-data	0.0	1.0	(1.0)	---
total time in days	11.0	6.0	5.0	45%

Results:

Integrating SCoP into warehouse that already contains PDB and SWISS-PROT.

Activity/ integration style	manual	meta-data	diff	%diff
understanding SCOP	2.0	2.0	0.0	0
writing wrapper	4.5	2.5	2.0	44%
modifying schema	0.5	0.5	0.0	0
writing mediator	4.0	0.0	4.0	---
modifying meta-data	0.0	1.0	(1.0)	---
total time in days	11.0	6.0	5.0	45%

Results:

Integrating SCoP into warehouse that already contains PDB and SWISS-PROT.

Activity/ integration style	manual	meta-data	diff	%diff
understanding SCOP	2.0	2.0	0.0	0
writing wrapper	4.5	2.5	2.0	44%
modifying schema	0.5	0.5	0.0	0
writing mediator	4.0	0.0	4.0	---
modifying meta-data	0.0	1.0	(1.0)	---
total time in days	11.0	6.0	5.0	45%

Conclusions

- **Meta-data provides a way to**
  - reduce the cost of integrating new sources
  - reduce the cost of maintaining an existing warehouse
  - reduce the cost of accessing non-integrated sources
  - design a powerful, and *intuitive*, query mechanism

While designing and using a meta-data infrastructure can be more challenging than traditional programming, it is the only way to develop a scalable, flexible, and useseable system.

Experiences Using the DataFoundry Meta-Data Based Infrastructure

Terence Critchlow

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory

critchlow@llnl.gov
www.llnl.gov/casc/datafoundry/

UCRL-VG-137596

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.