Road map for CIPRES software and database interactions

Rutger Vos, Mark Holder, and Terri Schwartz

September 6, 2006

Contents

1 Introduction

We need to clarify the interactions between CIPRES software, Tree Base II, and other efforts for coordinating phylogenetic databases.[1] It would be great if CIPRES software were able to use databases for multiple purposes and provide a programmatic front end to Tree Base II. Unfortunately there are more neat ideas than there are available person-hours to implement, so we need to think hard about priorities (and coordination).

Thinking of CIPRES software as the client, we can list some places where it would be nice to be using a database. Ranked from mundane (from the database research perspective) to difficult, these uses would be:

1. storing configuration data
2. persistent memory for temporary outputs of an analysis
3. a CIPRES service-wide shared memory
4. for checkpointing expensive analyses
5. a rich log of an analysis (primitive lab notebook) by storing inputs, outputs and significant artifacts.
6. a repository for all of the invocations and inputs of an analysis to allow users to playback an analysis (sophisticated lab notebook with some workflow interaction).

[1] Much of this document arose from a discussion between Val Tannen, Jin Ruan, Rutger Vos, Terri Schwartz, and Mark Holder on August 30, 2006. This document is in the CIPRES svn repository, so, when writing the author line I assumed that it would be maintained by the CIPRES-dev participants.

7. a publicly available repository of results (one of the chief roles of Tree Base II).
8. a means of implementing complex queries on phylogenetic objects.

CIPRES-dev really only has resources to think about working on the simple objectives (the first four) and the interactions with Tree Base II and the Crimson databases.

2 MyTreeBase

Most of these tasks don't require all of the data in Tree Base II, so Rutger was going to start working on an implementation of "MyTreeBase" – a local database that would use the Tree Base II schema, but would hold only a particular user's data (mainly artifacts from CIPRES analyses). Using this service would (presumably) make it easier for CIPRES to later add functions that allow a user to deposit an analysis to Tree Base II with very little effort from the user.

The original plan was that MyTreeBase would provide functions like:

```
Receipt putTree(in Tree);
Tree    getTree(in Receipt);
```

to CIPRES software.[2] This is not a trivial contribution at all (even if it isn't groundbreaking from the db perspective). CIPRES components do not (necessarily) share the same address space, so it is not easy to share data between components (other than the arguments and return values of functions). We don't have a clean system for communicating metadata about potentially useful objects that have been stored in a db (but part of that results from the fact that we haven't been able to cache objects in this way, so it was beside the point).

The goals of MyTreeBase overlap with some of the functionality that is proposed by Val Tannen and collaborators.[3] Implementing MyTreeBase functionality in CIPRES would require the definition of the query and deposition functions in IDL (simple methods like `putTree`, and `getTree` methods would be the first stage).

Writing a service that implements the MyTreeBase interface would have a choice of (at least) four strategies for adapting the data structs that CIPRES software uses for trees (and matrices and taxa) to the appropriate database operations.

2.1 Raw SQL

The CIPRES MyTreeBase adaptor composes SQL directly. The stack would be:
Database ↔ SQL2idl adaptor layer ↔ CIPRES service interface

This is a maintenance nightmare because schemas change.

2.2 Via object-relational mapping layer

The CIPRES MyTreeBase adaptor queries through an object-relational mapping layer, so we would have:
Database ↔ object-relational mapping layer ↔ orm2idl adaptor layer ↔ CIPRES service interface

We could pick between

- Java and Hibernate: http://www.hibernate.org/1.html

[2] an adaptor around database could also implement the AsyncTreeIterator interface nicely.
[3] for the time being knowlege of the proposal content is privileged information that Val shared with Rutger and Mark

- Perl and DBIx::Class: http://search.cpan.org/dist/DBIx-Class/lib/DBIx/Class/Manual.pod
- Python and SQLAlchemy: http://www.sqlalchemy.org/

2.3 Use an OO DB

Val suggested using an object-oriented database in CIPRES-dev. The stack:
Database ↔ ZODB objects to IDL adaptor layer ↔ CIPRES service interface

ZODB in Python (http://www.zope.org/Documentation/Articles/ZODB1) is an option. This would not help us connect up to the Tree Base II, so the MyTreeBase code wouldn't contribute to our longer term database needs.

2.4 Via text files

The CIPRES MyTreeBase adaptor interacts with the db using an intermediate file format that both the database and the adaptor can understand. So we would have:
Database ↔ RDBMS-XML mapping layer ↔XML2idl adaptor layer ↔ CIPRES service interface

File format options:

- NEXUS - probably not rich enough (though we do have importers and exporters that need to be used and improved).
- XML - we'd need to develop a schema, but we need to do this anyway

3 To Do

Using an object relational mapping layer (section 2.2) or XML seem like the viable candidates.

Either of these solutions require finding someone to write some MyTreeBase implementation that adapts CIPRES IDL interface to something closer to what the database stores. Depending on the solution we may need other handwritten code (the RDBMS-XML mapping layer may not come for free).

We also need to figure out what database we can easily distribute in order to implement the MyTreeBase functionality. We could punt on this and say that the user has to install a db, in the same way that the users have to have PAUP*. It would be nice to have an easy install though perhaps inefficient database (e.g. SQLite) for users that do not want to install another third party tool.

The first priority is figuring out who has time to start working on an implementation, or, from the CIPRES-dev perspective, an IDL interface for the CORBA view into the underlying model (which would be an encapsulation of the database). The person chosen may end up dictating the tools we use. One of the key targets of Val's group is to: "Develop an extensible core data model for phylogenetic information. The model will include a query language and extensible data structures." It would be great if we could get our system prototyped soon so that we could contribute in concrete ways to the discussion of that goal.