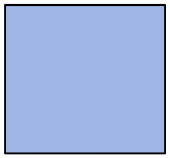
CIPRes Software Development/Architecture

Team leaders:	Wayne Maddison (UBC) David Swofford (FSU)
Other active participants:	Paul Lewis (UConn) David Maddison (Arizona) Mark Holder (FSU) Tandy Warnow (UT) Bernard Moret (UNM) Mark Miller (SDSC) Alex Borchers (SDSC) David Stockwell (SDSC) Terri Liebowitz (SDSC)

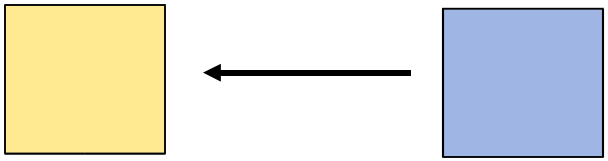
Overview of Architecture

Services

Server modules advertise their services



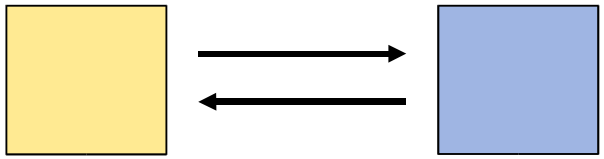
Client modules find and make use of those services



We define various services:
TreeRepository, *TreeSupplier*, *TreeInference*, *TreeEvaluator*, *ImageSupplier*, *MatrixSupplier*, etc.

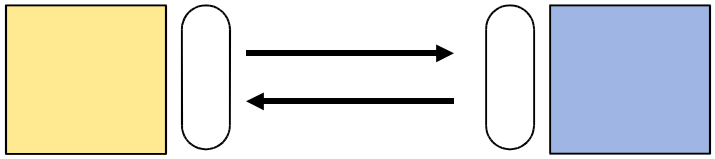
Overview of Architecture

Communication



A communication protocol is specified (currently SOAP). Any CIPRes module must be able to “speak” this. Communication is mediated by Web Services standards (SOAP, WSDL, UDDI).

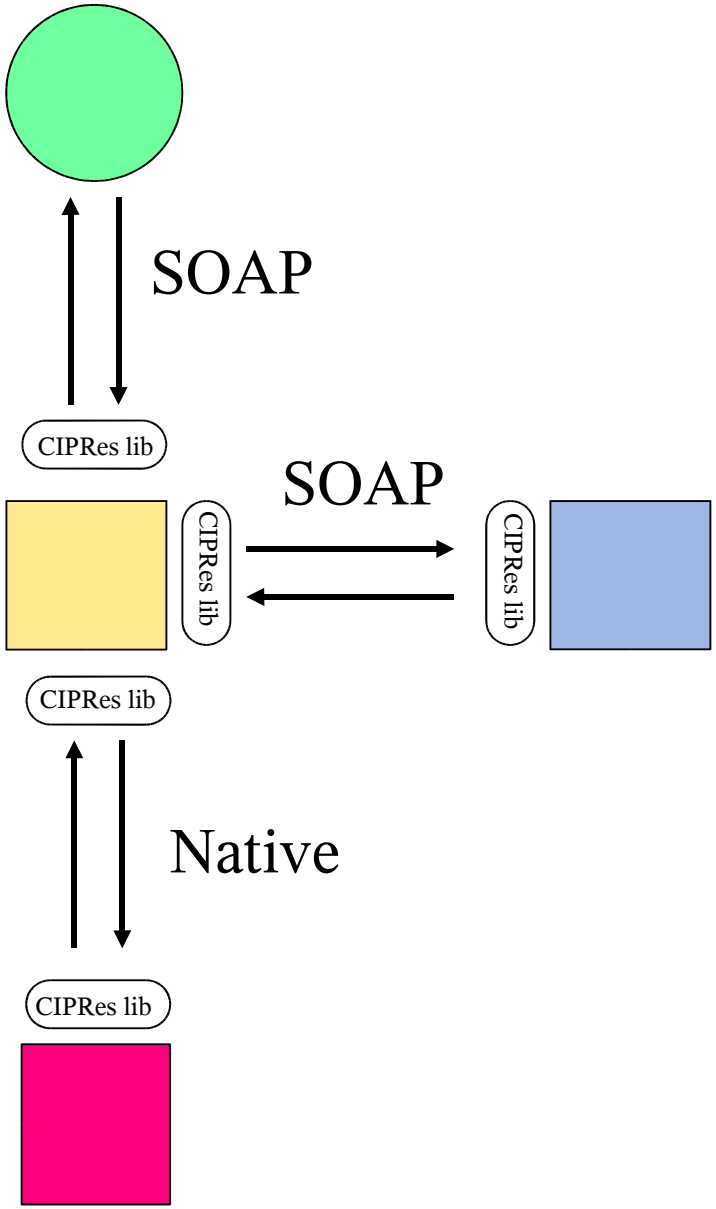
Overview of Architecture



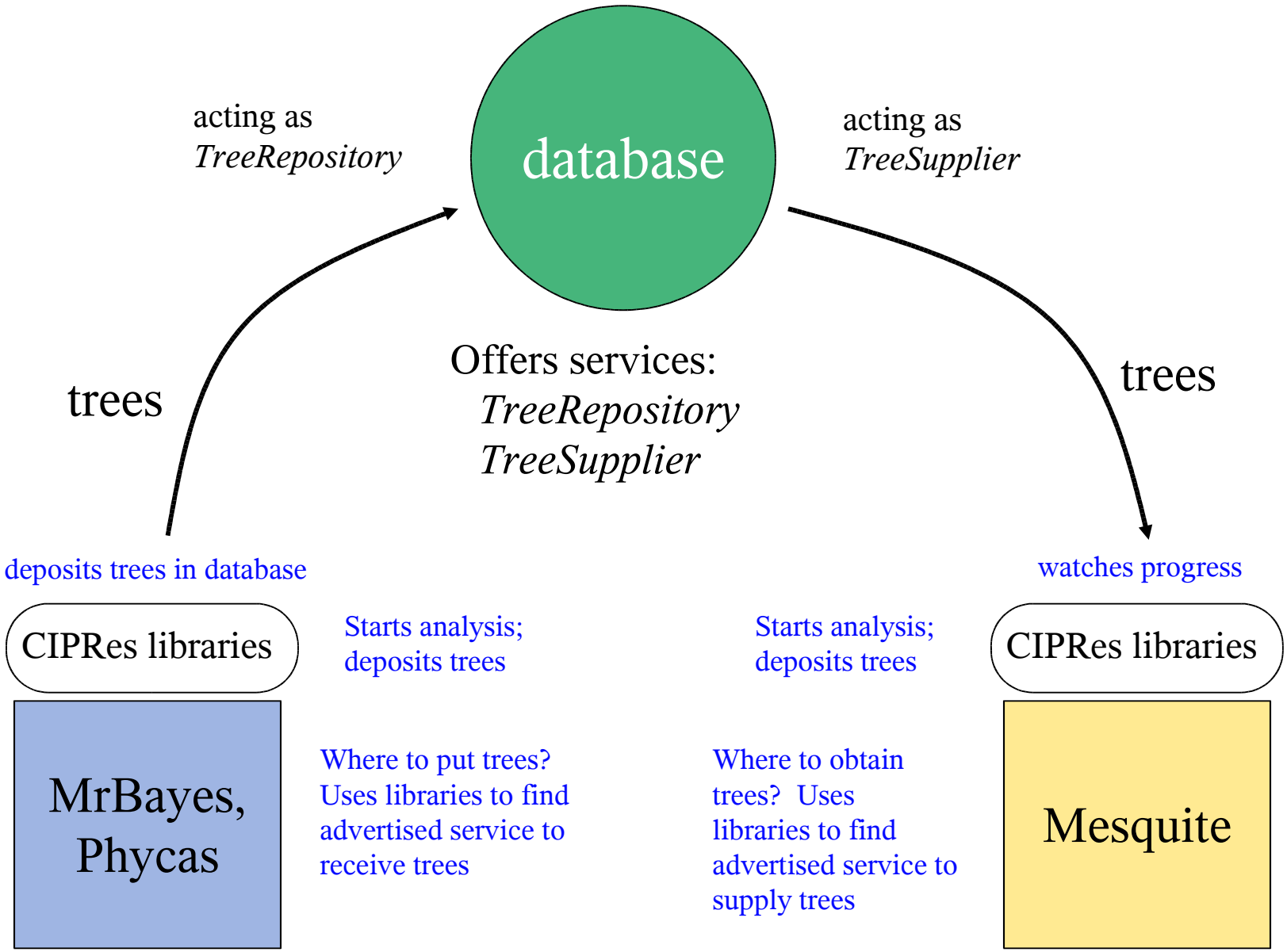
Java, C++, etc., libraries act as glue between programs and the CIPRes module communication system

- Libraries find services, speaking to one another via low level communication protocols
- Libraries can communicate with programs via native C++ or Java objects like *Trees* (as opposed to their XML representation)
- When possible, libraries pass information between modules not via XML but via native memory structures (C++, Java) for speed

Different modes of communication may coexist

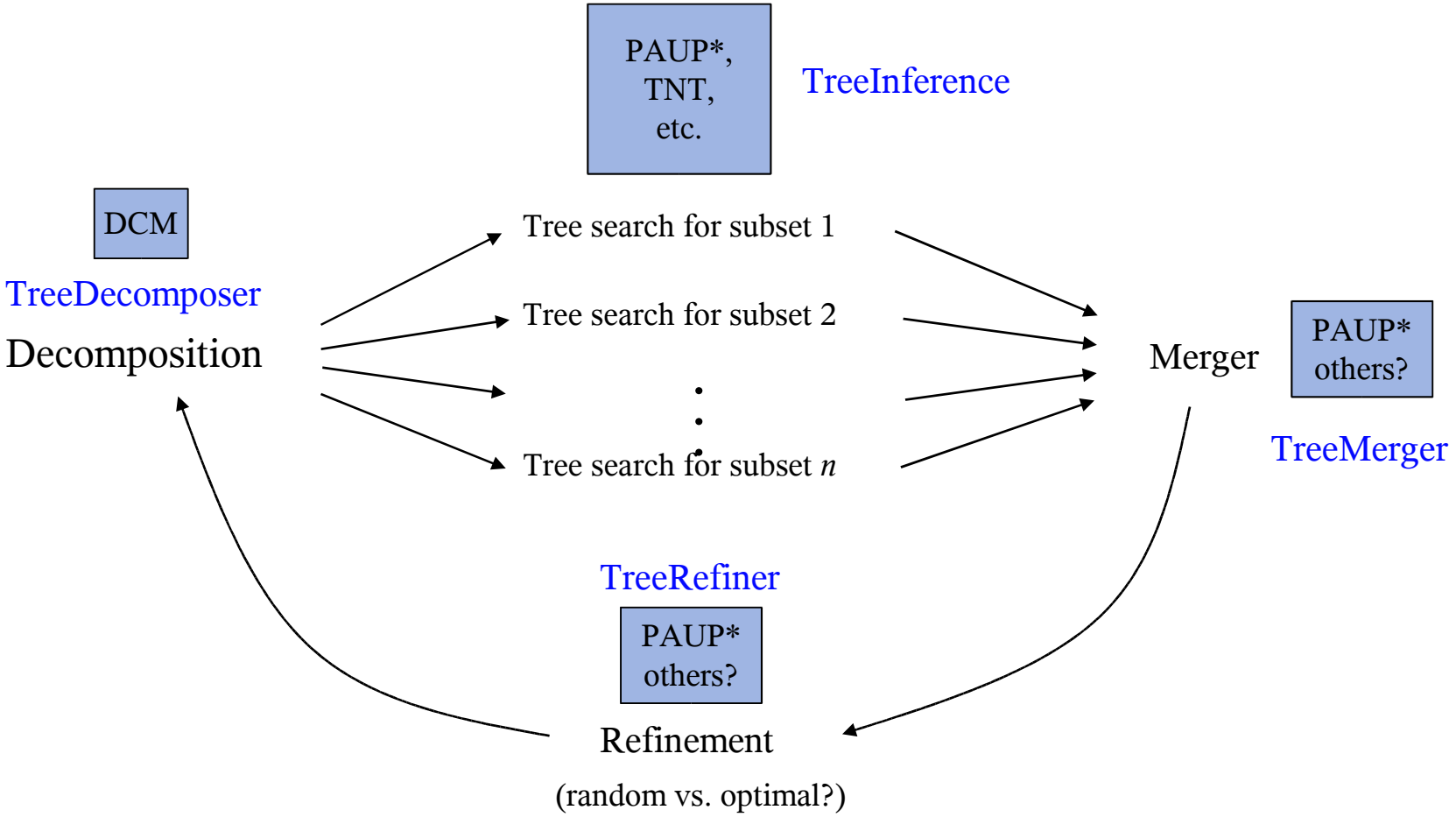


Overview (Example 1)



Overview (Example 2)

DCM (Divide & Conquer)



Other issues/plans

- Notebook/snapshotting/journaling

  Analysis commands and results stored permanently in database; analyses can be “replayed” using different data, updated service modules, etc.

- Choice between alternate service providers

  A system will be designed to “shop” for more specific services (preferences among otherwise equivalent providers & specific compatibility demands)

- “RAD”-style analyses

  Users can design analyses by choosing services from a palette and drawing arcs to illustrate the program flow. Once designed, these analyses can then be bundled as “double-clickable” applications.