Architecture and Interfaces

PhyloTastic

"An application is phylotastic to the extent that, in response to a user query S, it supplies expert phylogenetic knowledge of S, in a form useful for research, in a timely manner."

Turns out we need multiple moving parts to make that vision possible.

Minimal moving parts

- **TNRS** – maps "dirty" labels onto taxa
- **TreeStore** – is queried to identify the tree(s) that best match the taxa
- **Pruner/Grafter** – supplies the minimum spanning tree for the taxa
- **Dater** – supplies branch lengths / node ages proportional to time

PhyloTastic as an MVC app

- Prior to PhyloTastic-I we started thinking of the architecture as **Model-View-Controller:**
  - **Model** – the taxon that become a tree
  - **View** – whatever is the final serialization
  - **Controller** – that which maps user input onto manipulations of the **Model**

The PhyloTastic Controller

- Following the MVC design pattern, we need to architect a **Controller** that knows how to map user input onto manipulations of the **Model** to generate the requested **View**
- This means integrating the moving parts we previously identified as essential to PhyloTastic

How to integrate

- Moving parts are all web services
- No need to adopt a single programming language
- No obvious single way of defining interfaces, could be any (or all) of:
  - SPARQL endpoints
  - WSDL-based interfaces
  - Roll-your-own RESTful APIs

Prior art

- At PhyloTastic-I, the architecture group developed three integrated workflows:
  - **node.js** – Helena Deus developed a JavaScript-based workflow
  - **CGI** – Ben Vandervalk developed a Perl/CGI-based workflow
  - **Galaxy** – Rutger Vos developed a workflow inside the Galaxy workflow environment

Lessons learned

- On the positive side, it is apparently easy to glue the moving parts together as we came up with three working implementations
- On the negative side, we did not produce a conclusive definition of how it all *should* fit together: all solutions were very ad hoc

Standards

- To make PhyloTastic acceptable to scientists, the results must record the provenance of the data
- Some standards can record such metadata better than others
- On the other hand, the combination of web services and megatrees forces us to be concise

What we should *probably* do

- Describe the data types and parameters for each of the services
- Decide on terms for them (i.e. pseudo-ontologize them)
- Integrate the moving parts based on formal description of interfaces

What we should *not* do

- Have deep, long-winded discussions about esoteric ontological concepts
- Try to learn hip new technologies with too little tool support
- Be purists about our approaches