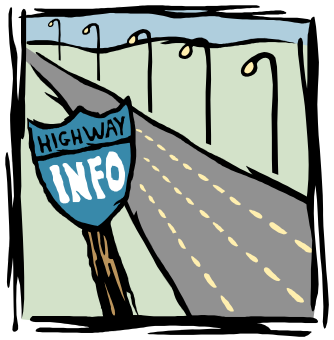
Success Thru Small Loosely Coupled Pieces:

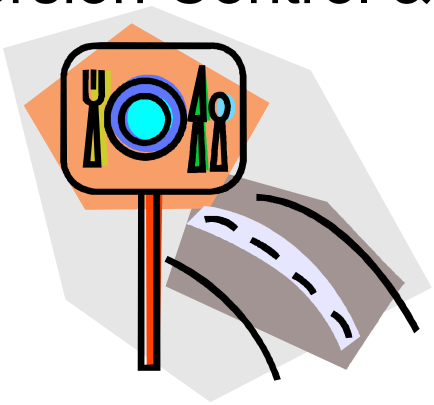
A View From Three Perspectives

Travel Itinerary



- Architectural Goals
- Three Piece-wise Architectures

- Construction Flexibility & Testing
- Version Control & Deployment



- Performance vs Design Flexibility
- Some SOA Advice

Architectural Goals

- **Solve the Problem**
  - **If you can't do this, why bother?**
- **Reduce & Manage Risk**
  - **How can you increase the likelihood of success?**
- **Complete within Time/Budget**
  - **No one likes to wait, or to spend more than they budgeted.**
  - **Particularly the people that pay your salary.**
- **Ease Future Maintenance**
  - **Total cost of ownership is often FAR more than for initial**
  - **development. Keep future costs low as well.**

Three Piece-wise Architectures

Unix Command Line

- Line oriented data formatting
- Small, specific commands
- Pipes, redirection, and simple shell scripting

PathPort

- XML document interchange
- Generic client with domain specific plugins
- Web services for data access and computational analysis

Service Oriented Architecture (SOA)

- XML document interchange
- Services around business processes and data types
- BPEL, BPM, and/or EDA “composition” of services

Unix Command Line

Line oriented data formatting

fstab, passwd, and various configuration files

Small, specific commands

grep, cut, sed, head, tail, find, users, du, diff, ps, sort

Pipes, redirection, and simple shell scripting

| supports simple input-output data flow

Various redirections read/save data streams from/to files

Script control logic enables simple program creation

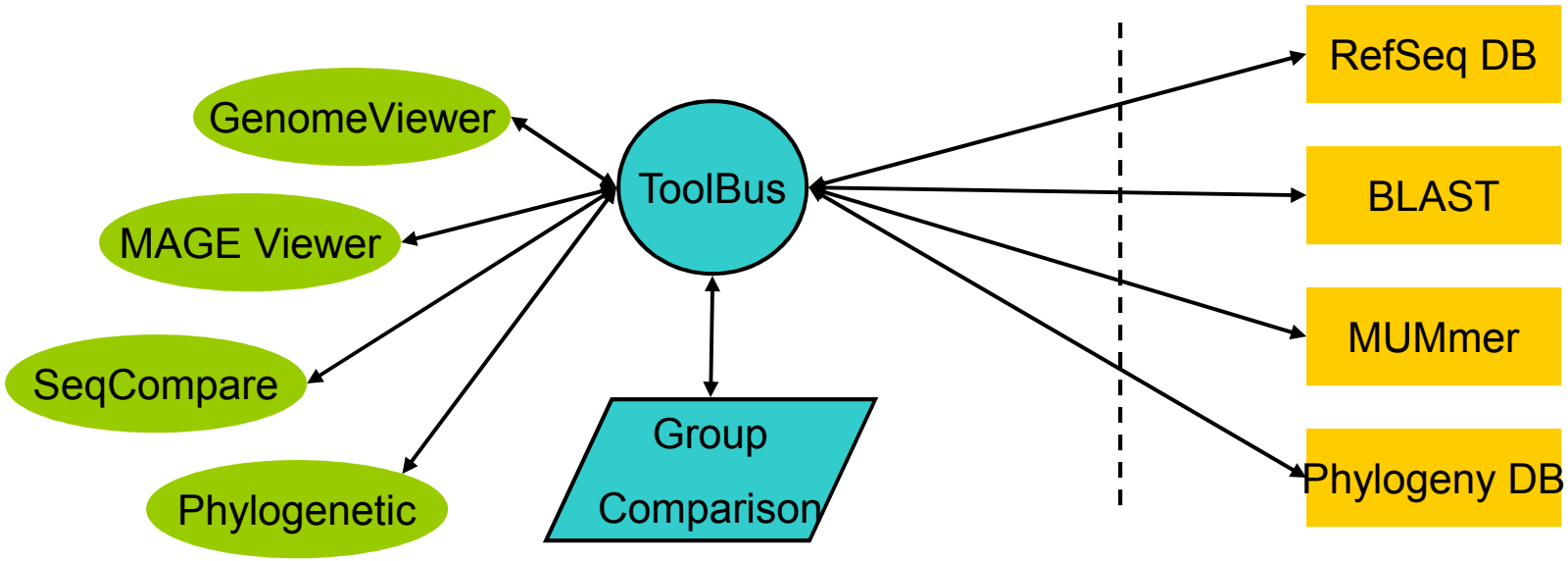
```
ps -ef | tail +2 | sort -n -r +7 -8 | head -1 | cut -c9-14
```

PathPort (http://pathport.vbi.vt.edu)

XML document interchange

DAS, MAGE-ML

Generic client with domain specific plugins



Web services for data access and computational analysis

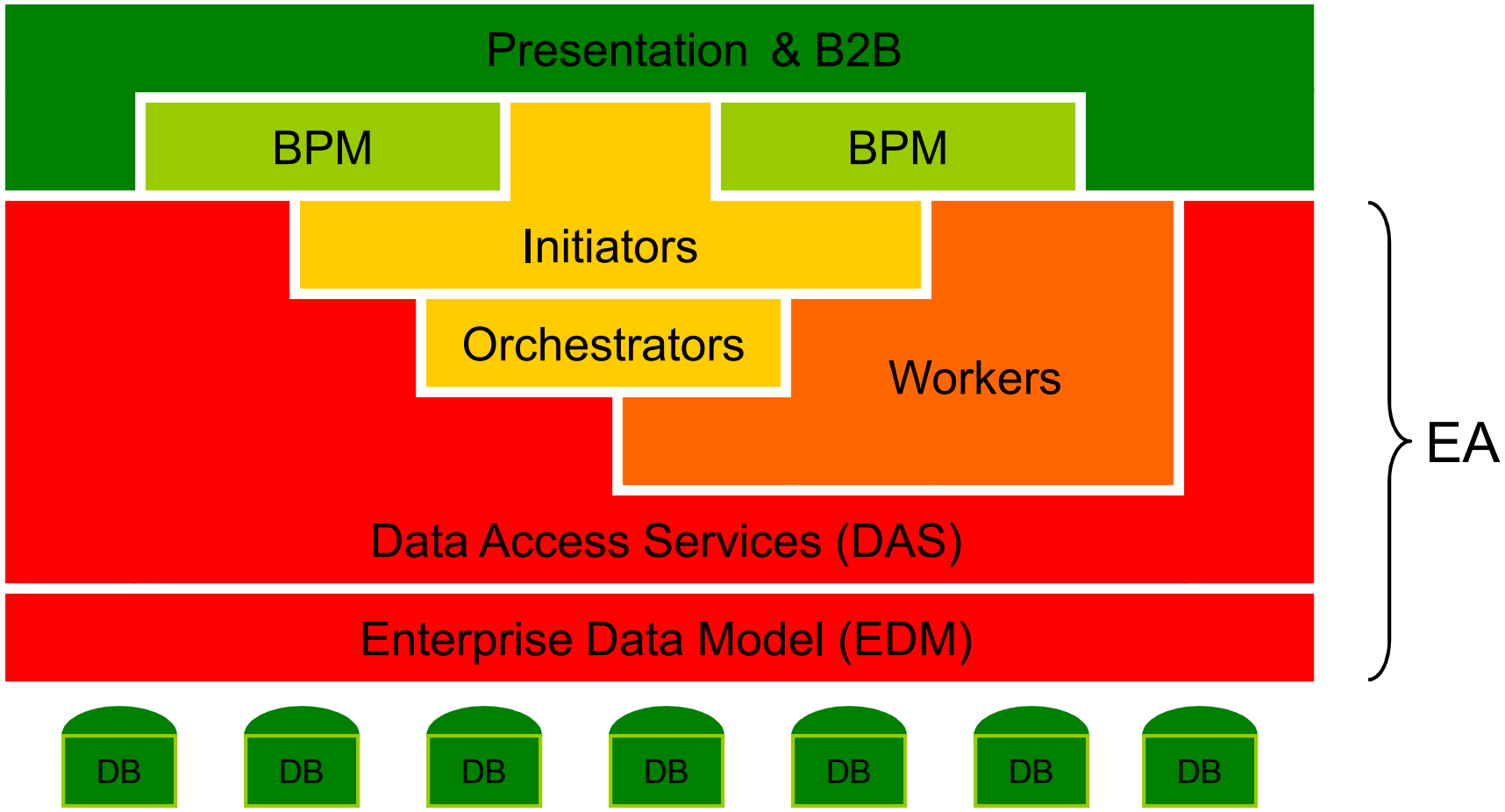
Access to chromosome/genome information

MUMer, BLAST, ClustalW, and other analysis tools

Service Oriented Architecture (SOA)

- XML document interchange
  - Based on XSD, WSDL, and SOAP standards
  - WS-I Basic Profile Complaince
  - Internal (corporate) interoperability standards
- Services around business processes and data types
  - Business aligned data access (policy, claims, dependents)
  - Business transactions (addressChange, premiumPayment)
- BPEL, BPM, and/or EDA “composition” of services
  - Bring Unix like shell scripting to (web) services
  - Automated vs human workflows (and mixes)
  - Short vs Long running transactions/workflows
  - SOAP over HTTP and JMS transport

SOA as Logical Enterprise Architecture



Construction Flexibility

Unix Command Line

- No hidden interactions between commands
- Commands can be designed/implemented independently
- Scripting can be used to build new commands ontop of old ones

PathPort (over 800K LOC)

- No dependencies between ToolBus and web services
- Plugins depend on ToolBus, but independent of each other
- Plugins and web services must agree on XML formats

Service Oriented Architecture (SOA)

- Loose coupling, stateless, and idempotent as much as possible
- Create enterprise datatypes around Enterprise Data Model (EDM)
- Use data access services to create service independence

Testing

Unix Command Line

- Verbatim I/O tests for backward compatibility
- Can use the commands themselves to script auto regression tests
- No good way to determine non-backward compatibility impact

PathPort

- Plugin compatability testing due to shared packages
- Used customized WS tests with scripting for regression tests
- GUI plugin testing required domain expert participation (slow)

Service Oriented Architecture (SOA)

- Migrating from custom clients to MST and Quality Center
- Regression testing aided by Quality Center test library
- End-client (e.g., web browser) testing via WorkSoft's Certify

Version Control & Deployment

Backward Compatible Changes

- Unix: new commands, new options to old commands
- PathPort: new plugins, new data and analysis web services
- SOA: new WSDL operations, new XSD types, making required type element/attribute optional

Non-Backward Compatible Changes

- Unix: removed command option, change in output formatting
- PathPort: Toolbus plugin interface changes
- SOA: Change to operation behaviour, new required element for existing XSD type

Version Control & Deployment

Backward Compatible Changes

- Regression test to ensure backward compatibility
- Replace existing version with new version

Non-Backward Compatible Changes

- Perform impact analysis
- Deploy as “new” capability
  - OR update clients as part of version replacement
    - AND regression test all updated clients

ALWAYS inform users/clients of change!

Performance vs Design Flexibility

Unix

- Character I/O with pipes and new process creation is slow
- vs
- Easy to reuse existing commands to create new ones

PathPort

- Web services are slow w/ larger payloads for data access/analysis
- vs
- Transparently add data and fix bugs with out client redistribution

Service Oriented Architecture (SOA)

- Web services are slow w/ larger payloads
- vs
- Greater reuse and composability for new business processes

Some SOA Advice

- Create organization standards around your platforms
- Leverage your EDM to design the XSD and create contract-first data access services
- Avoid building services to projects, build them for the enterprise if reuse is desired
- Use an up-to-date repository (with impact analysis support) for your XSD/WSDL and actively educate the design/develop community
- Use an ESB/WSM to measure service health proactively
- Think about versioning carefully (XSD-WSDL-code dependencies) and use platform independent load balancing

  Content based routing for non-compatible versions

  OR

  New load balanced end-points for each major version?
- And most importantly, maintain loose coupling between services

The whole is greater than the sum of its parts.

-- Unknown