Enhancing iRODS Integration: Jargon and an Evolving iRODS Service Model

Mike Conway
DICE Center – UNC-CH
IRODS User Meeting 2010

michael_conway@unc.edu
Skype: michael.c.conway

Overview

- Up until today – recent history
- Today – perspectives and issues
- Roadmap – target architecture and getting there
- Discussion, doing this right...

Recent History – Jargon 2.2.0

- Jargon had been in a holding pattern and transitioning from Google Code to IRODS SVN
- Consolidation in SVN for Jargon 2.2.0.
  - Collection of accumulated patches
  - Addition of unit testing
  - Some restructuring of build
- Main purpose – create canonical version and lay groundwork for increasingly aggressive changes.

Recent History – Jargon 2.2.1

- Collect all known patches and reported bugs.
- As many tests as could be written in the time period.
- Main purpose
  - Establish a level of stability.
  - Develop an 'SOP' for Jargon releases.

Recent History – Jargon 2.3.0

- Close on the heels of Jargon 2.2.1
  - Not a lot of time to make big changes
  - Still a somewhat 'conservative' approach
  - Don't break stuff
- More tests, including some 'functional' tests.
  - Multiple 'unreported' bugs caught by testing
  - Testing pays off with a much easier validation of the new IRODS Release
  - Backward compatability testing now part of SOP

Today – Jargon 2.3.1

- Jargon trunk will carry patches to most recent release, and will be test-compliant at all times
  - No patches! Grab the trunk and go
  - The trunk will always be 'better' then the last release
- Jargon 2.3.1 is a branch and feature release. (approx 1 month away).
- Main purpose
  - Get rid of baggage where we can
  - New IRODS 2.3 feature support
  - Refactoring more aggressive as testing better

Today
Starting with a perspective

- I knew some IRODS from enginFrame project
- I knew nothing of Jargon
- Background in enterprise Java development

Before taking about issues

- #1 – Props to Lucas
  - The XML protocol is complex, with many subtle twists.
  - Jargon has been used for a while, and that experience is embedded within the code.

Yes, I know, it's just a simple function to display a window, but it has grown little hairs and stuff on it and nobody knows why. Well, I'll tell you why: those are bug fixes.

-Joel on Software

Issues confronting developers and IRODS domain users

- Jargon is hard to use, especially for folks new to IRODS.
- IRODS is (necessarily) complex and feature-rich.
- Software development has moved on:
  - IoC
  - Testability (mocks, unit testing)
  - Mid-tier standards
  - SOAP and REST-ful services

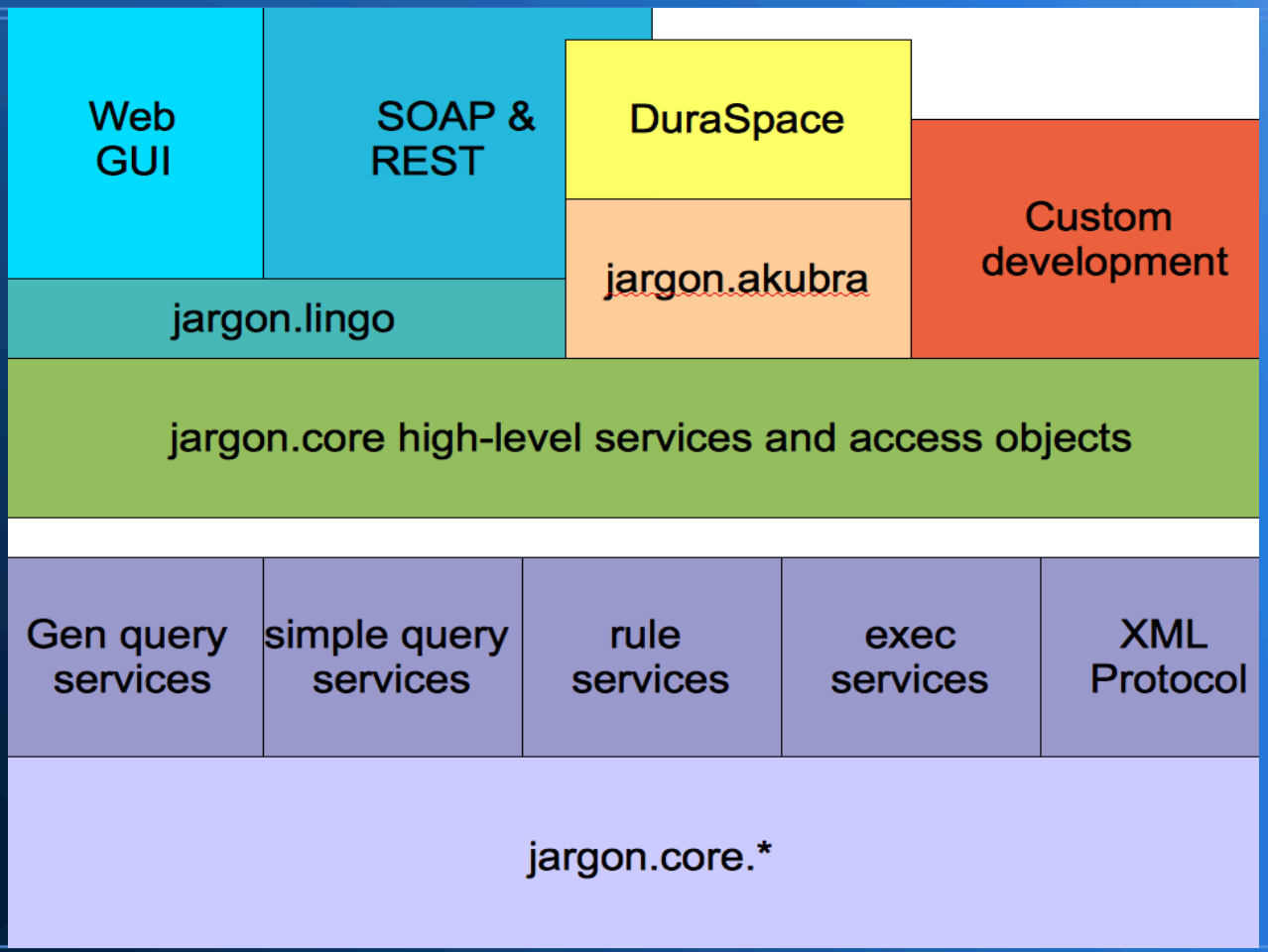
Issues
Interfaces – GUI and API

- Command line doesn't cut it, expectations have changed & IRODS has sophisticated capabilities.
- We can't create a one-size-fits-all GUI interface, and the call for new/custom interfaces will only grow.
- Public vs. private API, redundant pathways.
- Where is the boundary?
- DRY!!!!

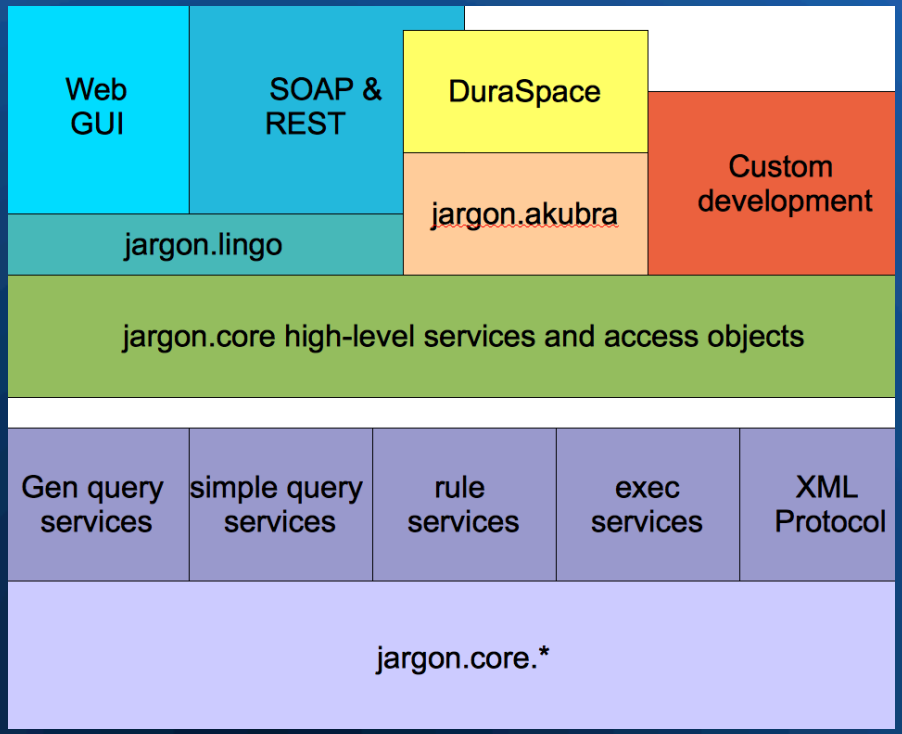
A mission statement

- Jargon will be a tool that feels familiar to developers, admins, and archivists, and that helps open up the IRODS data grid to new domains.
- Jargon will provide a clean foundation that enables new kinds of integration, and plays well with established and emerging platforms and standards.
- Jargon is a stack that works with mature open source tools to extend IRODS interfaces.
- The real action is IRODS, and Jargon will not get in the way.

Roadmap
Jargon is a stack

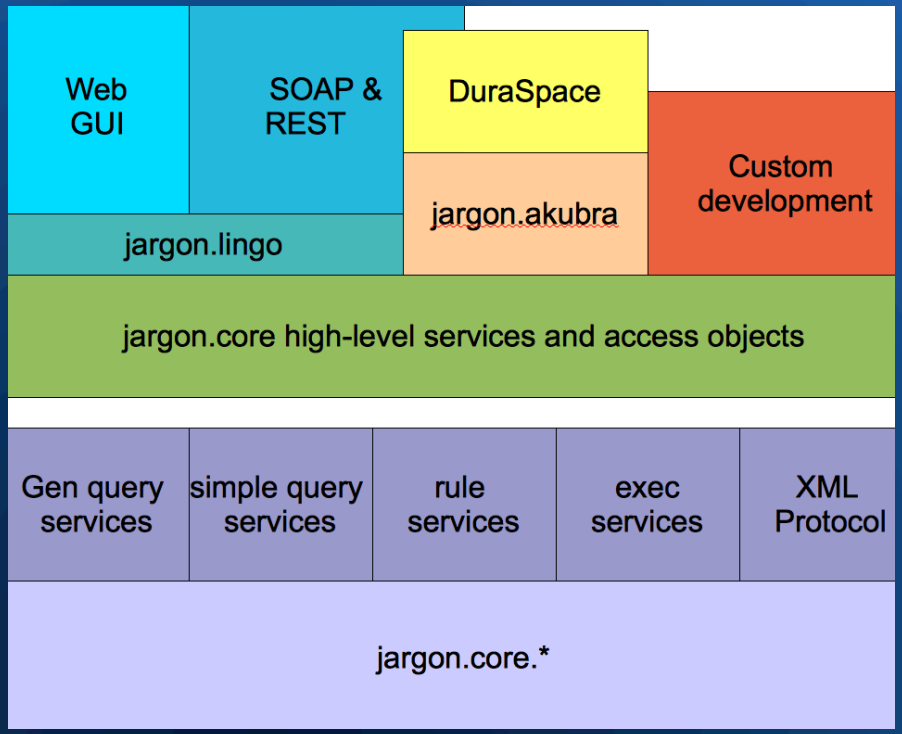


Stack elements



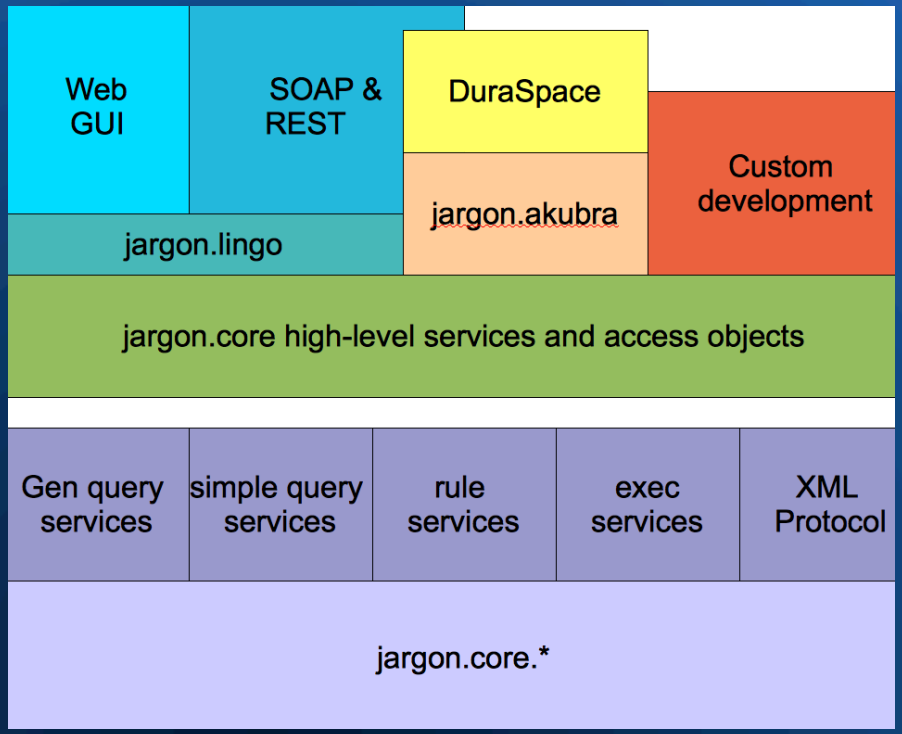
- jargon.core.*
  - Connections
  - low-level code
  - XML protocol

Stack elements



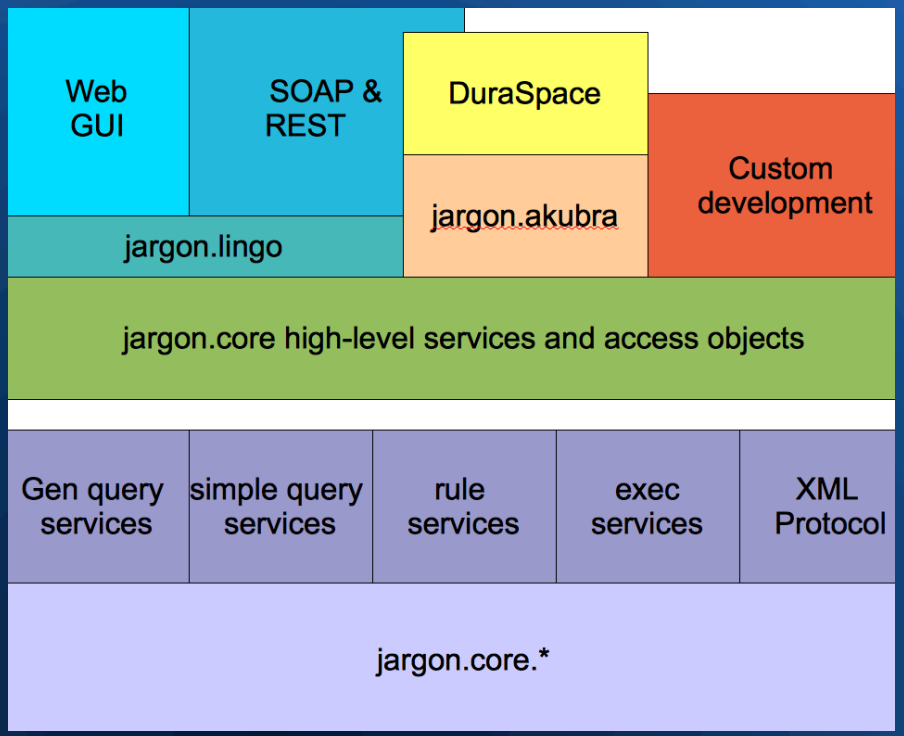
- jargon.core low-level sevices
  - Abstract 'meta' interaction modes
  - Mockable to points above
  - Example: General Query

Stack elements



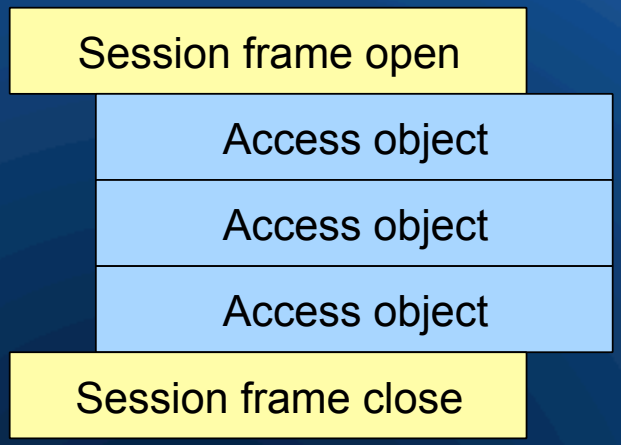
- Public API boundary
  - Only services and POJO's visible
  - No String[][]
  - No Tag{}
  - No sockets or packing instructions

Stack elements



- Access objects and AO's composed into high-level services

Services and AO's

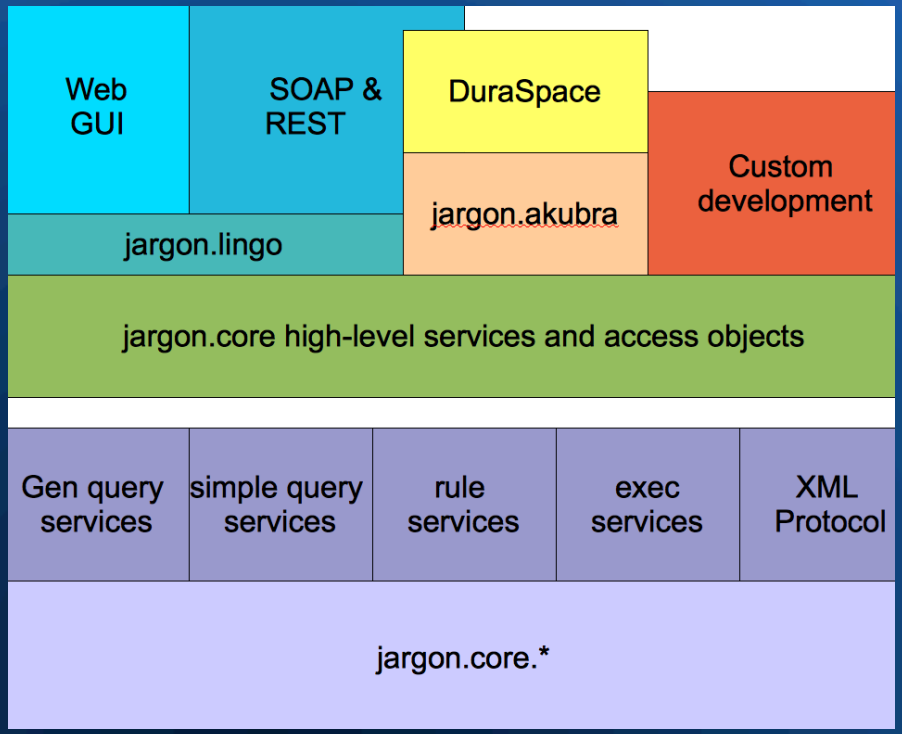


- Styled after Hibernate DAO's
- POJO's in and out of simple methods
- AO's composable into services inside or outside of 'Jargon'.
- Automatically manage connection.

Public API
Easier to use?

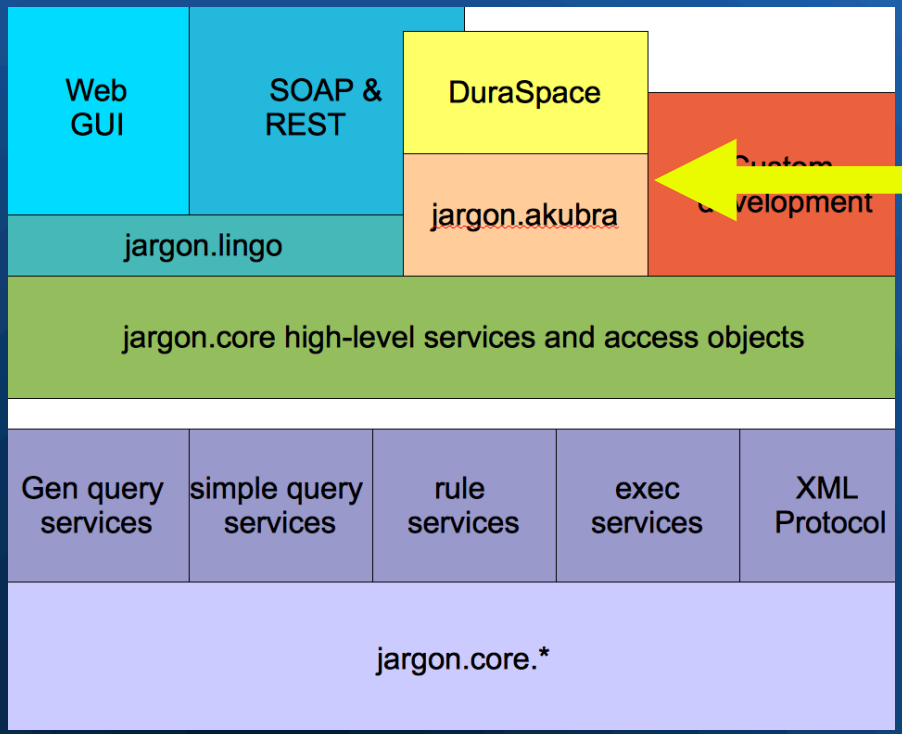
```
public interface ResourceAO {

    List<Resource> listResourcesInZone(String
zoneName) throws JargonException;

    Resource getFirstResourceForIRODSFile(IRODSFile
irodsFile) throws JargonException,
DataNotFoundException;

}
```

Stack elements



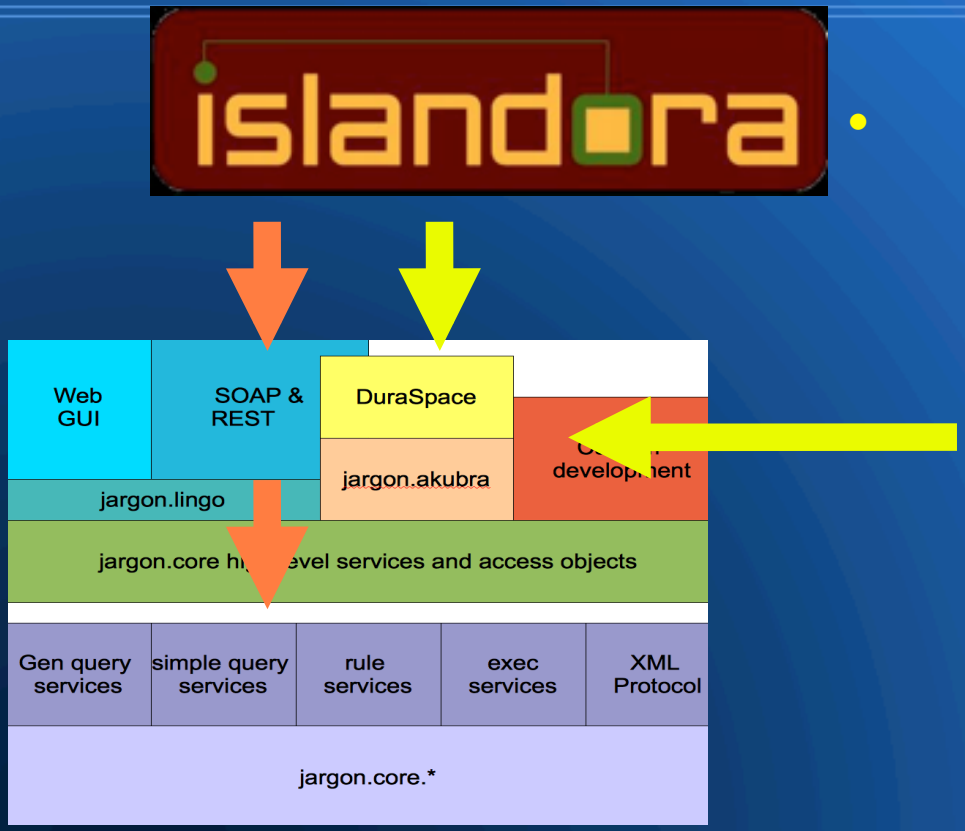
- Above the AO and service level
- Your development
- Integration libraries
- GUI

Stack Elements
Integration libraries



- DuraSpace integration as an example
  - Use Jargon services to integrate IRODS with Fedora
- Other libraries could follow

Stack Elements
Rich integration



- Islandora as an example
  - Leverage integration through Akubra to present IRODS to Islandora
  - Extend through REST-ful access to IRODS-specific functionality

Stack Elements
jargon.lingo

- Out-of-the box web interface
  - Driver for stack development
  - Spring MVC and AJAX
  - JQuery
- Demo

Stack Elements
jargon.lingo

From web to REST-ful interface

```
@RequestMapping("/hotels/{hotelId}")
public String getHotel(@PathVariable String hotelId, Model model) {
    List<Hotel> hotels = hotelService.getHotels();
    model.addAttribute("hotels", hotels);
    return "hotels";
}
```

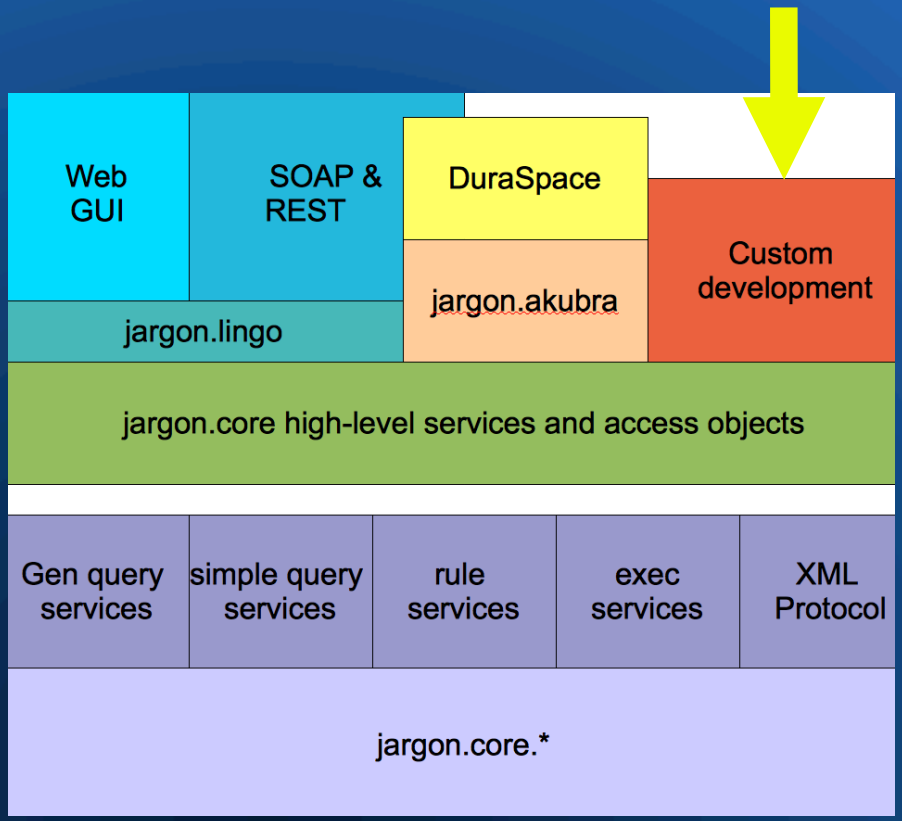
Stack Elements
jargon.lingo

- SOAP/WS-*
  - I don't know specifically yet
    - Axis
    - Metro
    - Spring Web Services
  - Somewhat out-of-scope in that mature tools can implement

Tactics

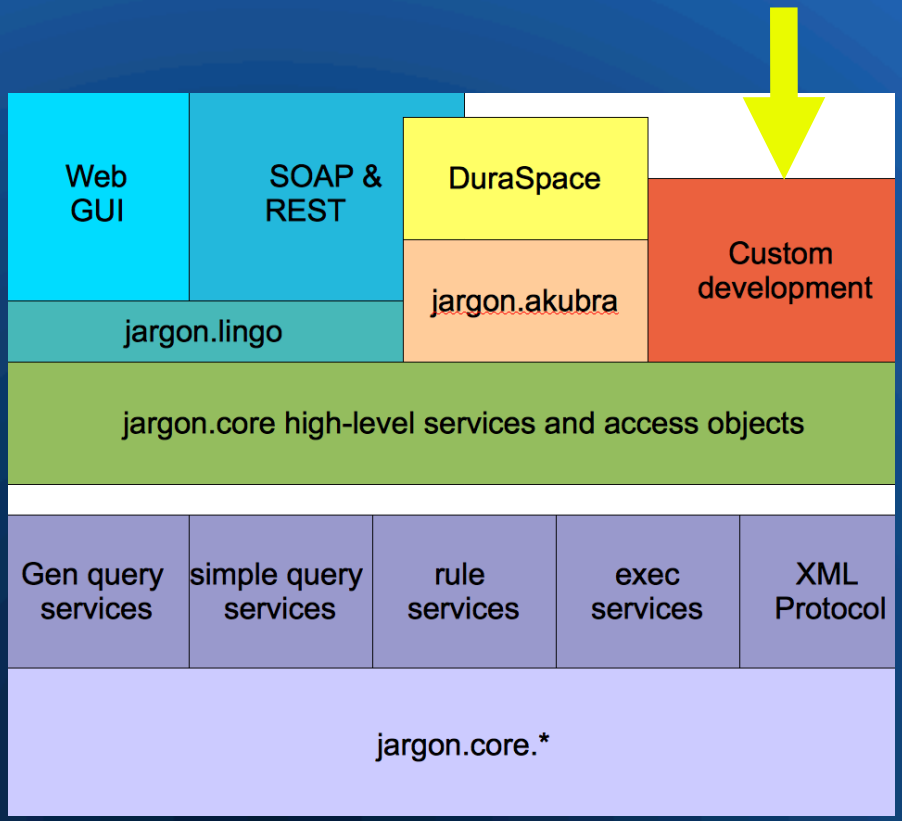
- More tests, quality improvement.
- Parallel development of prototype to define new API and drive mainstream refactoring.
  - New web admin built on prototype
  - Akubra built on prototype
- Improvements move into code stream.
  - Refactoring, testability
  - Code starts to mature
  - Solid launching point for future.

Stack elements



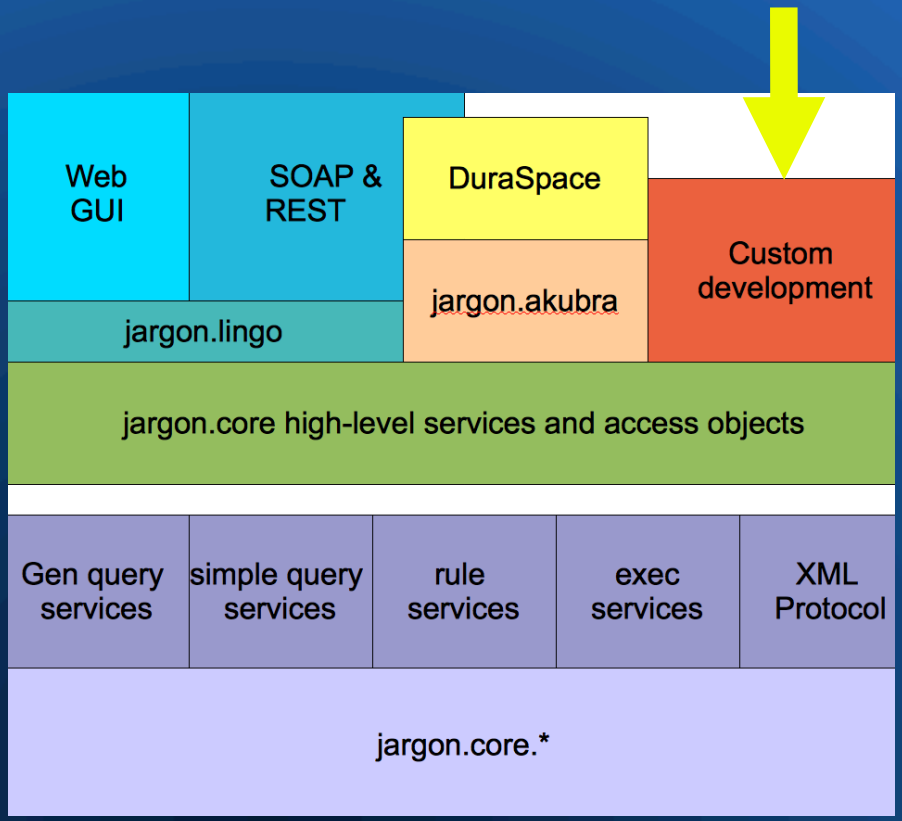
- Your apps, Your GUI's!!!
  - Important that Jargon is an effective enabler.
  - Important to test to run on commodity platforms such as Tomcat, Jetty, Glassfish, JBoss

Stack elements



- We cannot predict your app, but we can observe standards and practices!!!
- Jargon should enable YOUR toolkit.

Stack elements

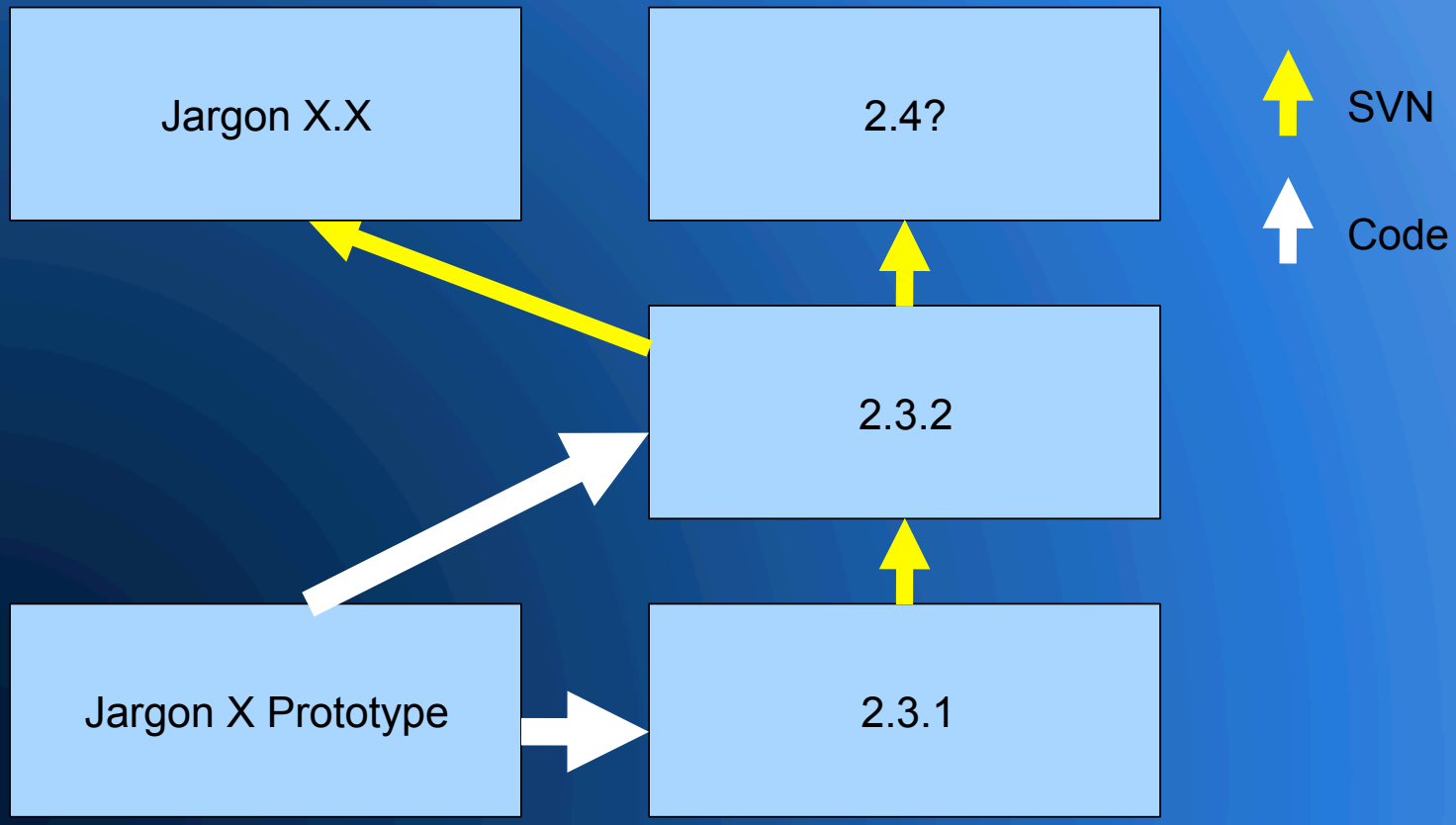


- I am an XXX developer...what about me?
- SOAP/REST
- Messaging?
- Dynamic Languages on JVM
  - Jython
  - Groovy
  - Jruby
  - Scala

Tactics

- Push prototype elements into code stream now
  - Packing instructions factored out
  - Current code broken up into smaller components for reuse
  - Transitional implementation of 'low-level' services
- Parallel development of Jargon X powering web interface and Akubra
  - Steer current Jargon towards prototype architecture and cross-pollinate streams

Jargon X



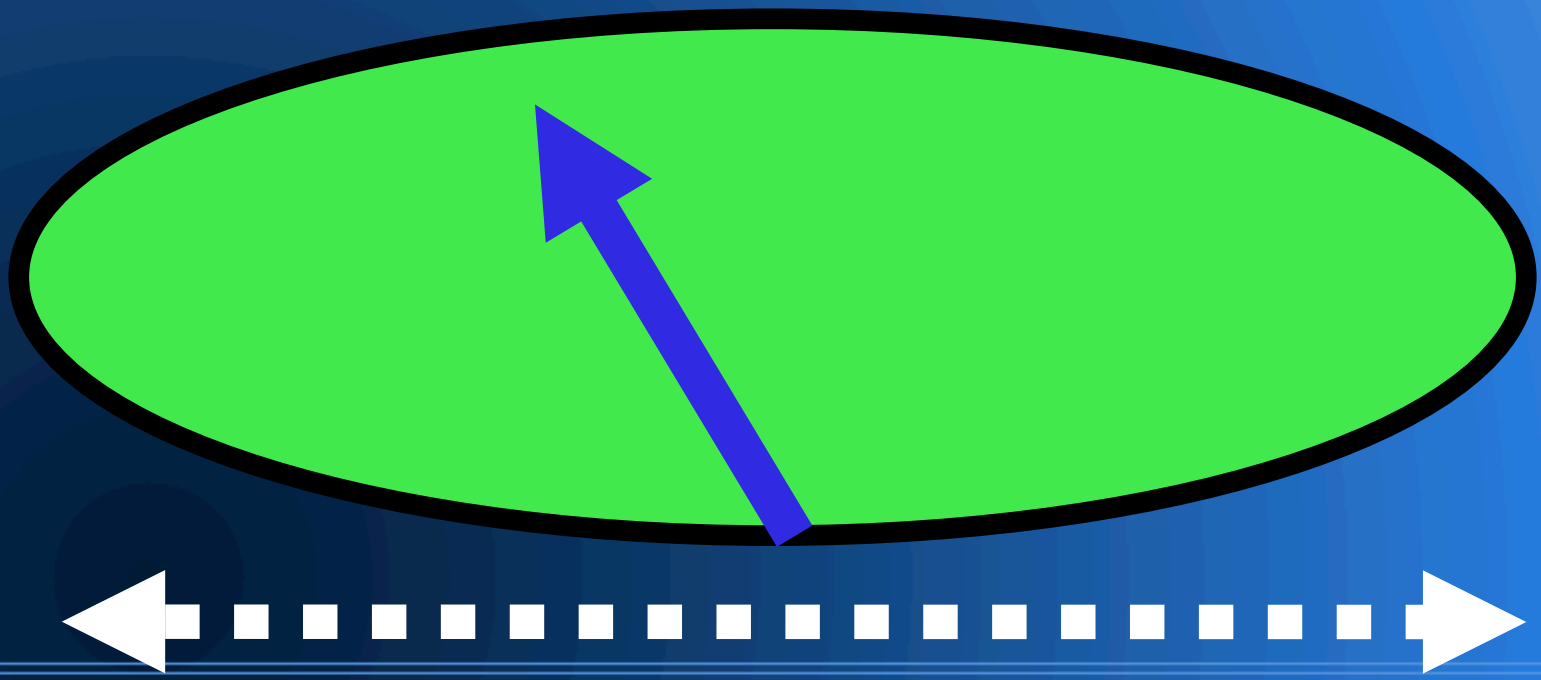
Make improvements now based on prototype, get into code stream for branching down the road.

Discussion

- Doing this right!
  - Use as much code in Jargon enhancements now
  - Break up Jargon into smaller components
    - Better testing now
    - Better re-use
  - Develop real things with Jargon X
    - Eat our own dog-food
    - Build needed capability

Discussion

- Where to set the dial???



Discussion

- How to engage as a community
  - If open source = better software, how can we enhance participation and leverage the community?
    - Other committers?
    - Environment for development
      - Testing
      - Continuous build
      - Process
      - Tools

Discussion

- Designing an interface for
  - Admin
  - User
  - Archivist
- As Jargon-Lingo interface development launches, how can we collaboratively design it?
- Other modalities?
  - SysTray 'icon'?
  - Islandora?

Discussion

- IRODS/Jargon relationship
  - Leveraging IRODS
  - Actions should run with data
  - IRODS interfaces outside of Jargon scope
    - What is available from Jargon, what is presented from IRODS server mechanisms
- Mapping an IRODS Service Model
  - Jargon is part of a much larger stack, what is Jargon's role?
  - What would a service model look like?

Code and Nuts and Bolts

- Connection handling
- Architecture
- Optimization
  - Code optimization
  - Networking optimization
- Jargon-x on SVN

Thanks!

- Your comments, needs, concerns are valuable
- This will not work without you!
- This presentation is as much a question as an answer, look at the prototype!
- Help make Jargon work
  - Contribute and Commit
  - Review and Test
  - Provide use cases
  - Migrate your code into the Jargon stream