Sakai Architecture and Roadmap

Charles Severance

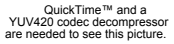
www.sakaiproject.org

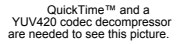
csev@umich.edu

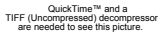
KYOU / sakai

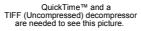
Boundary, Situation

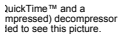
It takes a village to build a CLE….

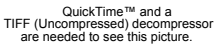


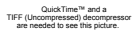


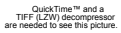


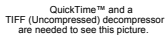


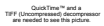


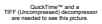


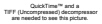












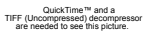


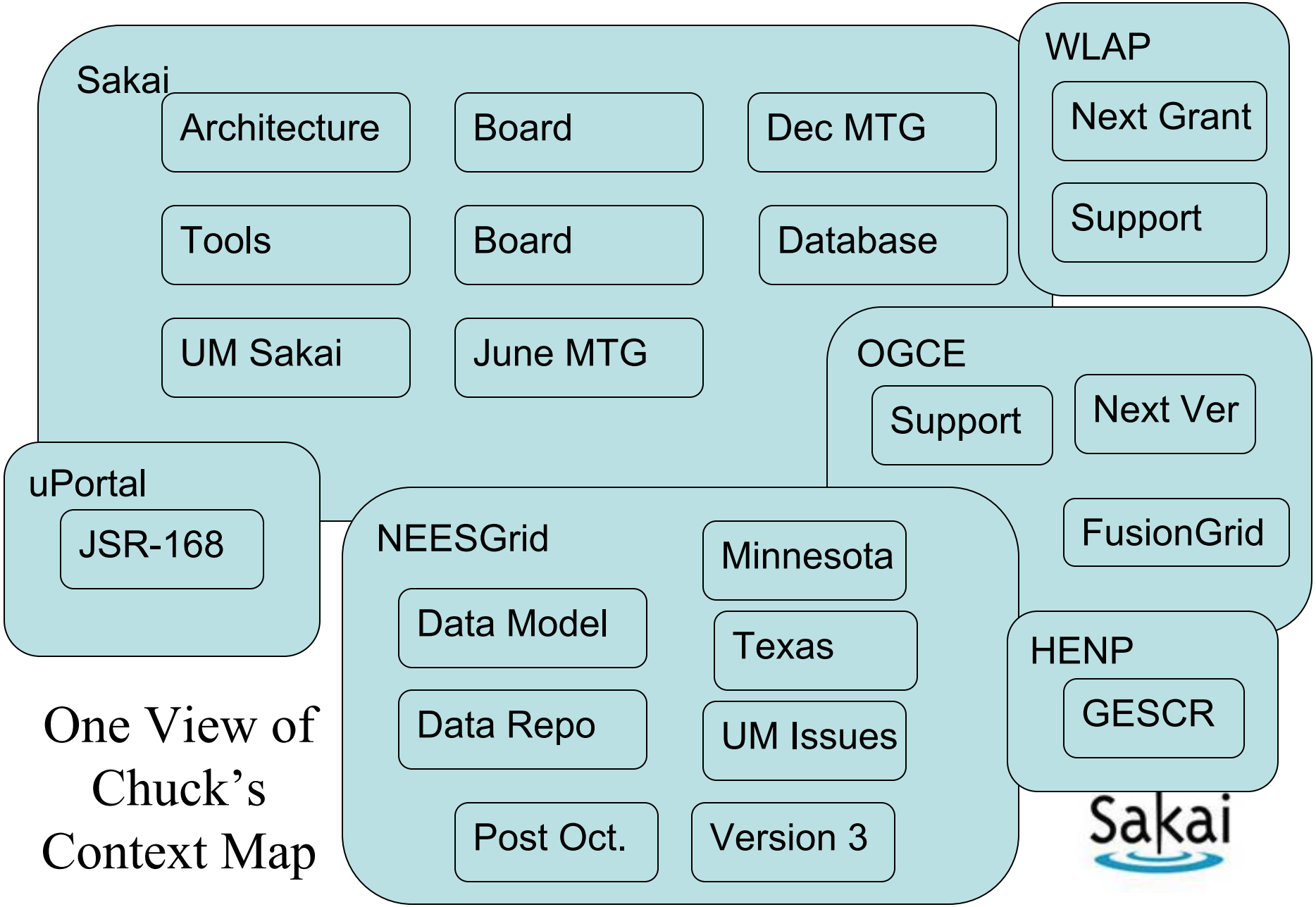
Photo blog: www.dr-chuck.com

sh

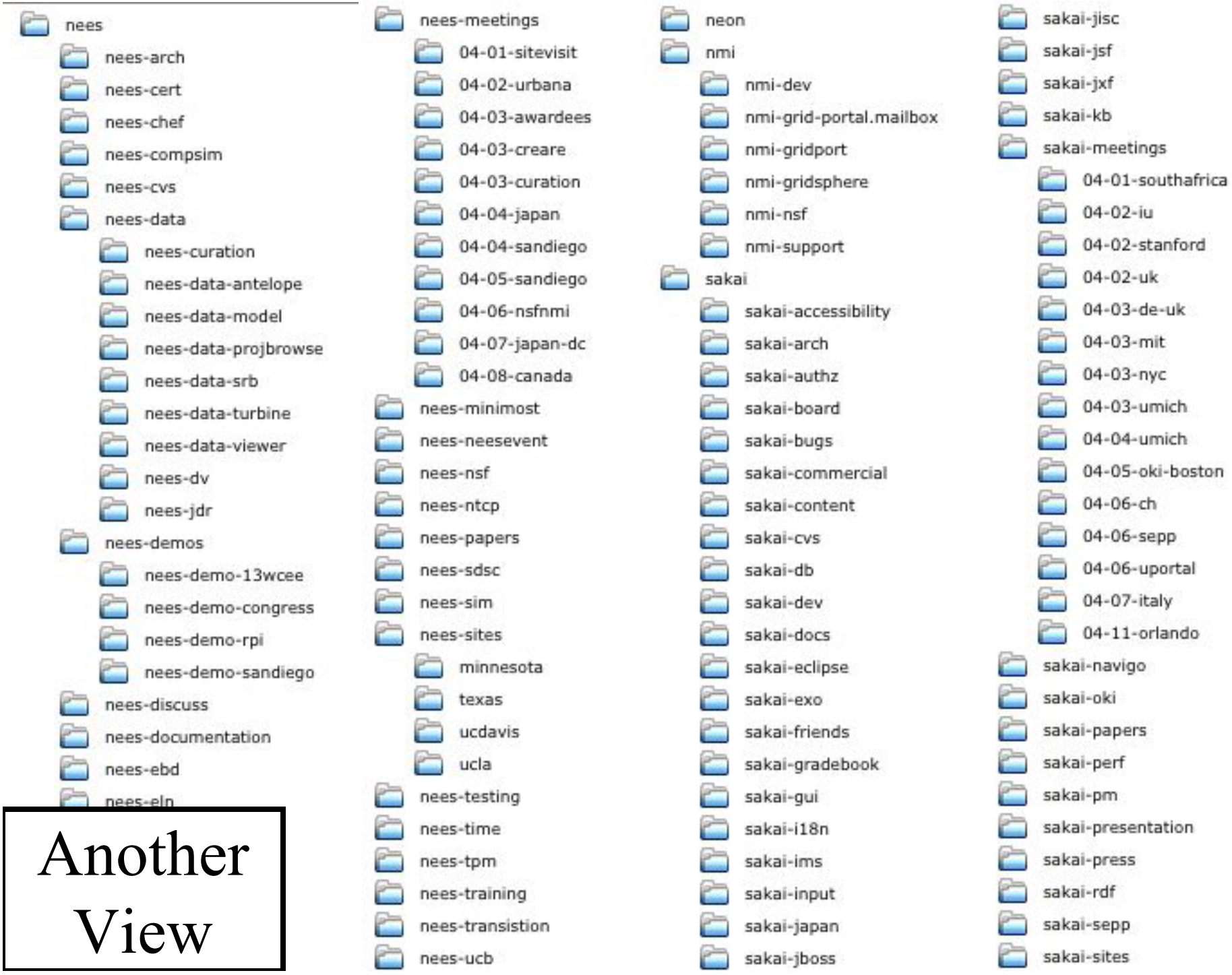
Collaboration Happens

- As individuals, we are parts of many groups and have many roles in those groups

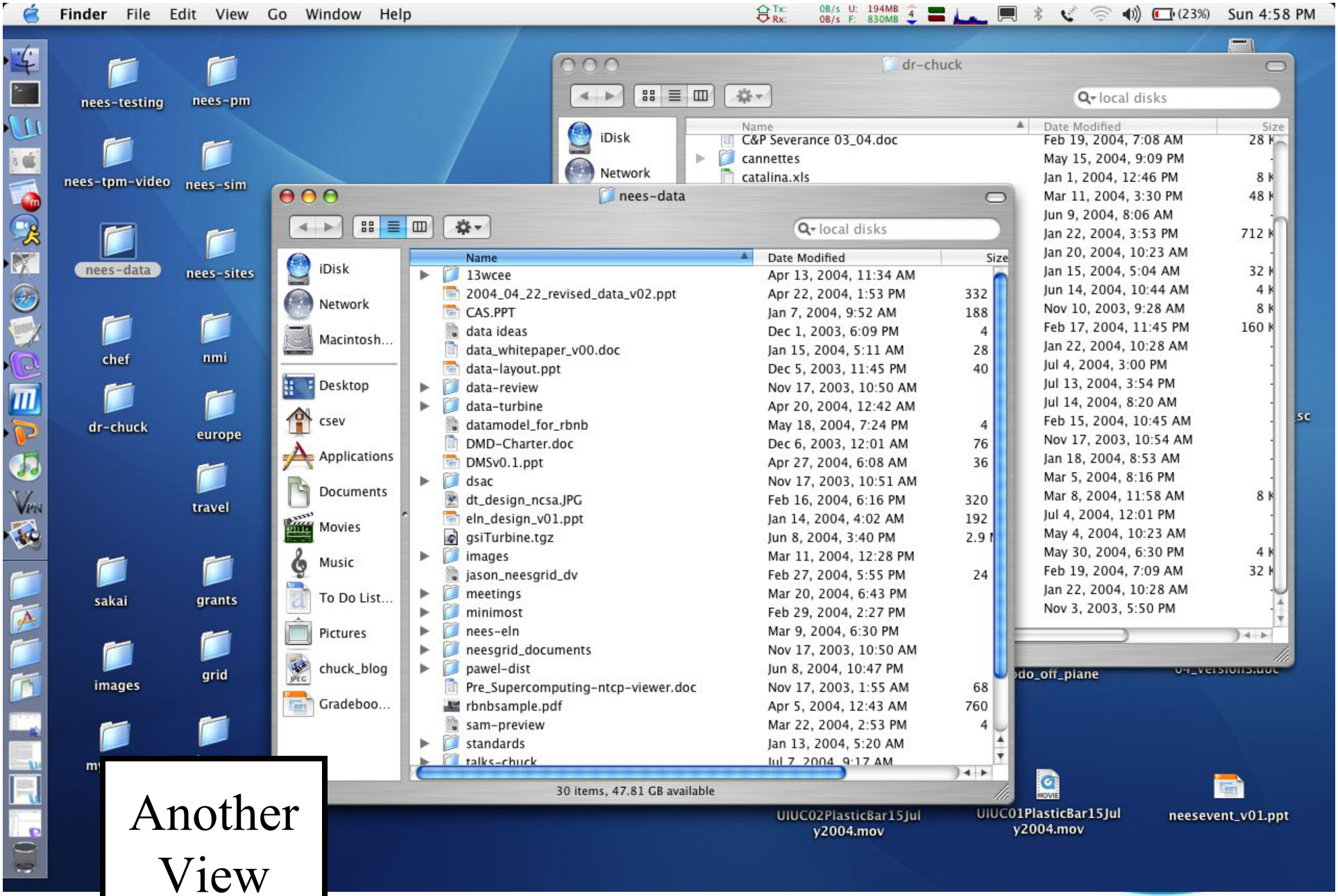
Sakai



One View of Chuck's Context Map



Another View



Another View

Maintaining the Map

- Read E-Mail and move to proper folders
- Copy attachments into folders
- Searching for information
- Making calendar entries from E-Mail

Sakai

Imagine Software

- That could create a new “context” in a few clicks
  - Enroll/invite others to the context as necessary in a few more clicks
- Context capabilities
  - E-Mail list (automatically extracts attachments and places them in folders which appear on your desktop)
  - Schedule (you can either see a “federated” schedule across all contexts or look at one context)
  - Persistent browser-based chat - quite useful during meetings when the Polycom or VRVS messes up :)
  - Resource area where anyone can upload files which appear on everyone’s desktop at the same time (WebDAV)
  - Threaded discussion area for the context
- Problem: There are literally hundreds of solutions to portions of this problem.

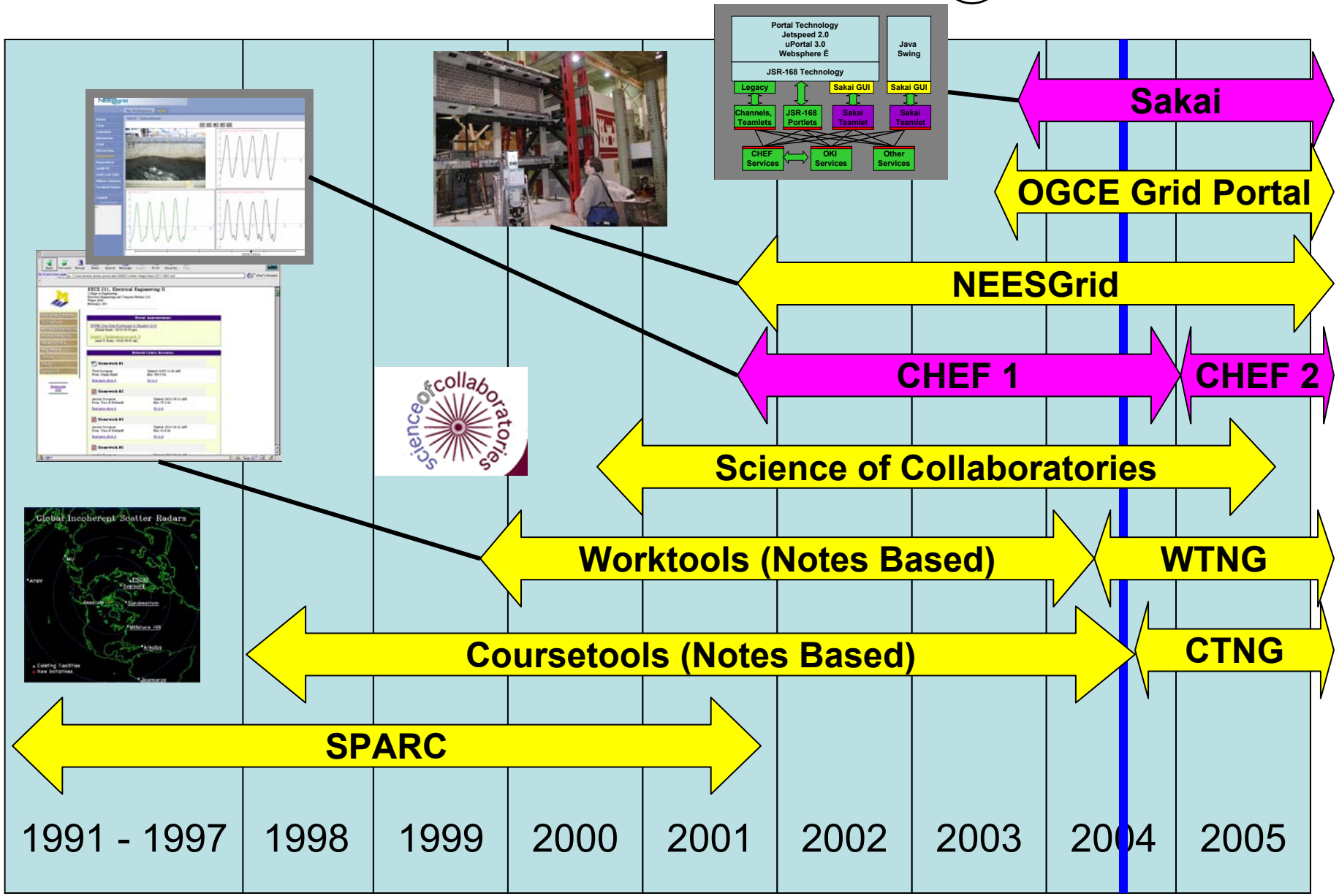
Sakai

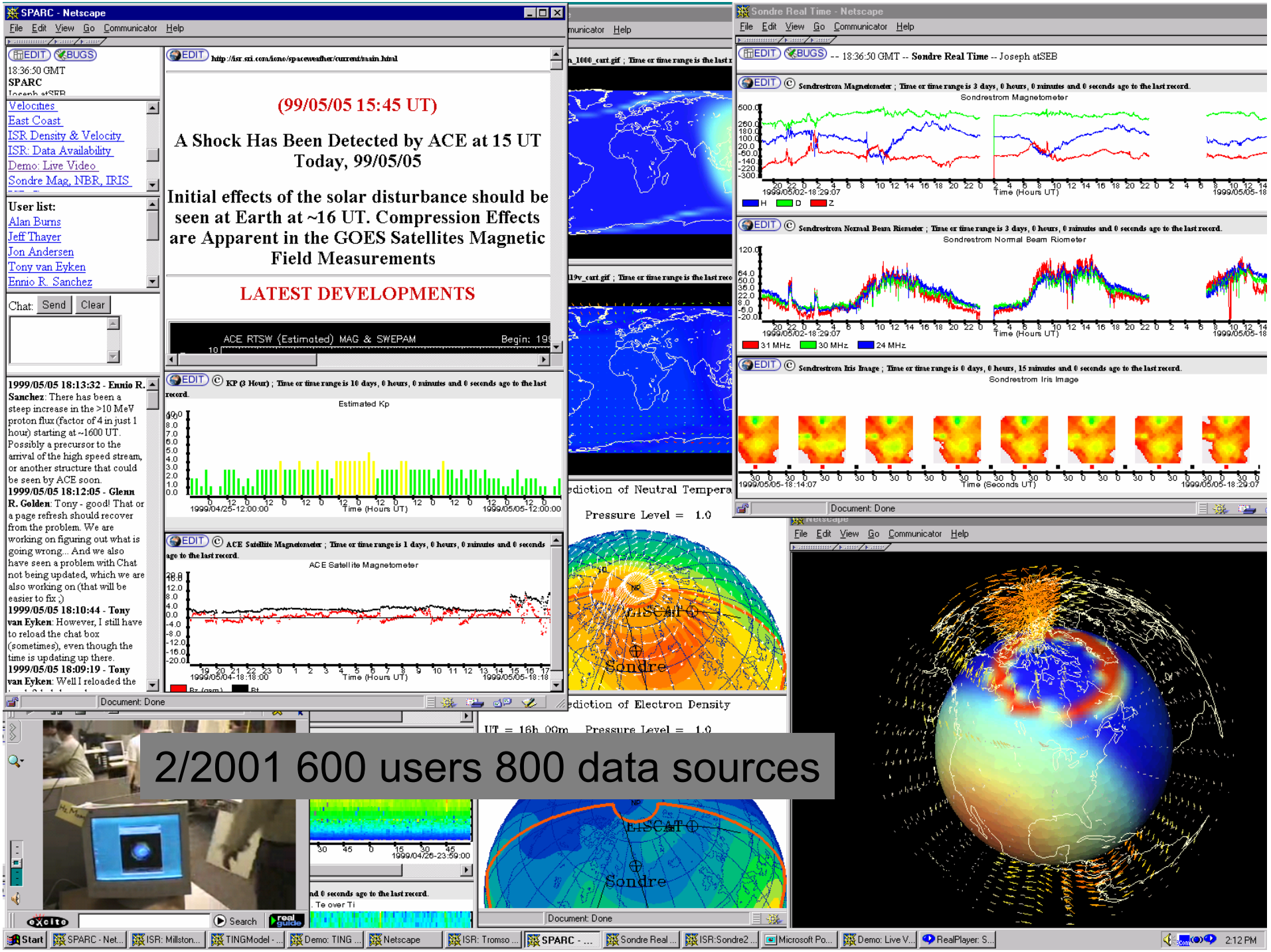
More Software

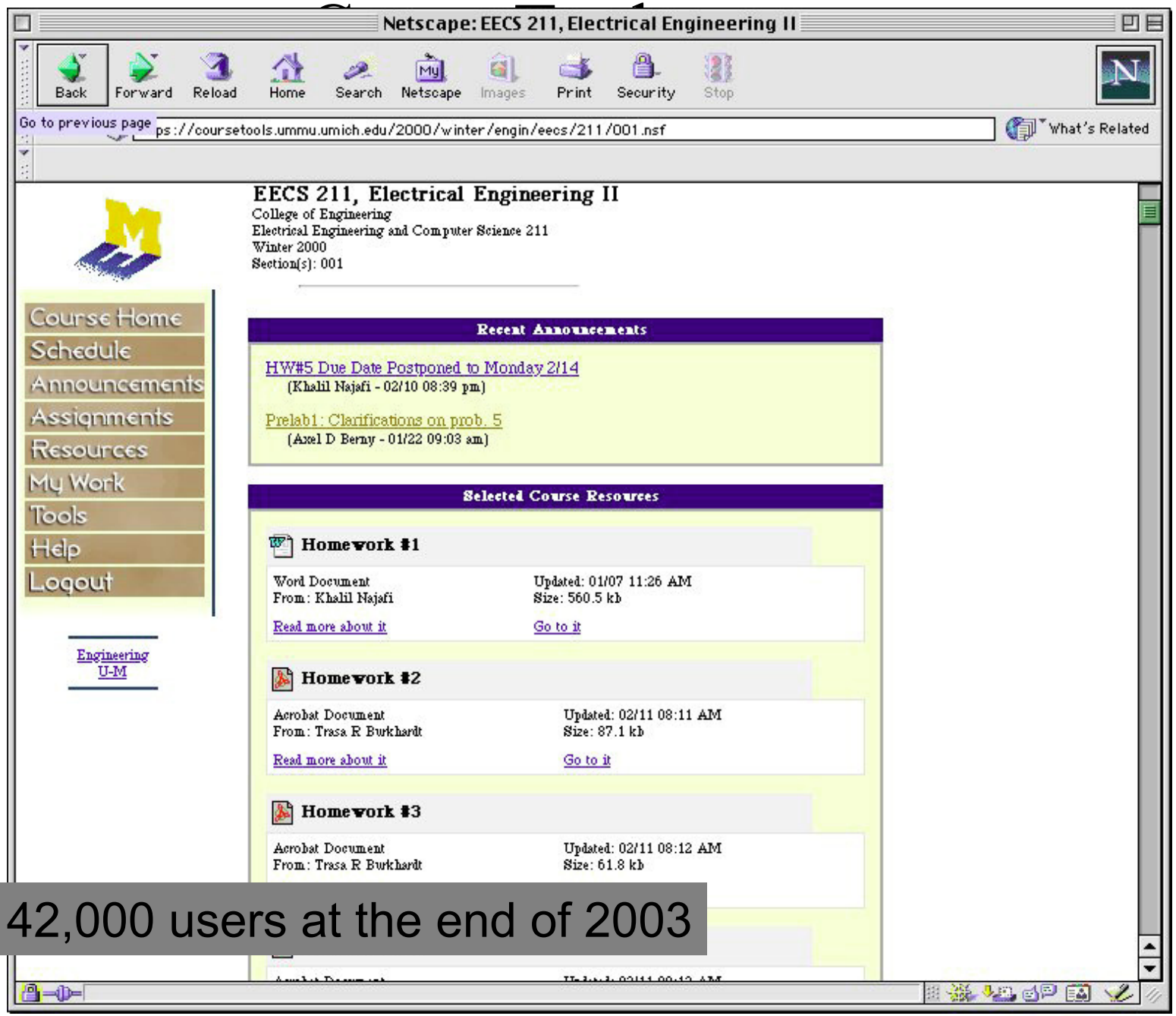
- A single place to see new activity in your “contexts”
- These contexts are stored on backed-up production servers rather than your desktop for many years
- A search across your contexts - that would be really cool
- The ability to customize each context in terms of look, feel, and capabilities
- The ability to build unique domain specific tools and interfaces to extend the mechanism using Portlets, Servlets, or Applets

Sakai

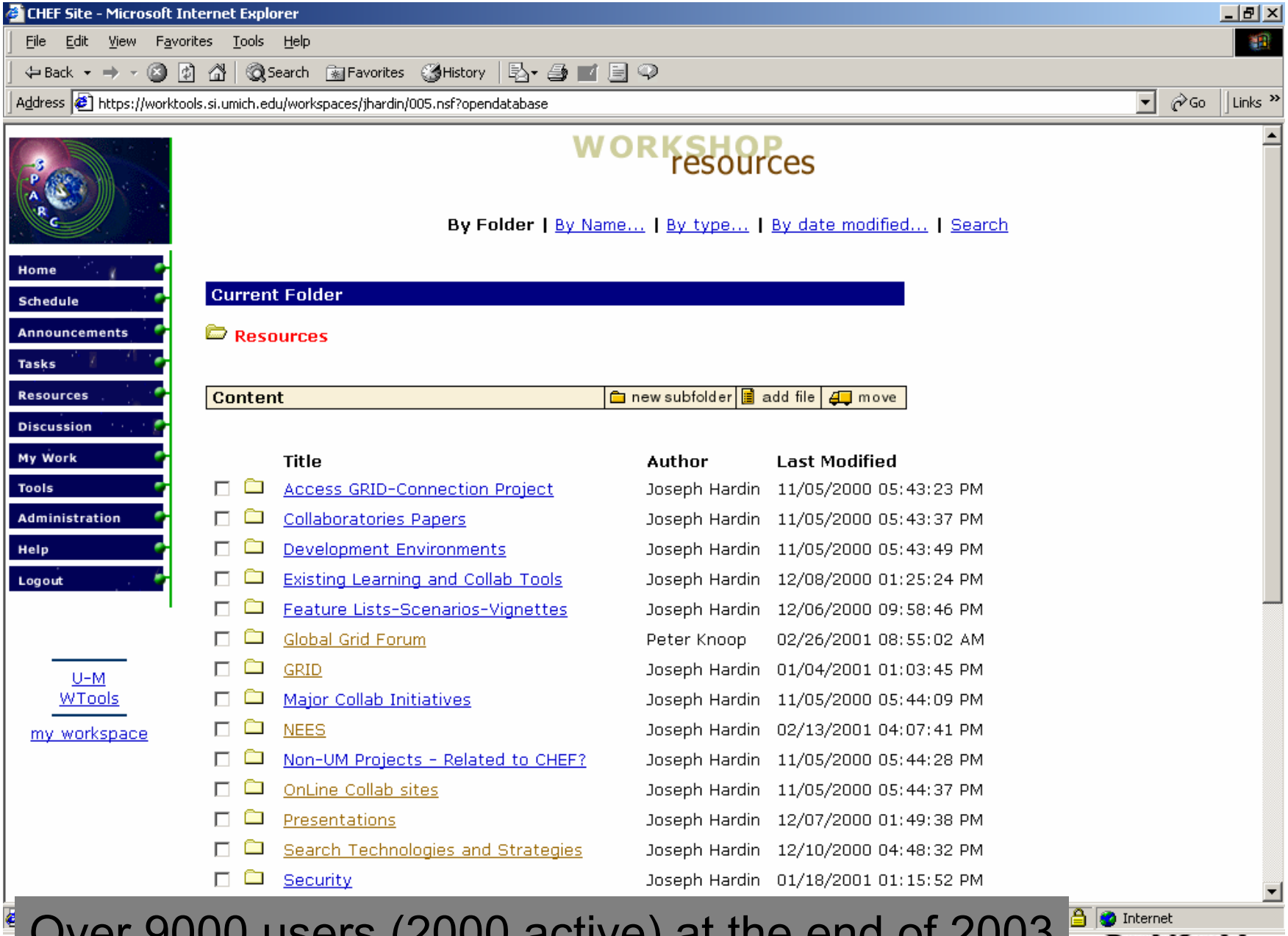
A 10-Year Collaborative Mission @ UM




SPARC - Netscape
(99/05/05 15:45 UT)
A Shock Has Been Detected by ACE at 15 UT Today, 99/05/05
Initial effects of the solar disturbance should be seen at Earth at ~16 UT. Compression Effects are Apparent in the GOES Satellites Magnetic Field Measurements
LATEST DEVELOPMENTS
Sondre Real Time - Netscape
Sondrestrom Magnetometer
Sondrestrom Normal Beam Riometer
Sondrestrom Iris Image
Estimated Kp
ACE Satellite Magnetometer
Pressure Level = 1.0
EISCAT
Sondre
2/2001 600 users 800 data sources



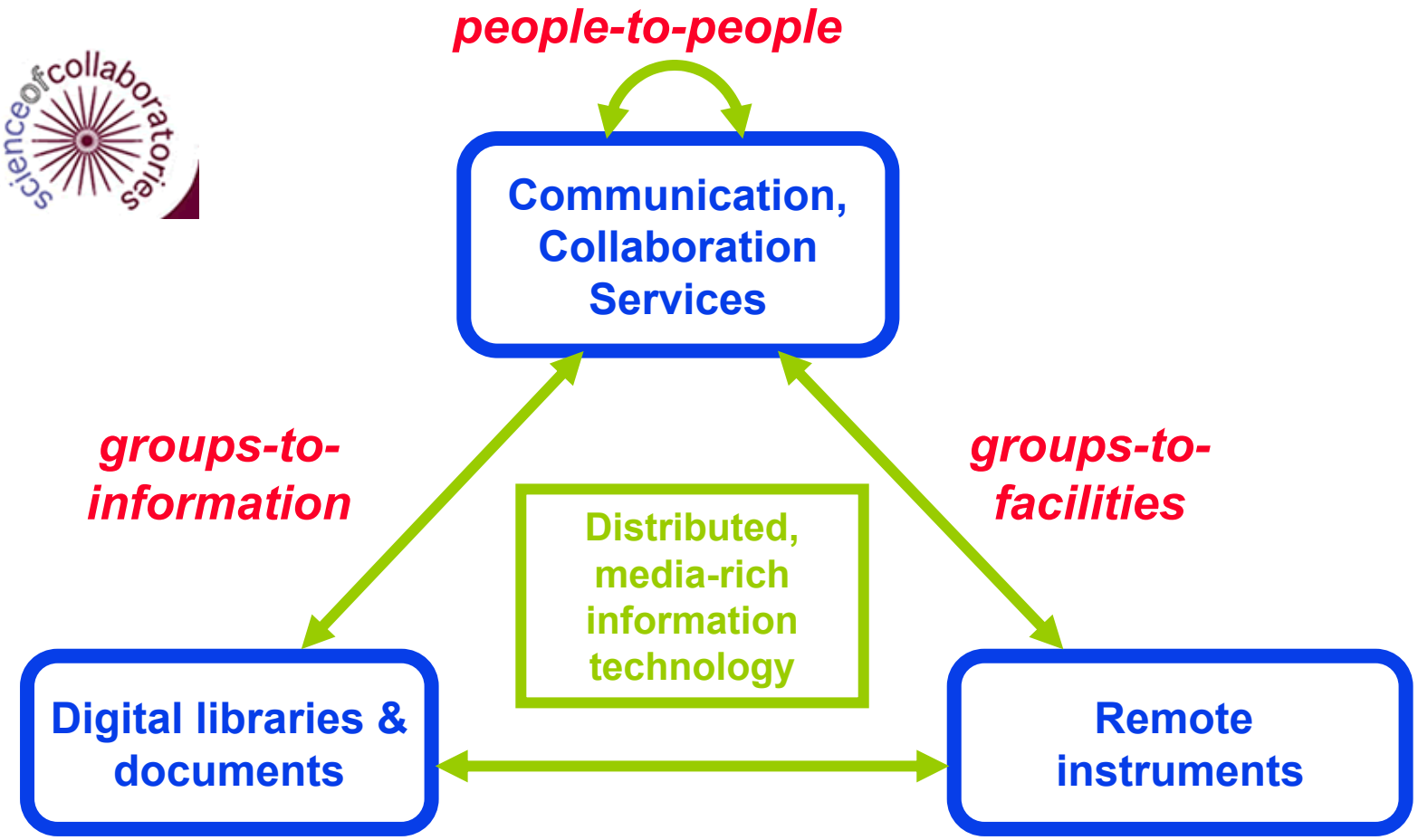
Over 42,000 users at the end of 2003



Over 9000 users (2000 active) at the end of 2003

Sakai

Science of Collaboratories



http://www.scienceofcollaboratories.org/

NSF Funded ITR

CHEF 1.0

- Fall 2001: CHEF Development begins
  - Generalized extensible framework for building collaboratories
  - “Best-of” CourseTools, SPARC, WorkTools
- Integrate across current UM projects and adopt relevant standards
- Funded internally at UM as replacement for CourseTools
- All JAVA - Open Source
  - Jakarta Jetspeed Portal
  - Jakarta Tomcat Servlet Container
  - Jakarta Turbine Service Container
- Build community of developers through workshops and outreach

Sakai

Not “just” a portal

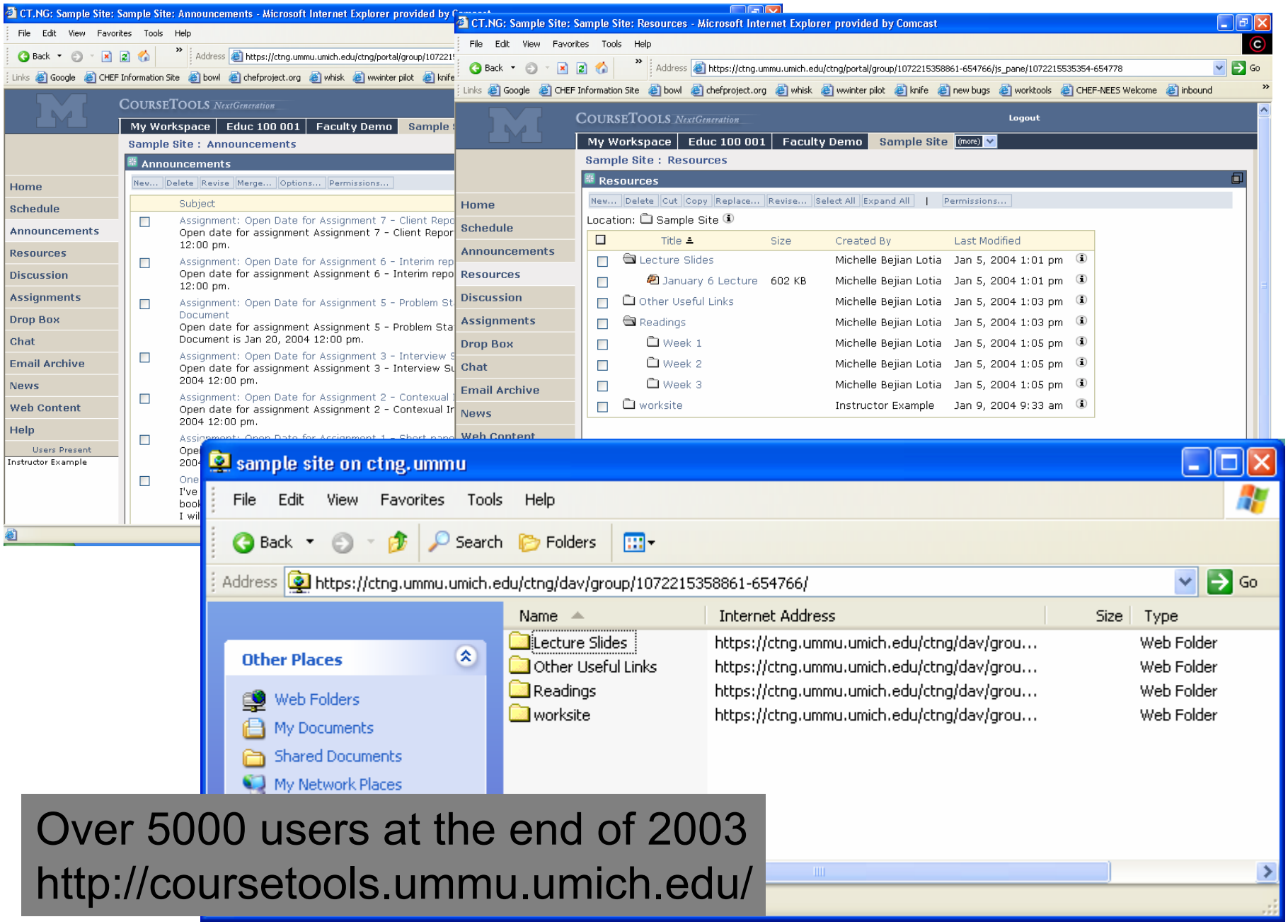
- Portals are a framework to deploy tools (aka rectangles) and focus on how the user wants to arrange their own “rectangles”
- While CHEF technically is a portal, the goal is for the tools to work together closely and seem to really be parts of a larger “tool”
- CHEF has a lot of features, (services, presence, notification, etc..) which bridge the gap between portal and application framework

Sakai

CHEF Applications

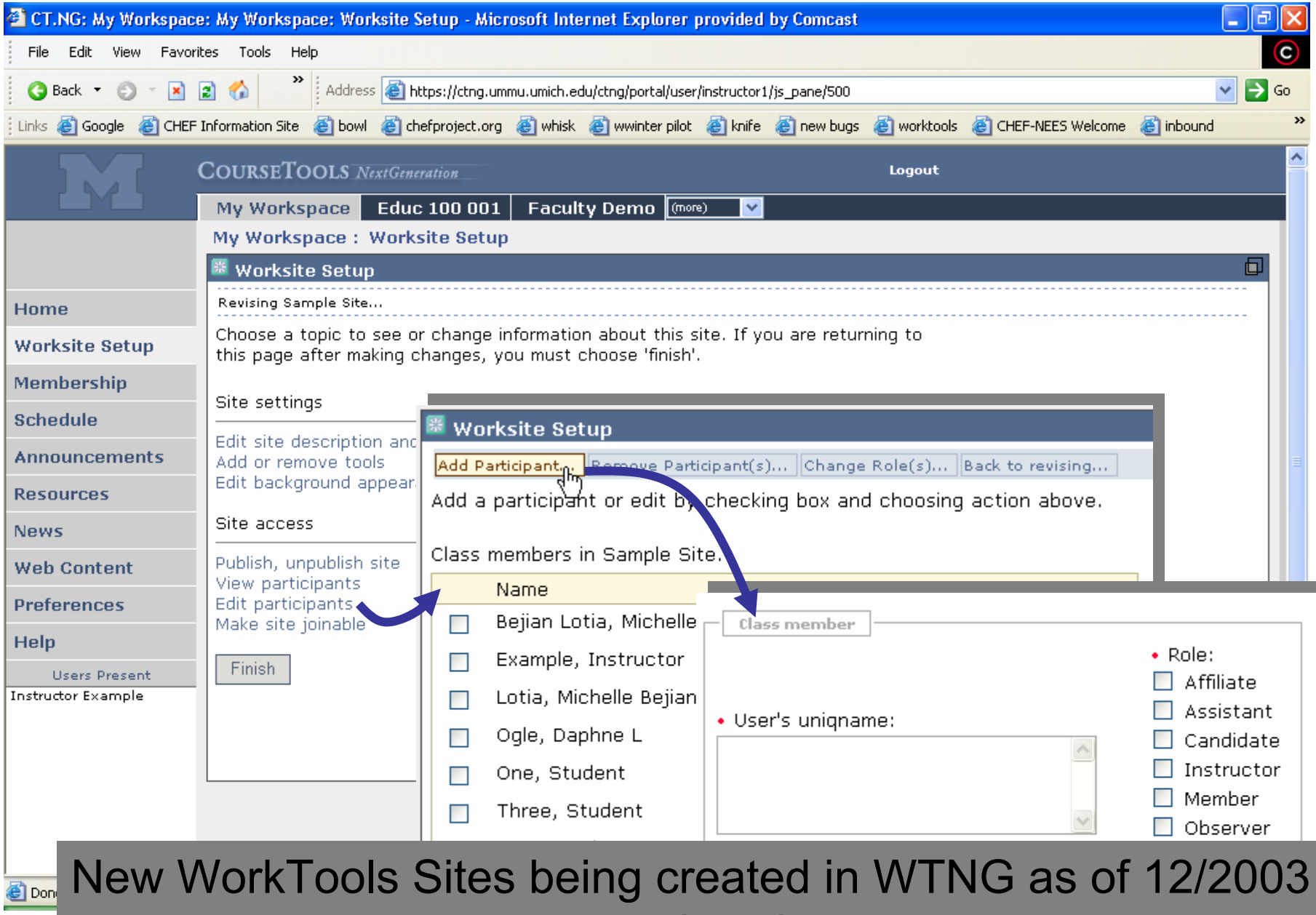
- CourseTools Next Generation
- WorkTools Next Generation
- NEESGrid
- NSF National Middleware Grid Portal

Sakai



Over 5000 users at the end of 2003

http://coursetools.ummu.umich.edu/



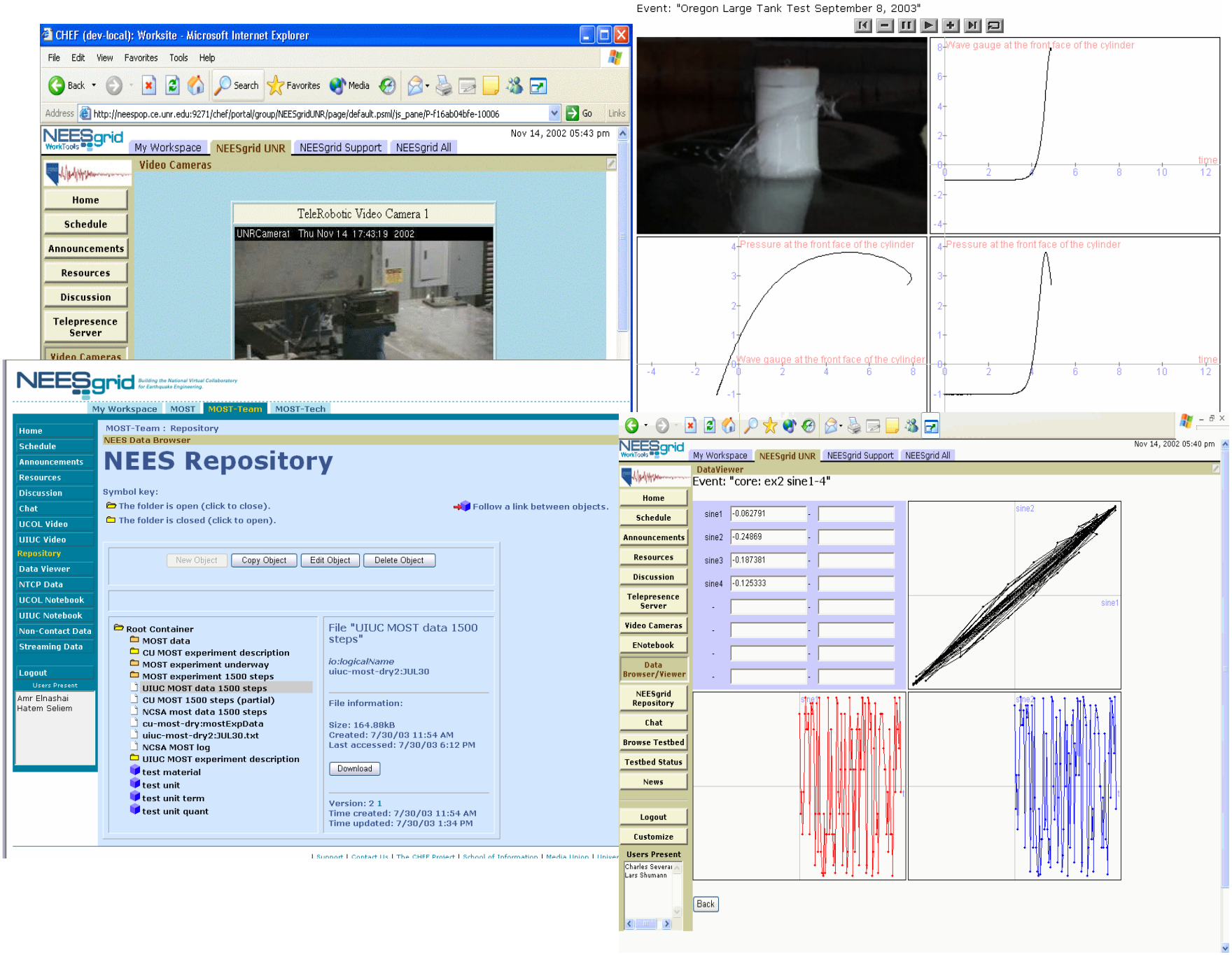
New WorkTools Sites being created in WTNG as of 12/2003
Run on the same servers as CTNG.

NEESgrid

Network for Earthquake Engineering Simulation

NSF Funded.
NCSA, ANL, USC/ISI, UM, USC, Berkeley, MSU

Sakai


Event: "Oregon Large Tank Test September 8, 2003"
CHEF (dev-local): Worksite - Microsoft Internet Explorer
Wave gauge at the front face of the cylinder
Pressure at the front face of the cylinder
Pressure at the front face of the cylinder
Wave gauge at the front face of the cylinder
NEESgrid
My Workspace
NEESgrid UNR
NEESgrid Support
NEESgrid All
Nov 14, 2002 05:43 pm
Video Cameras
Home
Schedule
Announcements
Resources
Discussion
Telepresence Server
Video Cameras
TeleRobotic Video Camera 1
UNRCamera1 Thu Nov 14 17:43:19 2002
MOST-Team : Repository
NEES Data Browser
NEES Repository
Symbol key:
The folder is open (click to close).
The folder is closed (click to open).
Follow a link between objects.
New Object
Copy Object
Edit Object
Delete Object
Root Container
MOST data
CU MOST experiment description
MOST experiment underway
MOST experiment 1500 steps
UIUC MOST data 1500 steps
CU MOST 1500 steps (partial)
NCSA most data 1500 steps
cu-most-dry:mostExpData
uiuc-most-dry2:JUL30.txt
NCSA MOST log
UIUC MOST experiment description
test material
test unit
test unit term
test unit quant
File "UIUC MOST data 1500 steps"
io:logicalName
uiuc-most-dry2:JUL30
File information:
Size: 164.88kB
Created: 7/30/03 11:54 AM
Last accessed: 7/30/03 6:12 PM
Download
Version: 2 1
Time created: 7/30/03 11:54 AM
Time updated: 7/30/03 1:34 PM
Nov 14, 2002 05:40 pm
DataViewer
Event: "core: ex2 sine1-4"
sine1 -0.062791
sine2 -0.24869
sine3 -0.187381
sine4 -0.125333
Back
Logout
Customize
Users Present
Amr Elnashai
Hatem Seliem
Charles Severance
Lars Shumann

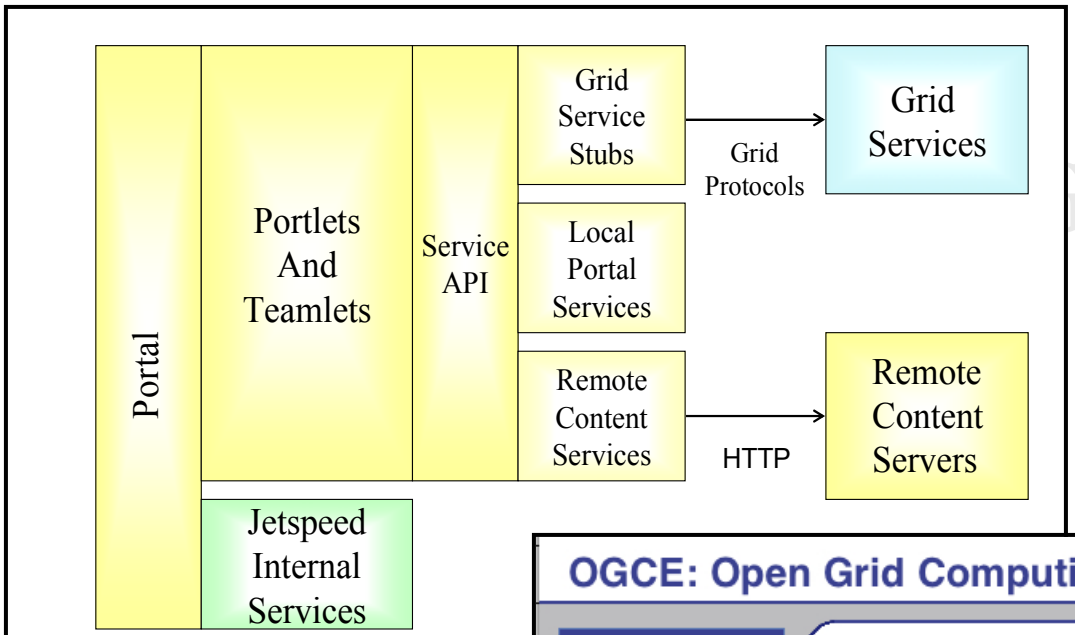
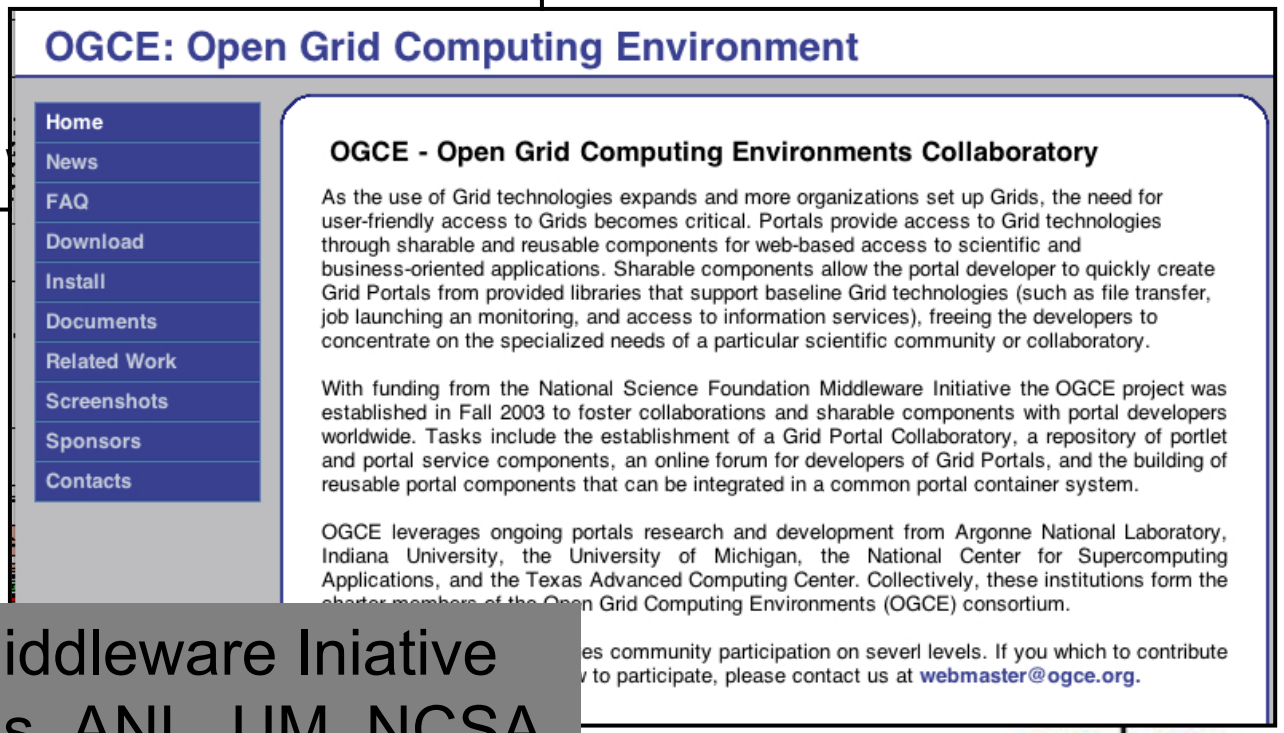


Figure 4: The revised portal architecture portal services.

www.ogce.org



NSF National Middleware Iniative
Indiana, UTexas, ANL, UM, NCSA

What we learned in 10 years.

- Portal technology is a good idea - forces component approach - functionality does not “smear”
- Portals are not just aggregators of independent information - but can be an application framework
- Many (but not all) tools can be used for both teaching and learning and research collaboration
- Separating functionality into lightweight GUI components and pluggable services with strong and well-specified APIs allows significant reusability
- GUI elements program to abstract service interfaces - not databases, file systems, LDAP, etc. - this allows great flexibility.

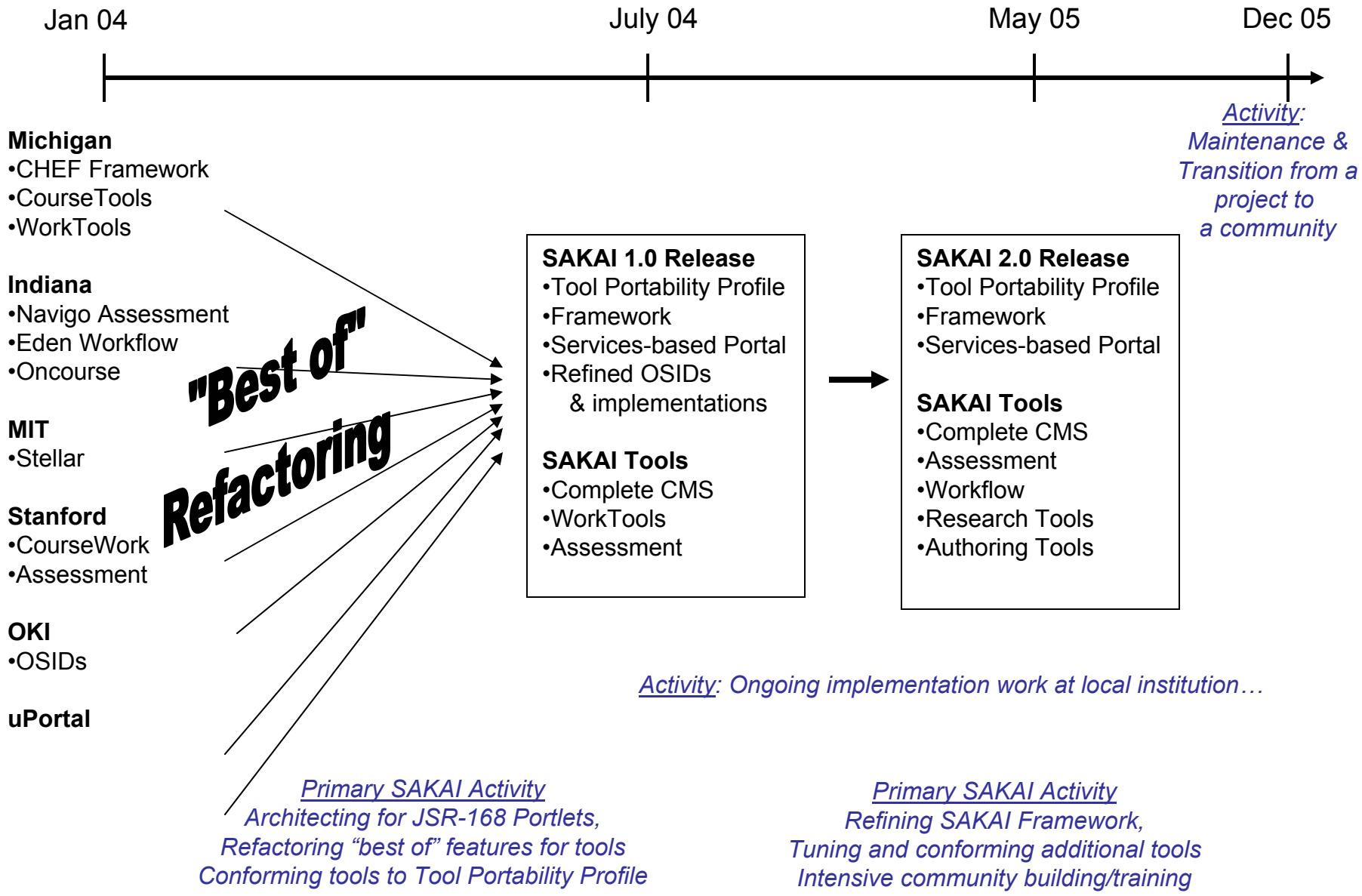
Sakai

While we were building collaboratories…

- The Open Knowledge Initiative (OKI) at MIT was developing APIs for learning management systems - involving many universities (UM, Indiana, Stanford, and MIT were strong participants)
- Indiana, Stanford, MIT all developed learning management system
- Java Community Process (JCP) produced JSR-168 - The "unified" portal standard API
- Oasis developed the Web Services for Remote Portals (WSRP) standard
- The open-source uPortal portal project had quietly moved into the #1 open source portal (#4 including commercial vendors)

Sakai

So we got together and drew an über collaboration picture…



THE CHRONICLE OF HIGHER EDUCATION

Today's News

Thursday, January 22, 2004

4 Colleges Collaborate on Open-Source Courseware

By ANDREA L. FOSTER

In what may be a big threat to commercial providers of course-management systems, four universities have announced a $6.8-million collaborative venture to create open-source courseware tools and related software for higher-education institutions.

The universities developing the system, called the Sakai Project, are the Indiana University system, the Massachusetts Institute of Technology, Stanford University, and the University of Michigan at Ann Arbor, which will lead the effort.

HEADLINES

Bush proposes $250-million job-training program in visits to community colleges

U. of Pennsylvania to name Amy Gutmann of Princeton as its new president

Record companies sue 532 computer users, accusing them of illegally sharing songs

4 colleges collaborate on open-source courseware

境

Sakai Core Members

- Universities
  - Indiana
  - Michigan
  - MIT
  - Stanford
- Projects
  - Open Knowledge Initiative (OKI)
  - uPortal - JaSIG
- Funding ($6.8M - 2 Years)
  - Mellon Foundation
  - Hewlett Foundation
  - Partners Program
  - Core member match

Sakai

What we agreed to build…

- A Collaborative Learning Environment
  - Open Source
  - Uses OKI (Open Knowledge APIs)
  - Uses uPortal as its portal framework
- Similar to
  - Blackboard
  - WebCT
- And all four core institutions would deploy the commonly developed software

Sakai

Sakai 1.0

- Site based collaboration environment
  - Worksite management
  - E-Mail Lists
  - Threaded Discussion
  - Resources (folders) with WebDav support
  - Chat
  - No search yet :(
  - Many other tools
- Beta Release July 15, 2004
- Production site available at ctools.umich.edu

More Sakai Beta Tools

Admin: Alias Editor (chef.aliases)
Admin: Archive Tool (chef.archive)
Admin: Memory / Cache Tool (chef.memory)
Admin: On-Line (chef.presence)
Admin: Realms Editor (chef.realms)
Admin: Sites Editor (chef.sites)
Admin: User Editor (chef.users)
Announcements (chef.announcements)
Assignments (chef.assignment)
C. R. U. D. (sakai.crud)
Chat Room (chef.chat)
Discussion (chef.discussion)
Discussion (chef.threadeddiscussion)
Dissertation Checklist (chef.dissertation)
Dissertation Upload (chef.dissertation.upload)
Drop Box (chef.dropbox)
Email Archive (chef.mailbox)
Help (chef.contactSupport)
Membership (chef.membership)
Message Of The Day (chef.motd)
My Profile Editor (chef.singleuser)
News (chef.news)
Preferences (chef.noti.prefs)
Recent Announcements (chef.synoptic.announcement)
Recent Chat Messages (chef.synoptic.chat)
Recent Discussion Items (chef.synoptic.discussion)
Resources (chef.resources)
Sample (sakai.module)
Schedule (chef.schedule)
Site Browser (chef.sitebrowser)
Site Info (chef.siteinfo)
Web Content (chef.iframe)
Worksite Setup (chef.sitesetup)
WebDAV

Sakai Going Forward

- Focus on the"Learning" of Collaborative Learning Environment through 2Q05
  - Getting ready for production deployment at the four partner sites
  - Improving the look and feel of the software
  - Many feature enhancements (to satisfy four + 60 schools)
  - New GUI Programming Environment based on Java Server Faces
  - Building new set of Sakai APIs (Java)
    - Based on OKI - Enabling RDF
- Move into OGCE and NEESGrid starting 3Q04
- Release 2.0 - 2Q04

Sakai

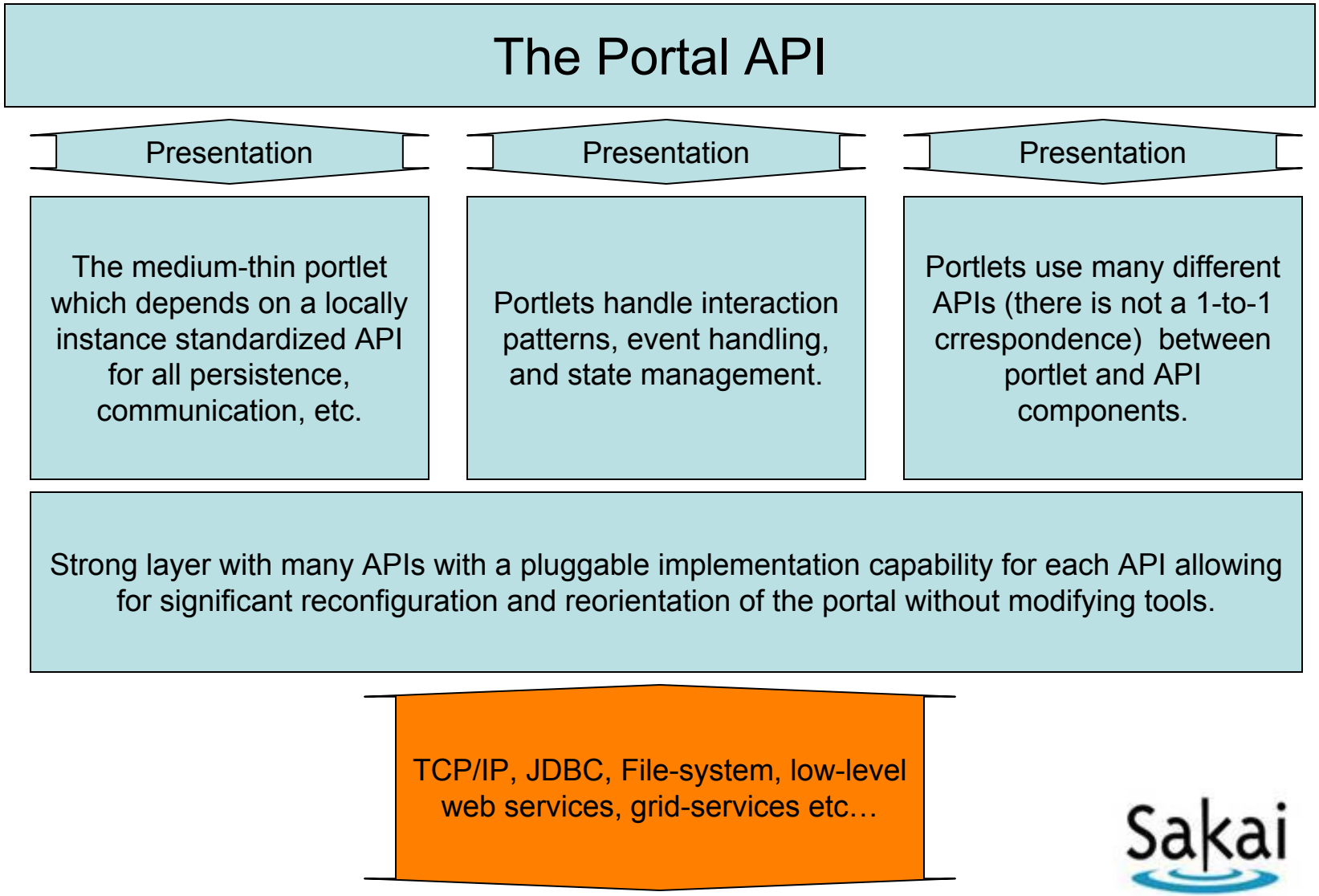
Sakai Architecture

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

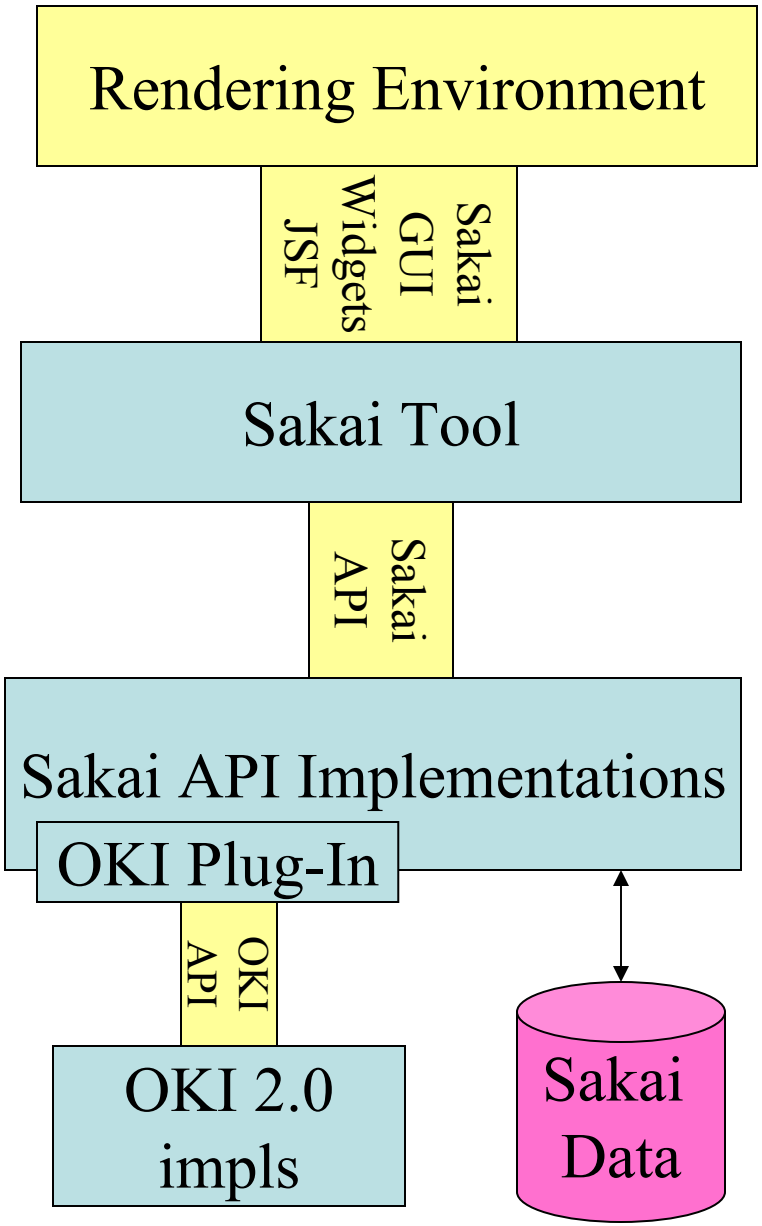
Sakai

Portlet Design Patterns - Where does the information come from / go to?



The Sakai Tool Portability Profile Framework

The Sakai API is based heavily on the OKI API but focused on the portability and interoperability of Sakai tools. The Sakai API should be though of as value add on top of the OKI APIs. The Sakai APIs encode what OKI would call “out-of-band” agreements explicitly into method calls, parameters and return values.



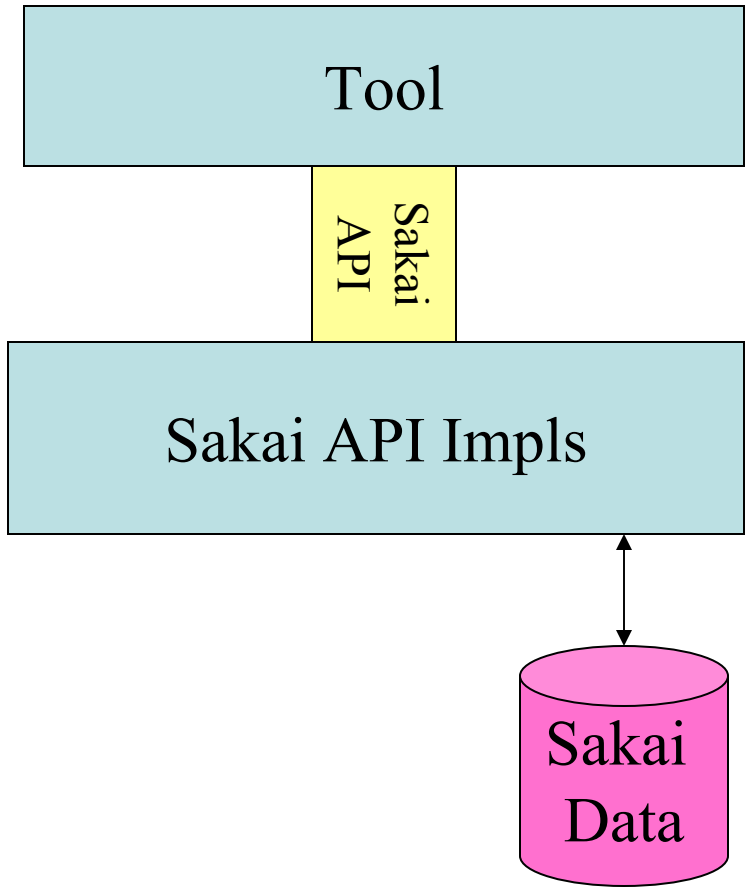
Sakai Application Programming Interfaces (APIs)

- Make tool development easier
- Promote data and tool portability between Sakai environments
- Hide some data management details
- Error handling
- Provide re-usable system and application services to tool developers

Sakai

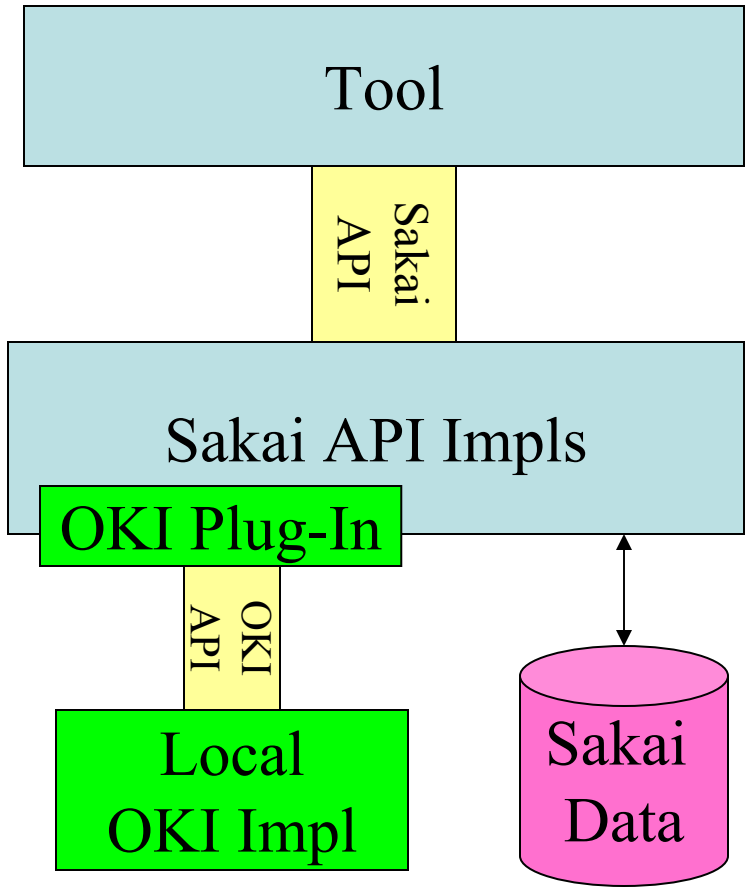
Simple Deployment

In a simple deployment, the Sakai system may just use the Sakai provided API implementations and point it at a database connection, let Sakai build the tables - and off we go.



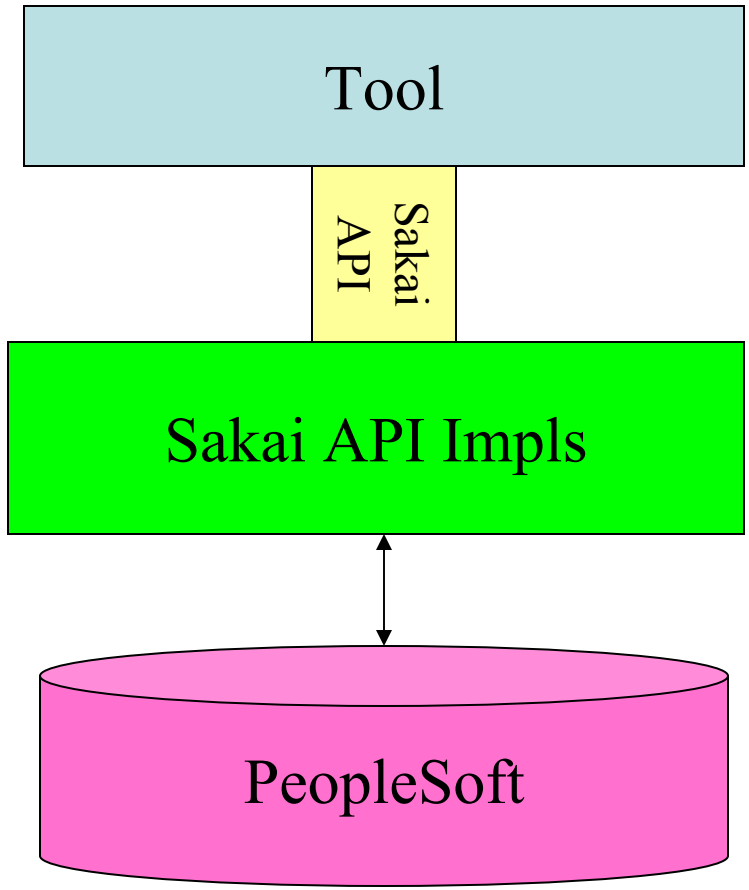
Basic
Local Customization

In the most common situation, local sites will want to customize a few things - perhaps AUTHN, AUTHZ, a few DR's. Sites will write/procure/configure OKI implementations which plug into the Sakai implementations. The Sakai implementations are configured to "federate" between data from the plug-in and the Sakai data as desired by the site.

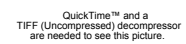


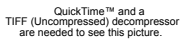
Drastic Local Customization

At some level, it would be possible to completely re-implement the entire Sakai API for the particular component (i.e. grading). Because the Sakai APIs have no "out-of-band" agreements, the tools cannot perceive that the implementation has been changes.



Concepts and Terminology





Sakai

Sakai Technology

- Hibernate for object persistence
- Sakai APIs that imitate OKI OSIDs
- Tools are based on APIs and servlets
- JavaServer Faces separate out the presentation from tool logic
- uPortal integrates tools at the UI level

Sakai

Specific TPP Elements

- GUI: Java Server Faces + Sakai Widgets
- Framework API
  - Best practice: Setter-style dependency injection
    - Both tools and services are components*
    - Cross-webapp service framework
  - Service locator also supported
- No (zip, zero, nada) framework imports required

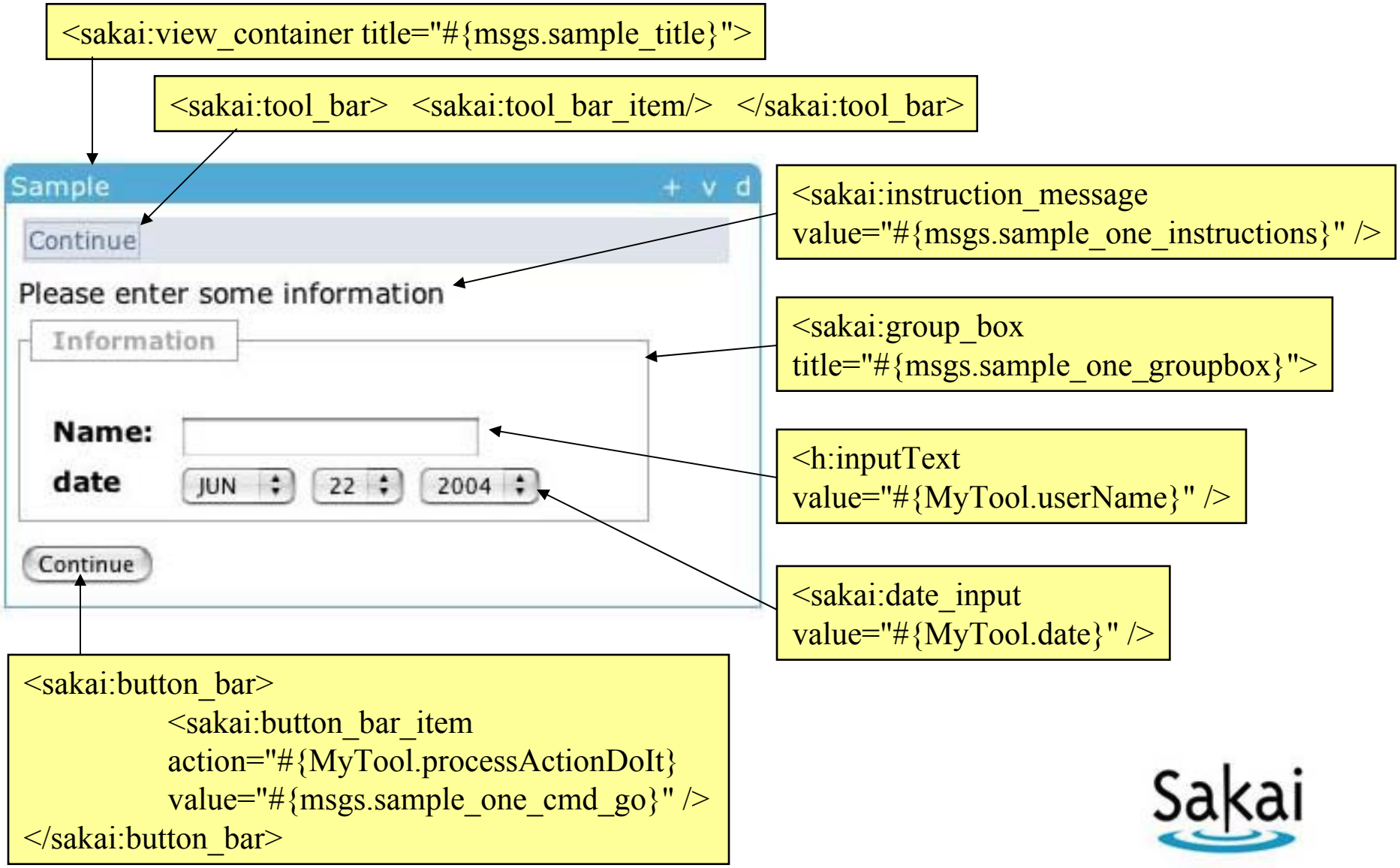
Sakai

JSF Mini Tutorial

- Document-based layout which relates a view stored in a set of beans using a set of widgets (button, input, drop-down) and a set of action methods associated with buttons, etc.
- There are no URLs (munged or otherwise)
- Additional Sakai widgets within JSF in insure look and feel at a higher level across tools

```
<sakai:tool_bar>
  <sakai:tool_bar_item
     action="#{AnnouncementTool.processActionListNew}"
     value="#{msgs.annc_list_new}" />
  <sakai:tool_bar_item
     action="#{AnnouncementTool.processActionListDelete}"
     value="#{msgs.annc_list_delete}" />
…
```

JSF is Used to Describe the UI



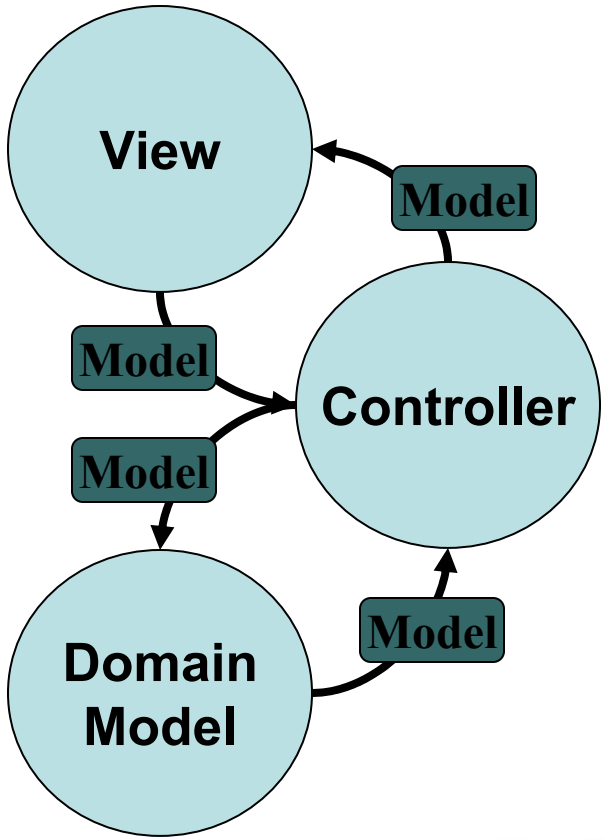
Model View Controller Mini Tutorial

- Domain Model
  - Long Term Persistence
- Controller
  - Orchestra conductor
  - Retrieves the Model (subset of Domain Model)
  - Holds state in terms of the user interaction
  - Decorates the Model with view specific information
  - Selects Layout
  - Hands layout and Decorated-Model to View
  - Processes the returned Decorated-Model from View
- View
  - Renders Decorated-Model to user using layout
  - Handles user interaction (possibly with validation)
  - Returned Modified-Decorated-Model to Controller

Sakai

MVC Mini Tutorial

- Domain Model
  - Persistence
- View
  - Renders decorated model
- Controller
  - Orchestrates
- Model
  - Data moved between elements



Sakai

Inversion of Control Mini Tutorial

- Components code to interfaces, not implementations
- When a component needs an implementation for a particular interface, how does it find the implementation
- Four basic approaches (formerly known as Level 0-3 IoC)
  - Service Locator (Turbine, Avalon)
  - Interface Injection (Avalon)
  - Setter Injection (Spring)
  - Constructor Injection (Pico)

http://www.martinfowler.com/articles/injection.html

Sakai

Service Locator

- Component calls a service locator with the desired interface as a parameter
  - Can gather dependencies dynamically this is useful if dependency lookup is expensive
  - This does force an explicit dependency on the framework

```
class MovieLister...
    MovieFinder finder =
        (MovieFinder) ServiceLocator.getService("MovieFinder");
```

Sakai

Interface Injection

- The component declares that it “implements serviceable” which triggers the framework to poke in the implementations via well-known methods (Avalon)
  - Often this is used to inject a service locator and then that service locator is used to garner other dependencies

Sakai

Setter Injection (Sakai preferred)

- The component simply provides bean-style setter and getter methods for the dependent interfaces it needs - framework constructs object then calls the setter for anything that is already in the framework (auto-wire)
  - No explicit dependency on the framework at all
  - Articulates well with all of the bean support in lots of places
  - Not able to declare which of the setters are required for safe operation unless this is in a XML config file

Sakai

Constructor Injection

- The component provides a number of constructors with the dependencies as parameters - the framework picks one and constructs the object will all of its needed dependencies.
  - The object is never in a “partially ready to go” state
  - Can express multiple acceptable dependency sets using different construtors
  - No explicit dependencies on the framework
  - Cannot work unless the framework is doing the constructing (I.e. no chance to “fake it” in Servlet of JSF managed objects)

Sakai

IoC Summary

- Setter injection is the best practice for new code
  - No explicit dependencies
  - Leverages bean support
  - Can be “simulated” easily when framework is not constructing objects
- We will always support Service Locator

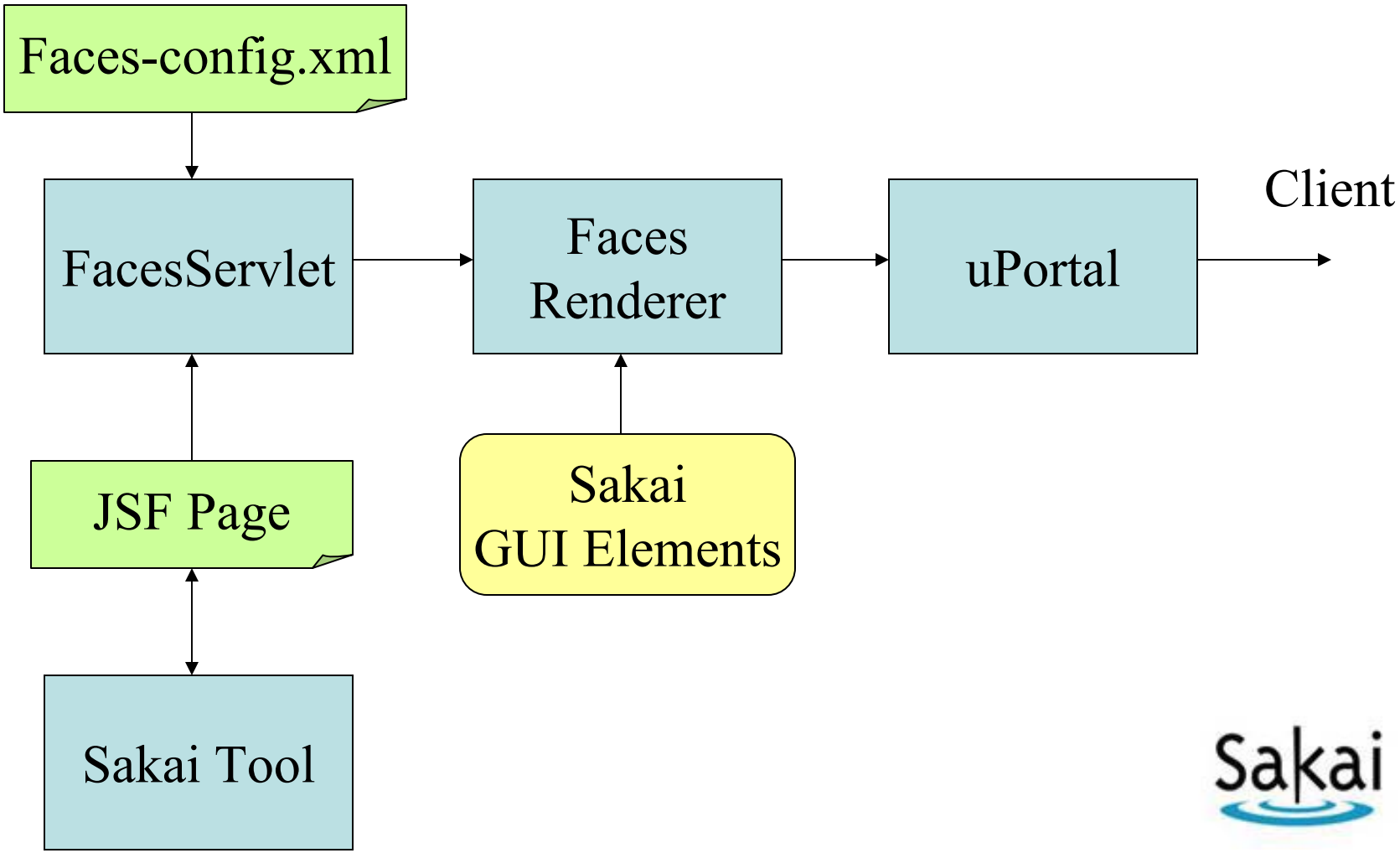
Sakai

Sakai Presentation

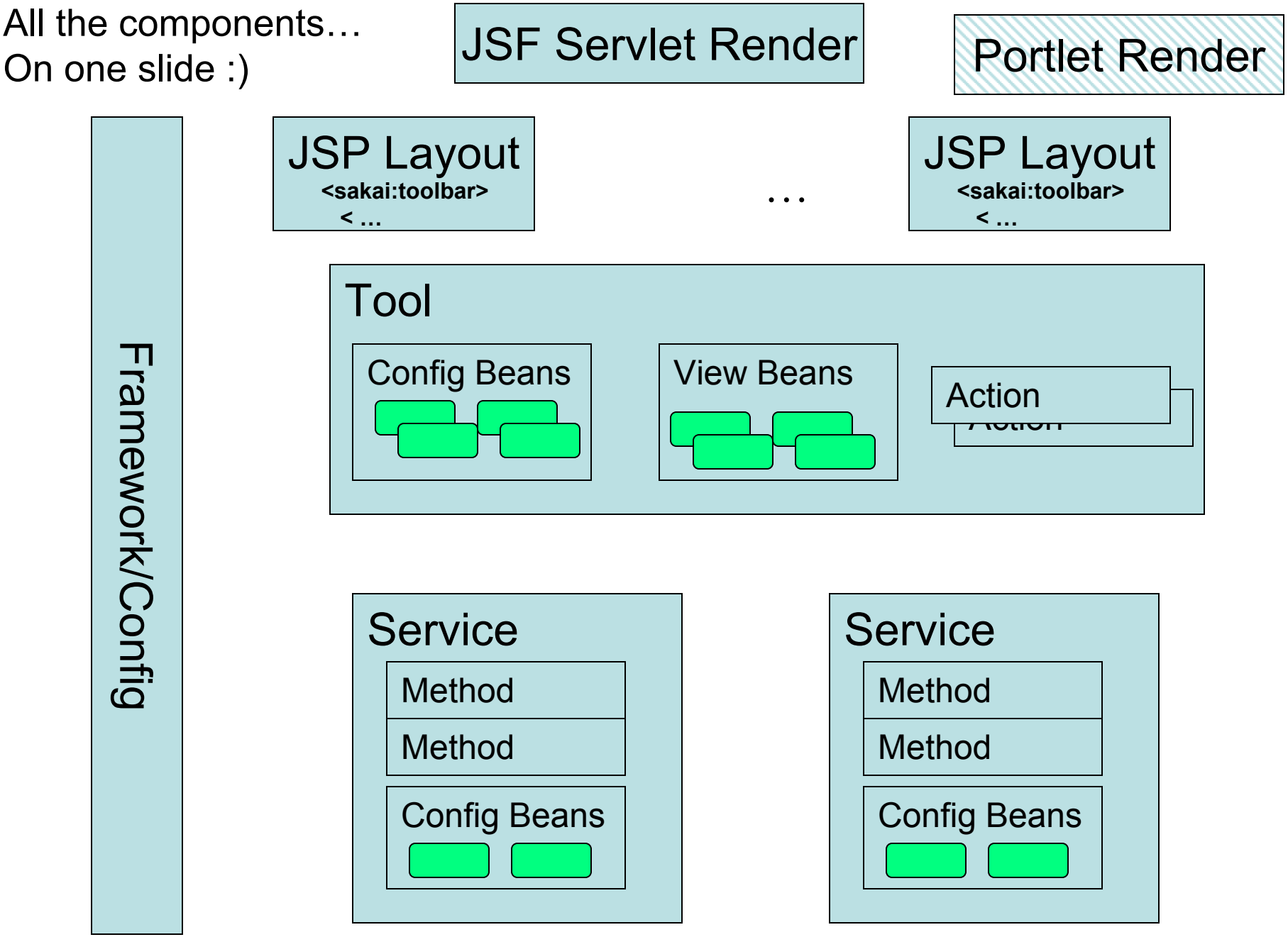
QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

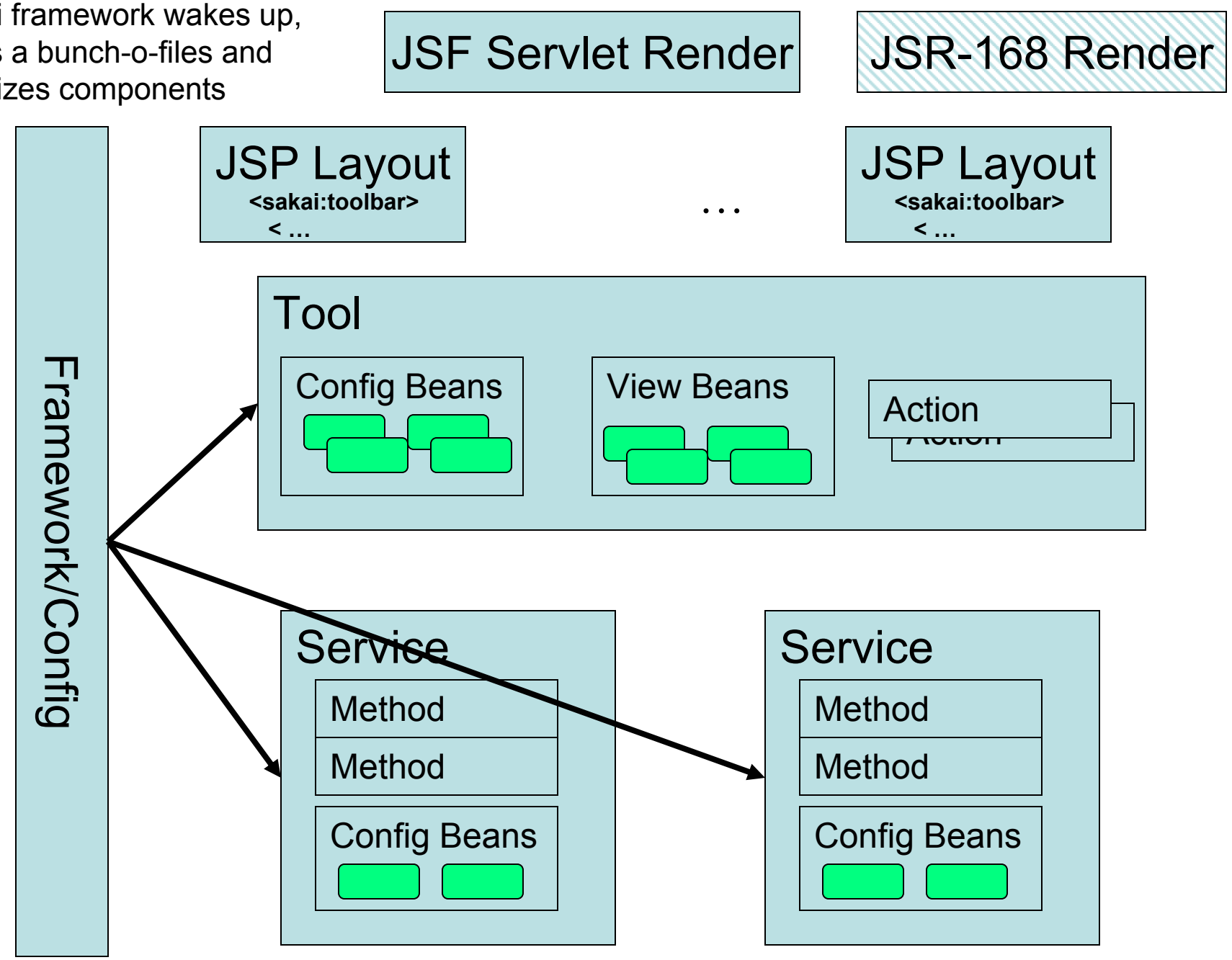
The Sakai User Interface Process



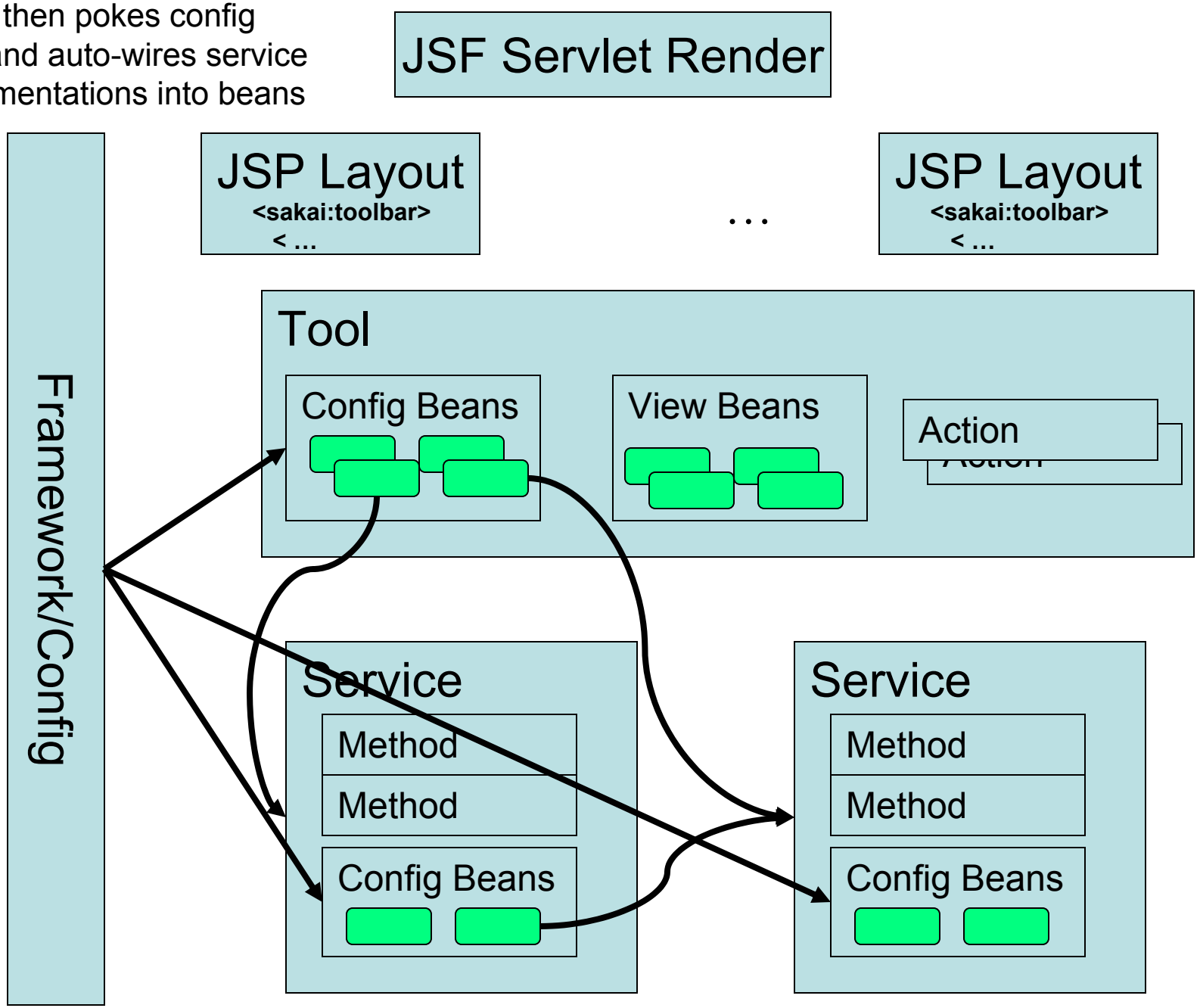
All the components…
On one slide :)



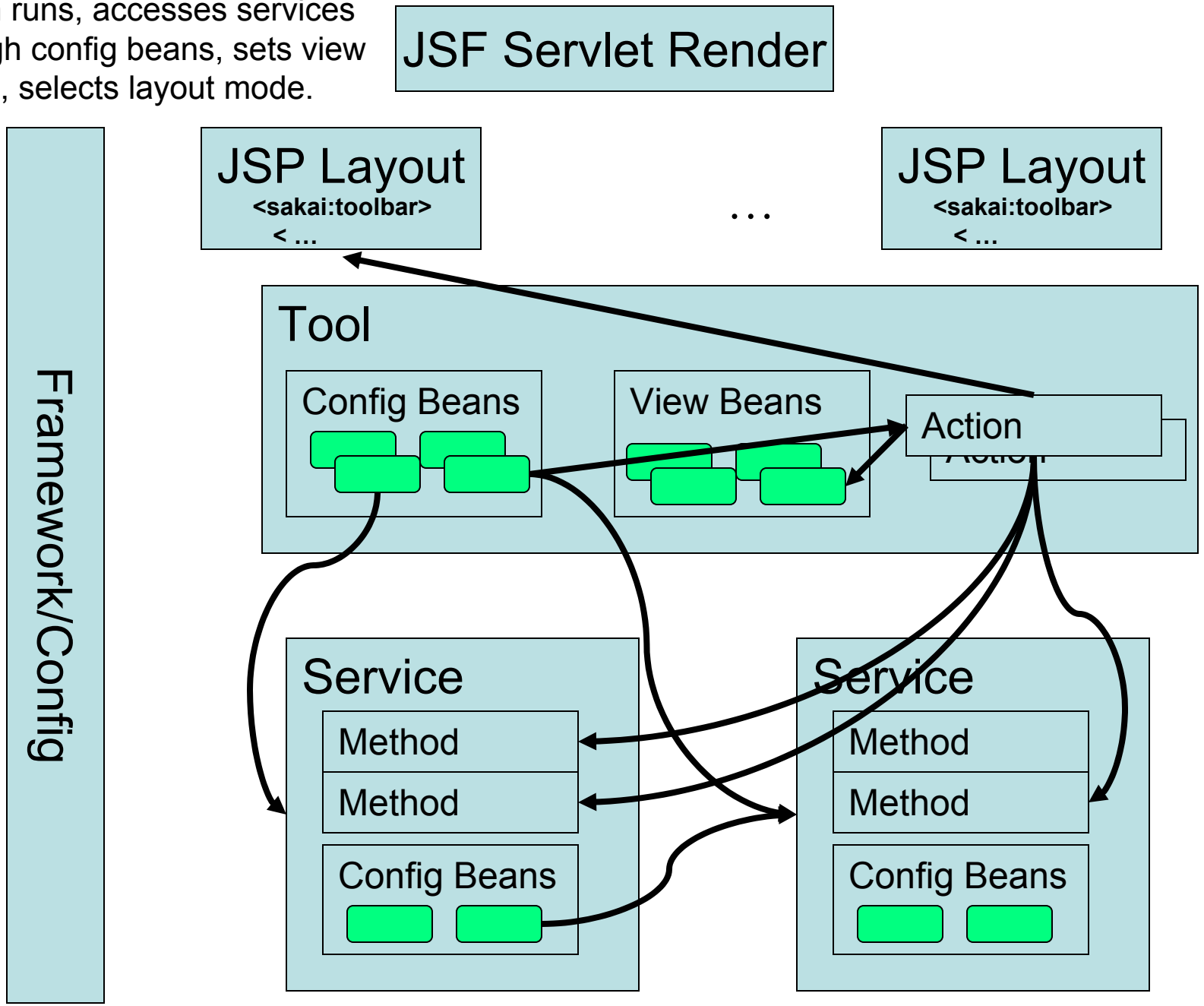
Sakai framework wakes up, reads a bunch-o-files and initializes components



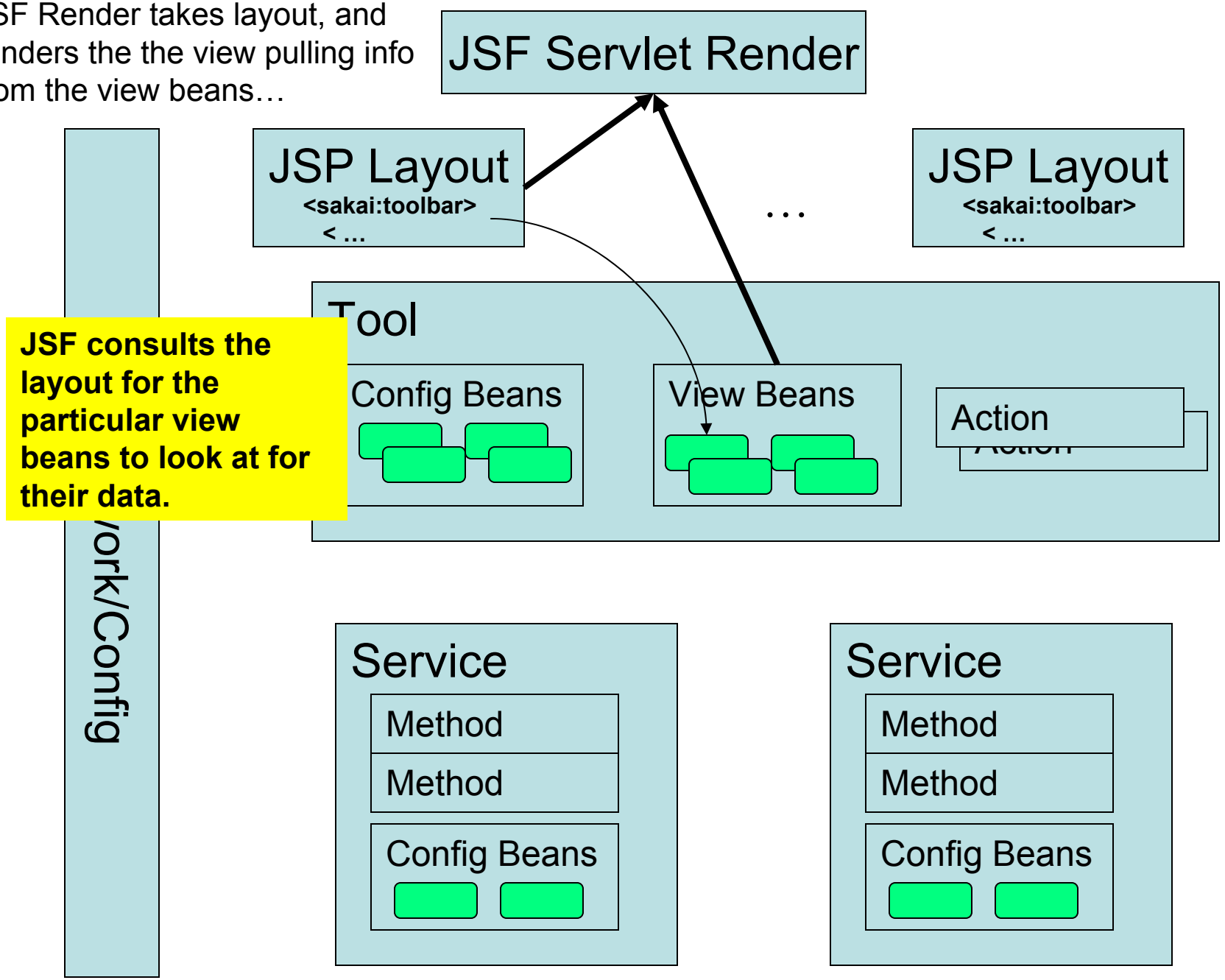
Sakai then pokes config data and auto-wires service implementations into beans



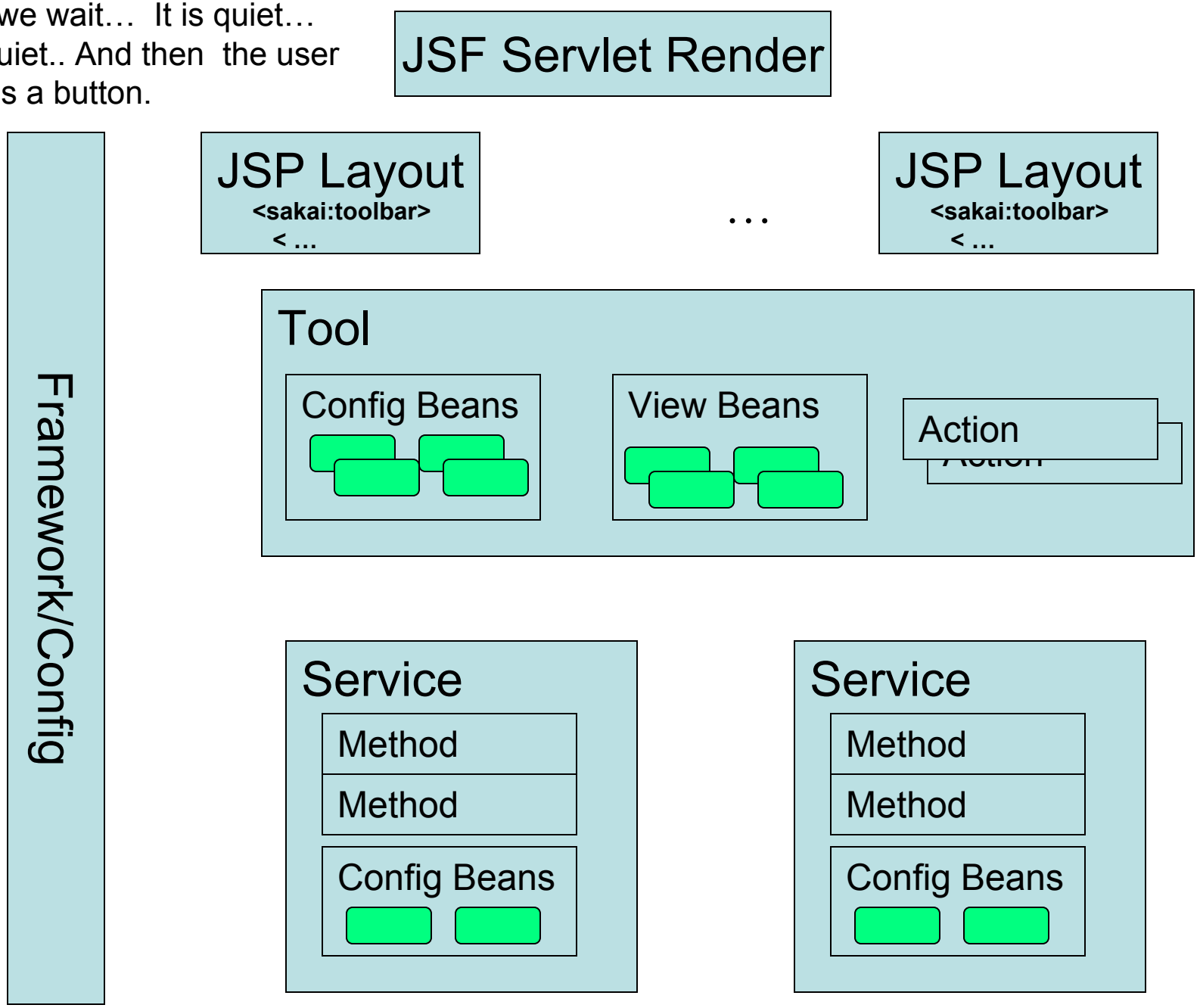
Action runs, accesses services through config beans, sets view beans, selects layout mode.



JSF Render takes layout, and renders the the view pulling info from the view beans…

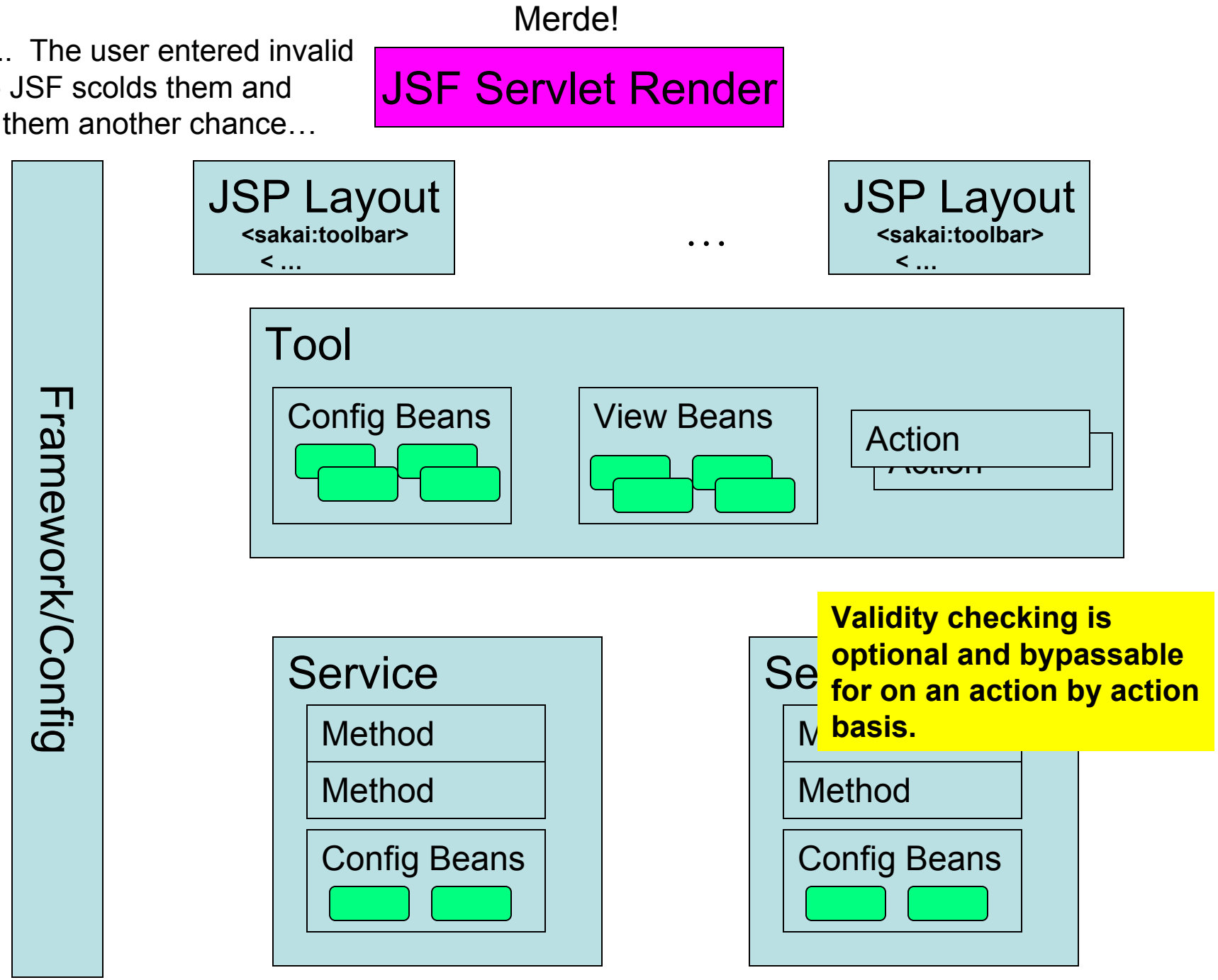


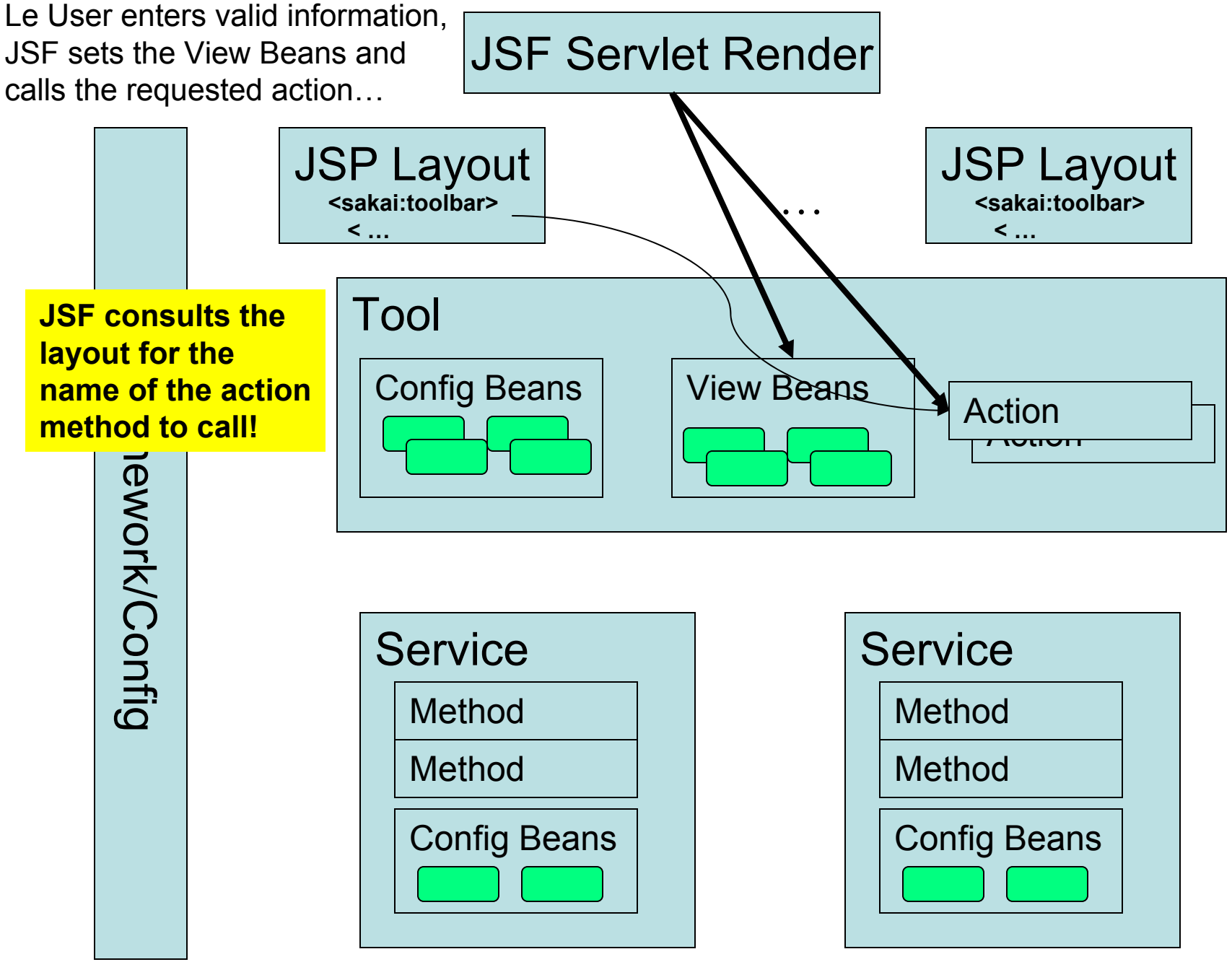
Then we wait… It is quiet… Too quiet.. And then the user pushes a button.



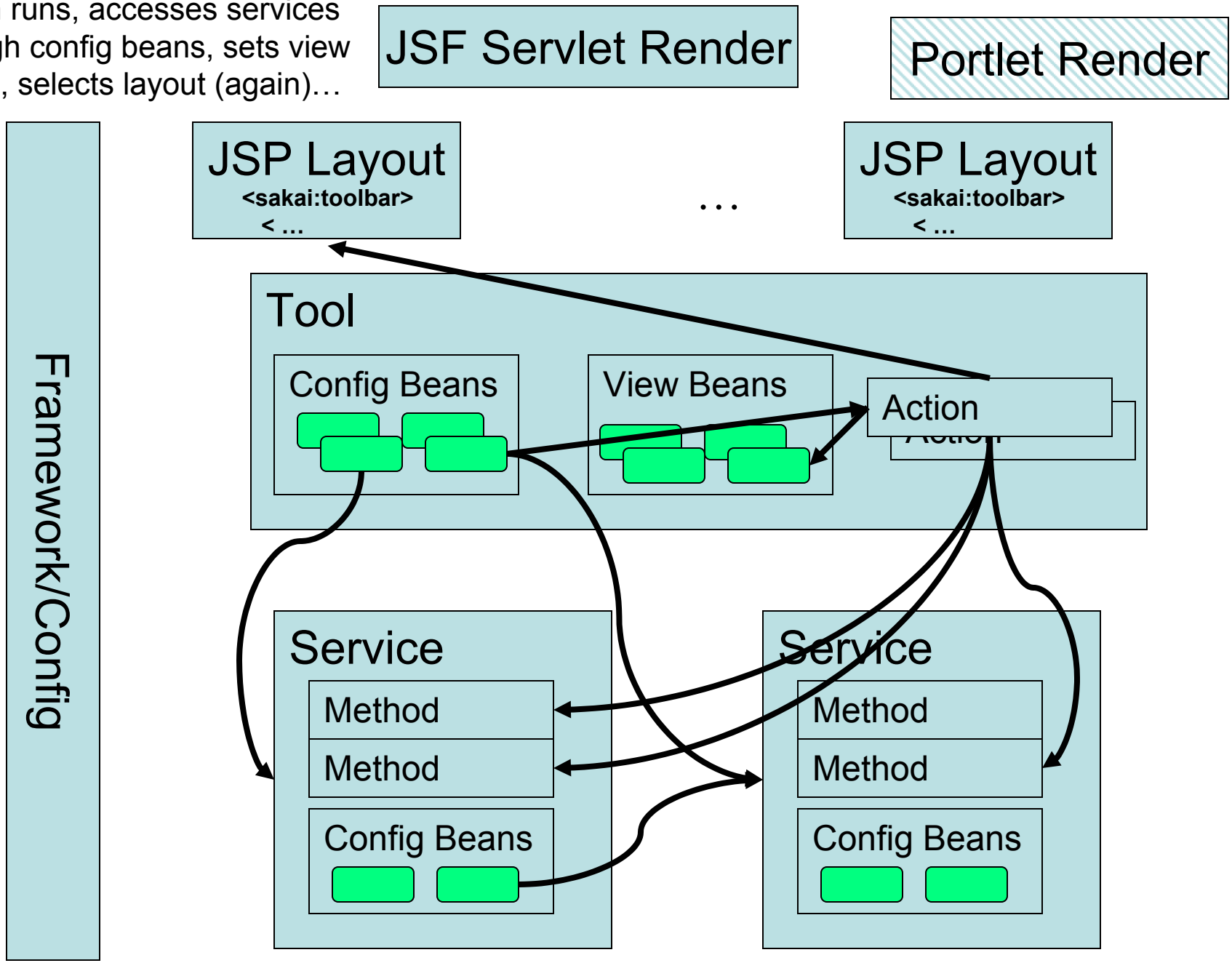
Merde!

Oops.. The user entered invalid data - JSF scolds them and gives them another chance…




Le User enters valid information, JSF sets the View Beans and calls the requested action…
JSF Servlet Render
JSP Layout
<sakai:toolbar>
< ...
JSP Layout
<sakai:toolbar>
< ...
JSF consults the layout for the name of the action method to call!
Tool
Config Beans
View Beans
Action
Service
Method
Method
Config Beans
Service
Method
Method
Config Beans

Action runs, accesses services through config beans, sets view beans, selects layout (again)…



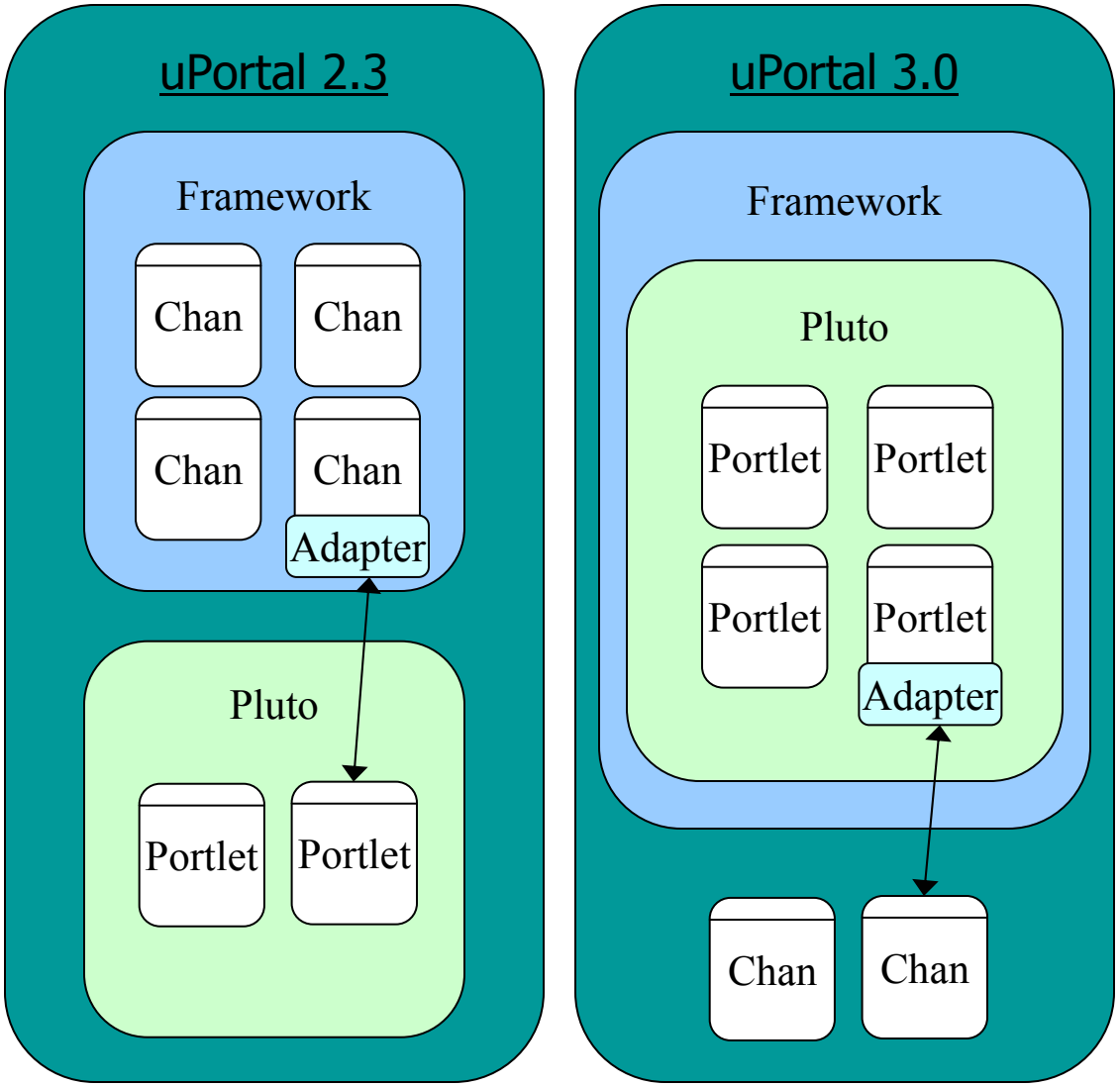
Sakai and uPortal / JSR-168

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

uPortal Portlet Roadmap

- uPortal 2.3
  - Support Portlets (JSR-168) via adapter
- uPortal 3.0
  - Implement Portlet Specification (JSR-168)
  - Support IChannel via adapter



Portal => Application Framework

- Portals are a framework to deploy tools (aka rectangles) and focus on how the user wants to arrange their own “rectangles”
- While Sakai has chosen to use a portal as a component integration technically, the goal is for the tools to work together closely and seem to really be parts of a larger “tool”
- Sakai has a lot of features, (services, presence, notification, etc..) which bridge the gap between portal and application framework

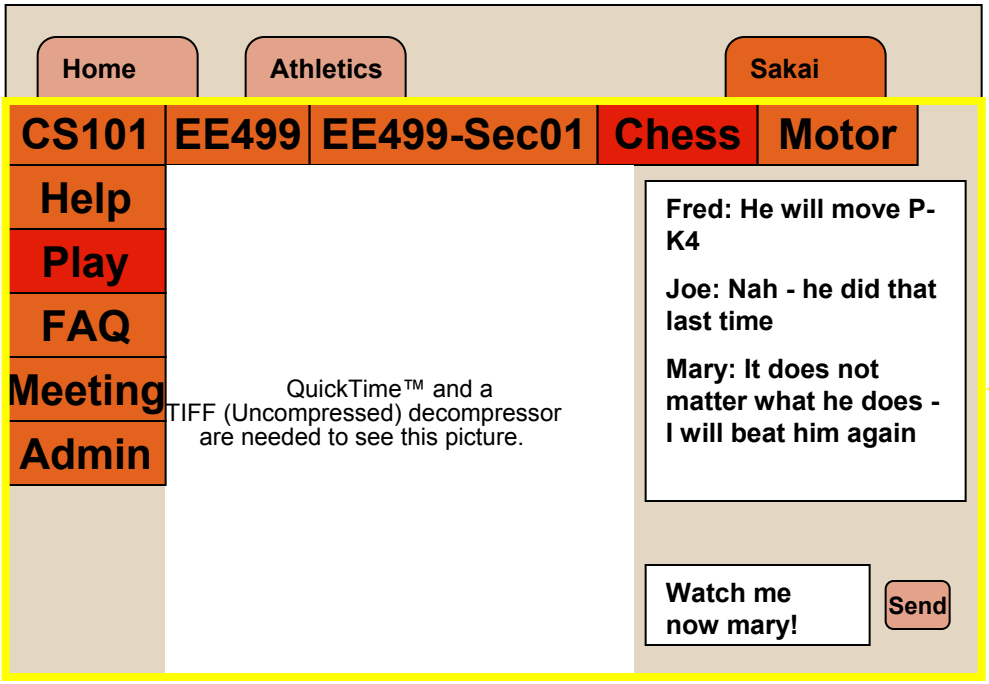
Sakai

Sakai 1.0 and uPortal

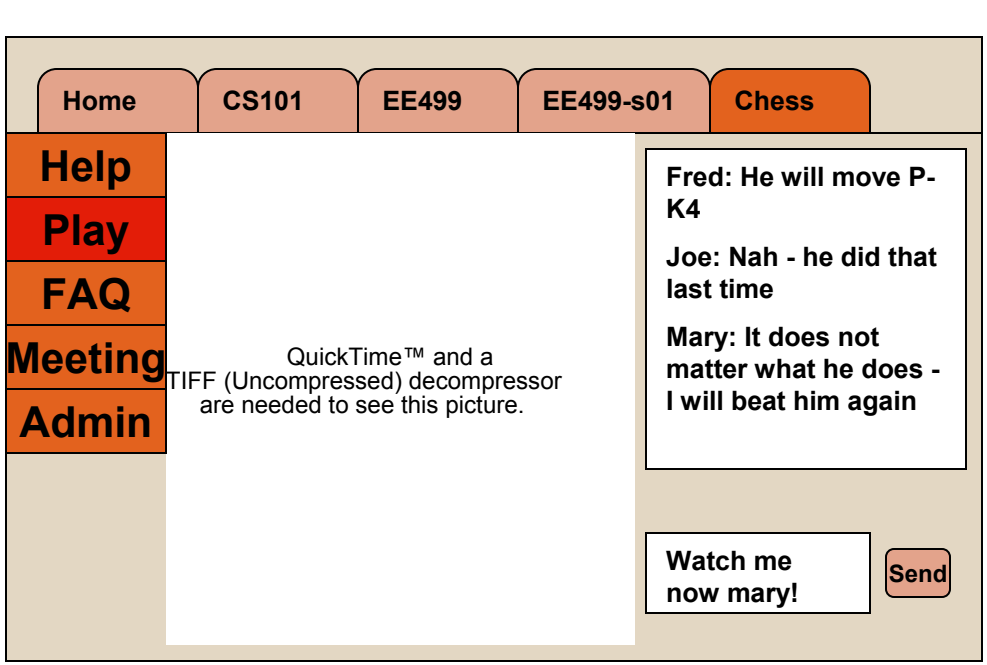
- The embedded version where the entire Sakai tool set appears as a single channel much like the “SuperChannel”. This can be installed in any standard uPortal environment.
- The “injected” version which uses a modified version of uPortal 2.3 with two-level navigation and configuration information coming from Sakai. This is pretty much a stand-alone learning management system using uPortal. The uPortal theme and structure will be altered to precisely display the hierarchical navigation needed by Sakai.

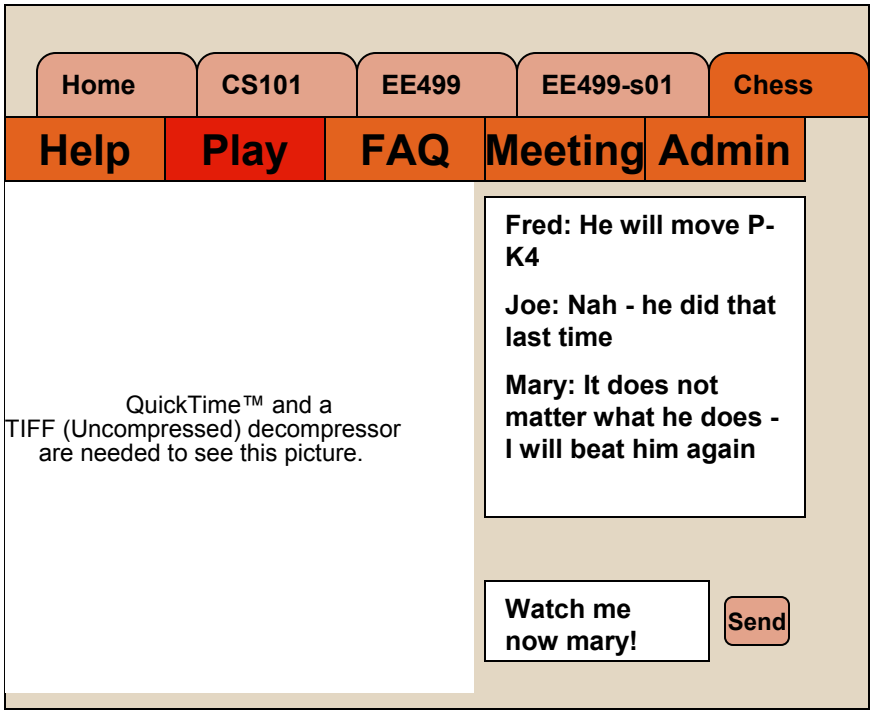
Sakai

Sakai 1.0: Embedded Version (uPortal 2.3)



Sakai 1.0: Injected Version (uPortal 2.3)





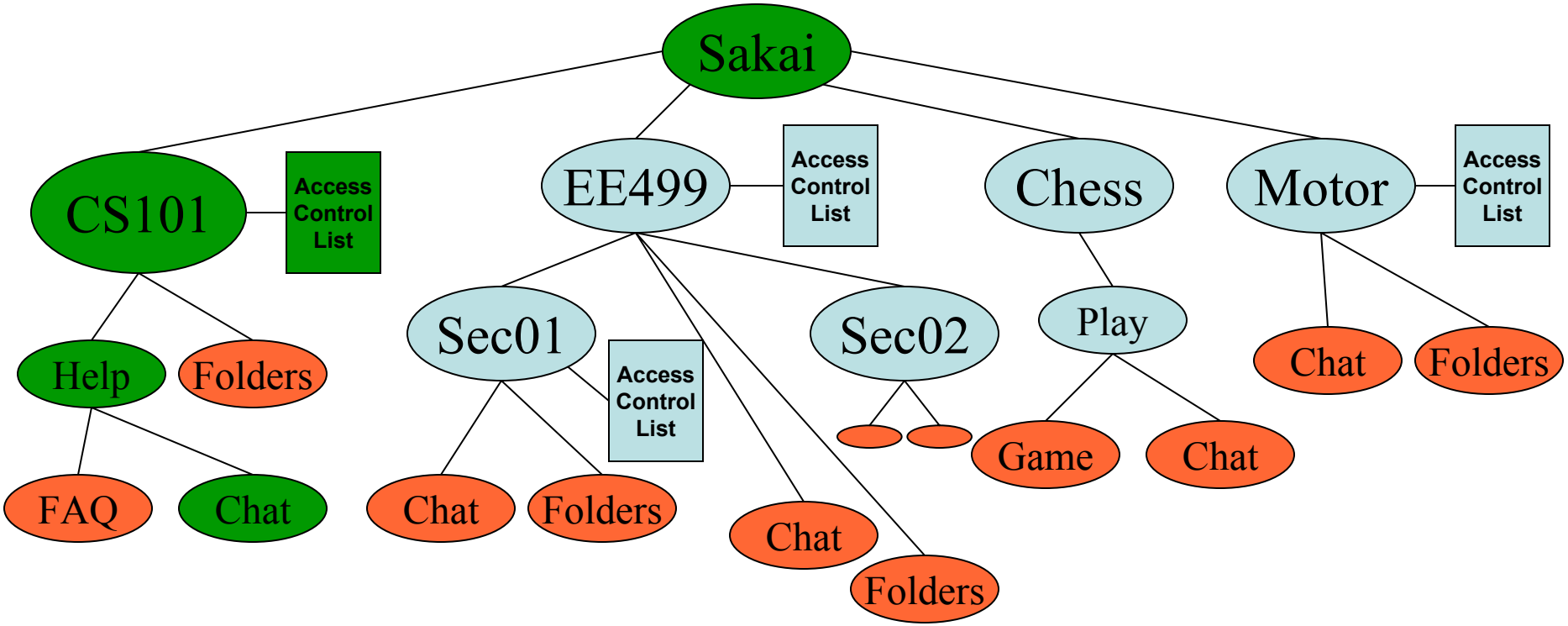
Sakai

Sakai 2.0 and uPortal

- The integrated version where Sakai tools simply are part of the set of channels which can be added to any uPortal environment. By placing a Sakai tool anywhere within the navigation hierarchy of uPortal, it becomes a collaborative element at that location.
- This is more complex than it sounds and as such will only work within uPortal and will require some modifications to uPortal that the Sakai effort is undertaking and contributing to the uPortal project.

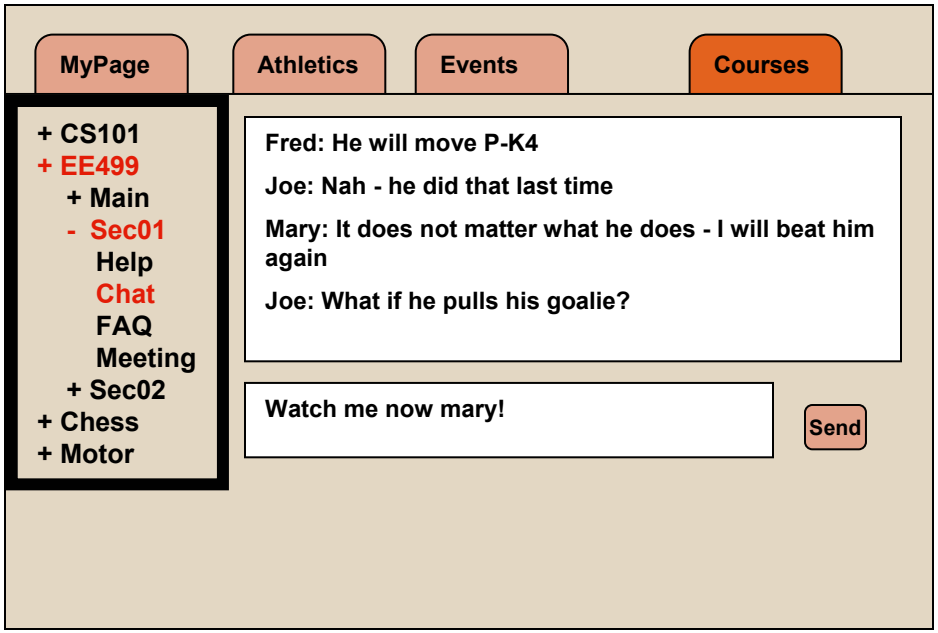
Sakai

The Hierarchy Challenge



Portlets/Channels need to know “where” they fit for inherited access control and to know the “context” in which they operate - “I am the Chat for CS101”. There are fragment administration issues. This is not specified in the JSR-168 spec. SuperChannel and Sakai Embedded are solutions which hide the hierarchy from the portal - but this is less than ideal because it would be nice to drop a context-sensitive “chat” tool anywhere in the portal.

Sakai 2.0: Integrated



Sakai

Advanced Sakai

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

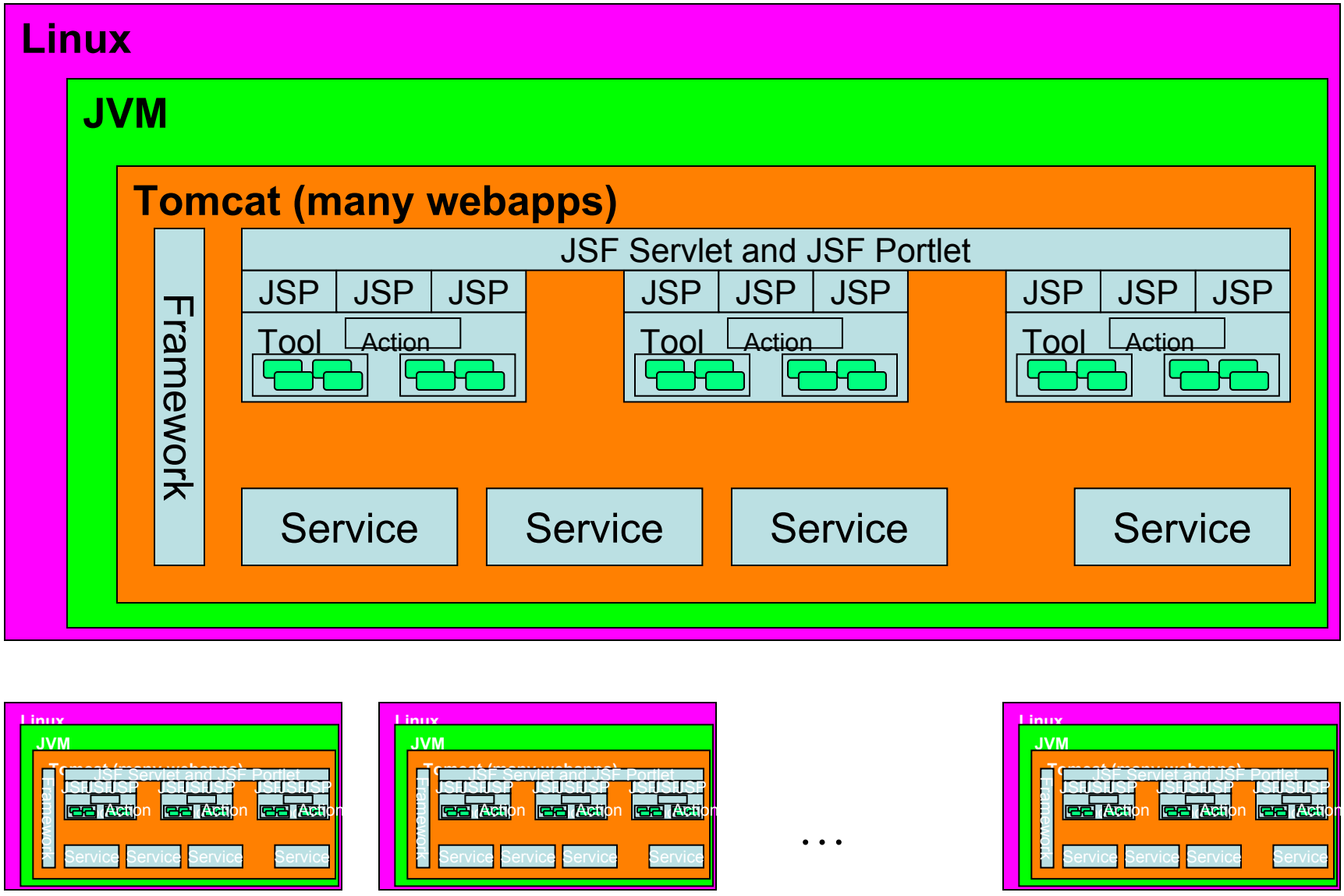
Sakai

Sakai Framework Possibilities

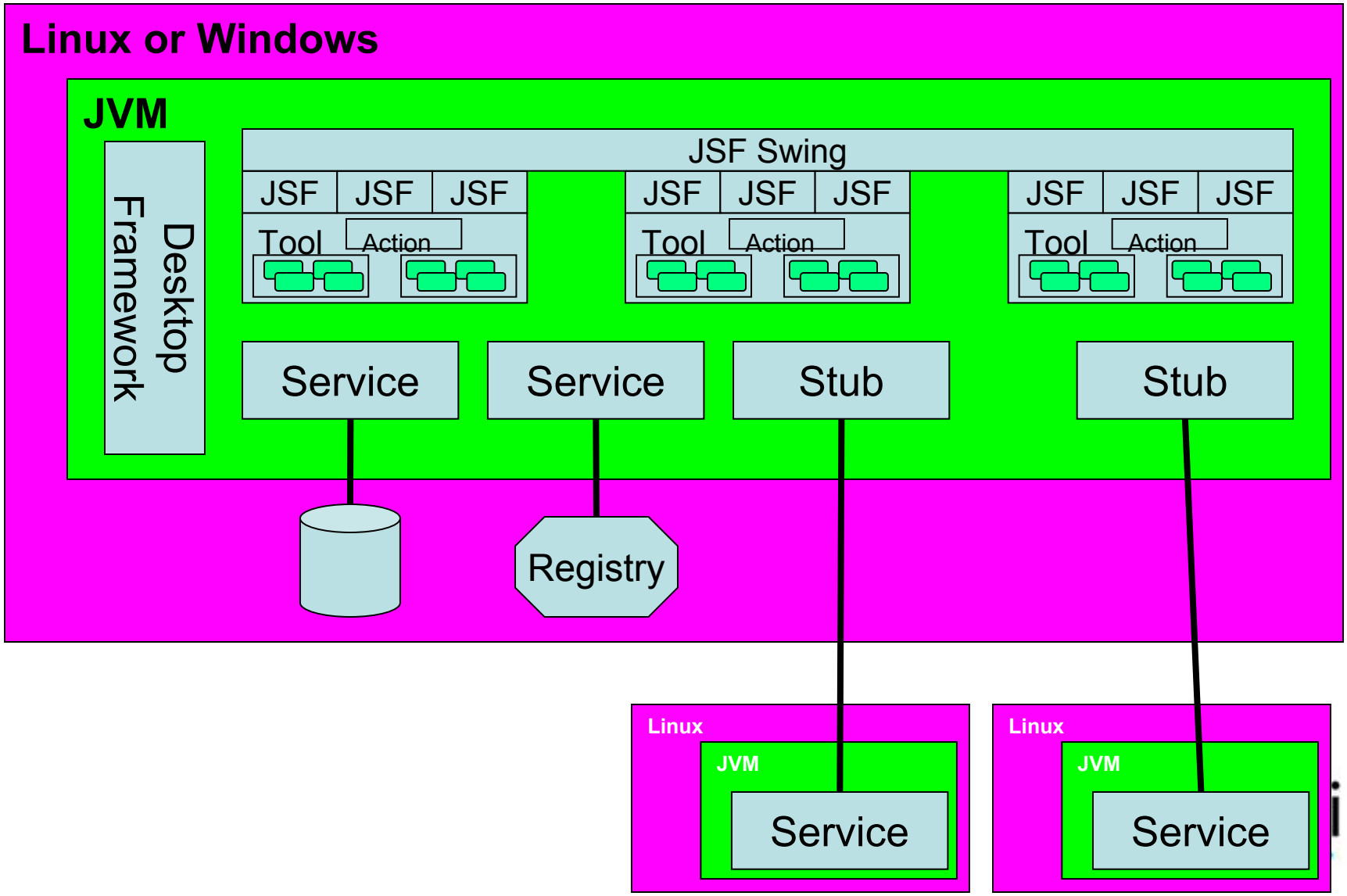
- Web server / Browser
  - This is what we are committed to do in 2 years
- Swing desktop
- Web Services - a few places
- Web Services - everywhere

Sakai

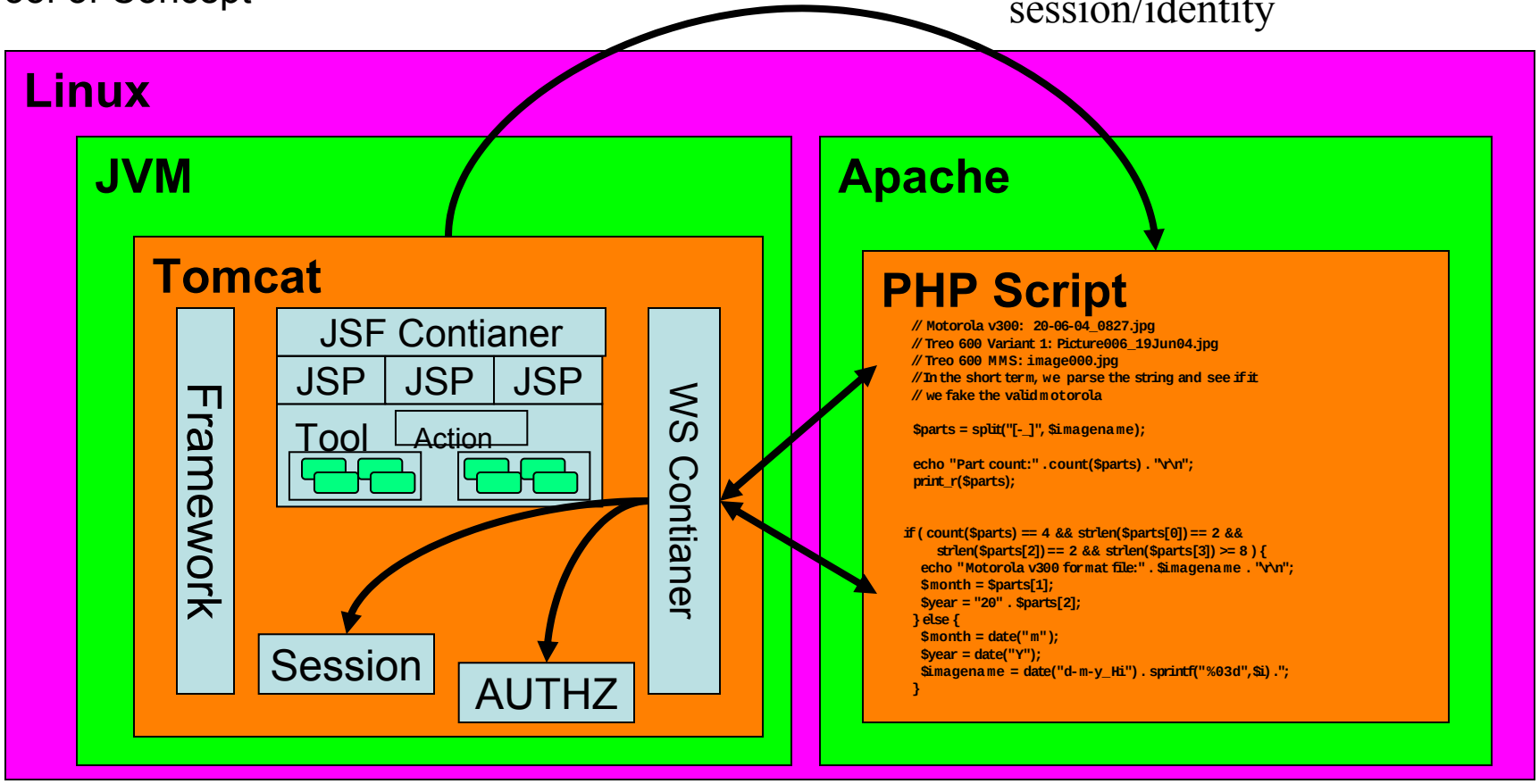
Plan A - Clusters of JVM's JSF implementations from SUN.



What if SUN provided a SWING JSF Render capability
and we hand-build desktop versions of services or
hand-build stubs which used ROMI or Web Services?

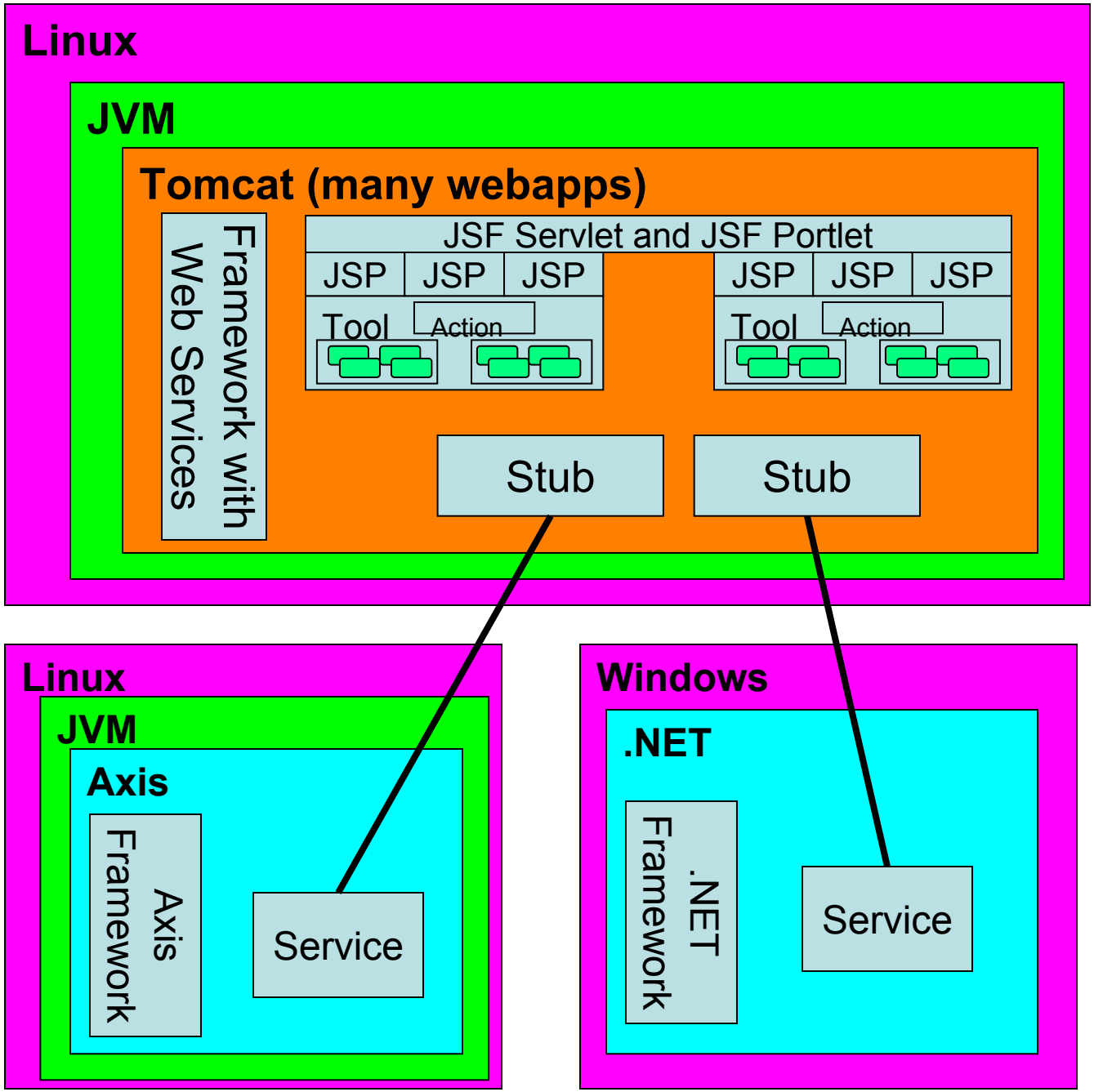


Web Services and
Cross Language
Proof of Concept



•How do we launch non-JAVA elements passing in basic identity, session information? WSRP? Ad hoc? cWebProxy? Do we include a back-reference handle to help resolve web services?
•How do we secure the web-services calls? What languages support
•Do we refactor services? Do we we implement that critical subset? Do we end up with new methods that are "yucky" but well-suited for web-services?
•Do we hide things in PHP behind an API with methods? Or do we just lay down some web services code in the PHP?

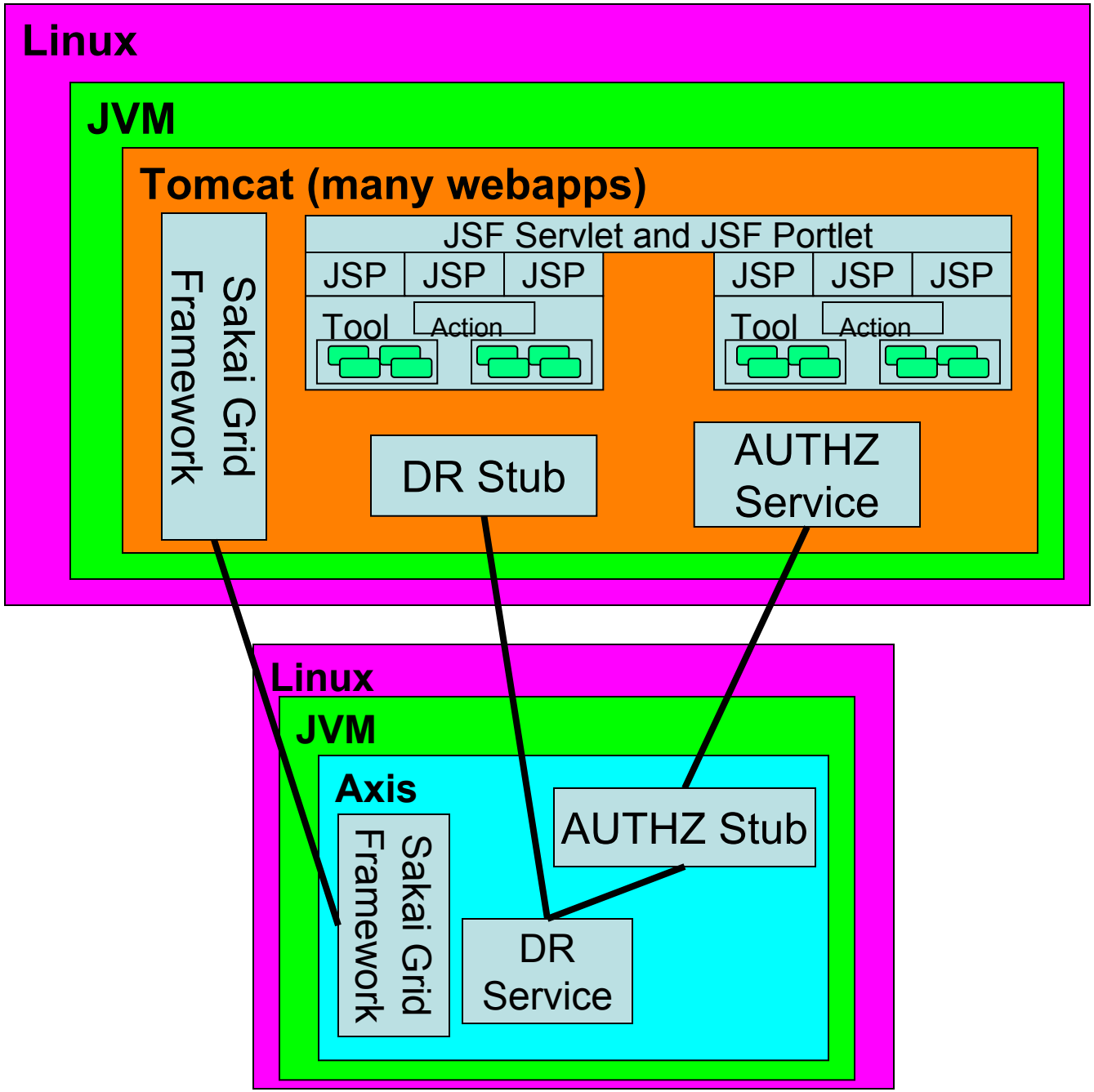
Sakai



What if we built a version of the framework that examined an interface using reflection and dynamically built a proxy, generated WSDL, and just made web service happen pretty much transparently other than declaring where services were to run in configuration…

Hopefully there will be a mechanism for secure web services, or perhaps we could simply use two-way SSL certificate exchange to force transport security…

Sakai



What if the frameworks were highly coordinated and in addition to dynamically generating stubs and placing web services, provided standardized mechanism for moving identity securely across web services, and the frameworks could perform dependency injection automatically when one service had a dependency on a service running on another server…

Hmmm. Sounds like the Grid.

Sakai

Why wait for web services?

- Our APIs and services will not really be mature until early 2005 - we may have to do major re-factoring as our code base grows and problems are identified
- Secure, identity preserving web services at a distance seem to be churning every 6 months.
- We are committed to deliver a full-featured high-performance product in the Java / Web Server / Browser space in two years.
- We don't have time to be the "tip-of-the-spear" on tracking every single web-service technology twitch.
- Web services are great fun for "point solutions" but are painful as a basis for a framework right now

Sakai

Why start on web services?

- Short term: Sakai API implementations can use Web Services hidden behind the API (collecting point solutions)
- Web services are changing right now
  - WSRF - Web Services Resource Framework
  - Generic Security Services Application Program Interface (GSS-API) defined in RFC 2853 and JDK 1.4.2
- Service Injection means that it is “Possible” to build a Sakai Web-Services Framework without changing services code.

Sakai

Summary

- This a journey - we are just at the beginning
- Thank you for your time
- Sakai is a well funded effort which will product a portal framework which will support both basic JSR-168 portlets and Sakai-style portlets as well.

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

QuickTime™ and a
TIFF (Uncompressed) decompressor
are needed to see this picture.

Sakai